UNIVERSITÉ PARIS-EST

ÉCOLE DOCTORALE : MATHÉMATIQUES ET SCIENCES
ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

Thèse de doctorat

SPÉCIALITÉ : MATHÉMATIQUES APPLIQUÉES

présentée par

François PACAUD

Decentralized Optimization Methods for Efficient Energy Management under Stochasticity

---

Thèse préparée au CERMICS, École des Ponts ParisTech

Soutenue le 25 Octobre 2018 devant le Jury composé de :

JURY

Rapporteurs :	Frédéric Bonnans	École Polytechnique
	Andy Philpott	University of Auckland
Directeurs :	Pierre Carpentier	ENSTA ParisTech
	Michel De Lara	École des Ponts ParisTech
Examinateurs :	Nadia Oudjane	EDF R&D
	Nicolas Petit	Mines ParisTech
	R. Tyrrell Rockafellar	University of Washington
	Andrzej Ruszczyński	Rutgers University

Abstract

New energy systems are designed to absorb a large share of renewable energy in a decentralized fashion. Their optimized management raises specific issues. We study mathematical formulation as large scale multistage stochastic optimization problems. We focus on time and space decomposition methods in a stochastic setting.

In the first part of this manuscript, *Time decomposition in optimization and management of home microgrids*, we apply stochastic optimization algorithms to the management of small scale microgrids. We compare different optimization algorithms on two examples: a domestic microgrid equipped with a micro Combined Heat and Power generator and a battery; a domestic microgrid equipped with a battery and solar panels.

In the second part, *Mixing time and spatial decomposition in large-scale optimization problems*, we extend the previous studies to larger microgrids, where different units and storage devices are connected together. As a direct resolution by Dynamic Programming of such large scale systems is not tractable, we propose original algorithms mixing time decomposition on the one hand, price and resource spatial decompositions on the other hand.

In the third part, *Contributions to Stochastic Dual Dynamic Programming*, we focus on the Stochastic Dual Dynamic Programming (SDDP) algorithm, a well-known algorithm to solve multistage stochastic optimization problems. We present a new stopping criteria based on a dual version of SDDP which gives a deterministic upper-bound for the primal problem.

Résumé

Les réseaux électriques doivent absorber une production d'énergie renouvelable croissante, de façon décentralisée. Leur gestion optimale amène des problèmes spécifiques. Nous étudions dans cette thèse la formulation mathématique en tant que problèmes d'optimisation stochastique à plusieurs pas de temps. Nous analysons plus spécifiquement la décomposition en temps et en espace de tels problèmes.

Dans la première partie de ce manuscrit, *Décomposition temporelle pour l'optimisation de la gestion de microgrids domestiques*, nous appliquons les méthodes d'optimisation stochastique à la gestion de microgrids de petite taille. Nous comparons différents algorithmes d'optimisation sur deux exemples : le premier considère une microgrid domestique équipée avec une batterie et une centrale de micro-cogénération ; le deuxième considère quant à lui une autre microgrid domestique, cette fois équipée avec une batterie et des panneaux solaires.

Dans la seconde partie, *Décomposition temporelle et spatiale de problèmes d'optimisation de grande taille*, nous étendons les études précédentes à des microgrids de plus grande taille, combinant ensemble différentes unités et moyens de stockage. La résolution frontale par Programmation Dynamique de tels problèmes de grande taille s'avère généralement impraticable. Nous proposons deux algorithmes originaux pour pallier ce problème en mélangeant une décomposition temporelle par programmation dynamique avec une décomposition spatiale — par les prix ou par les ressources.

Dans la dernière partie, *Contributions à l'algorithme Stochastic Dual Dynamic Programming*, nous nous concentrons sur l'algorithme *Stochastic Dual Dynamic Programming* (SDDP) qui est actuellement une méthode de référence pour résoudre des problèmes d'optimisation stochastique à plusieurs pas de temps. Nous étudions un nouveau critère d'arrêt pour cet algorithme basé sur une version duale de SDDP, qui permet d'obtenir une borne supérieure déterministe pour le problème primal.

Remerciements

La route a été longue, et nombreuses sont les personnes sans qui je n'aurais pu arriver au bout du chemin.

Tout d'abord, j'adresse mes plus vifs remerciements à mes deux encadrants de thèse, Pierre Carpentier et Michel De Lara, ainsi qu'à Jean-Philippe Chancelier. Sans leurs précieux conseils — allant des mathématiques à la psychologie comportementale en passant par la preuve formelle de programme — il est évident que je n'aurais pu mener ce travail à bien. Leur soutien indéfectible et leur grande rigueur auront permis de mâtiner mes travaux de cette indispensable touche entomologiste. Je tiens aussi à remercier l'ensemble des chercheurs que j'ai eu la chance de côtoyer durant ces trois années de thèse. Tout d'abord, je remercie Frédéric Bonnans et Andy Philpott qui m'ont fait l'honneur de rapporter cette thèse. J'accorde toute ma gratitude à l'ensemble des membres du jury pour avoir été présents le jour de ma soutenance, ainsi que pour leur fructueuses remarques. Je tiens à remercier particulièrement Vincent Leclère pour ses conseils prodigués tout au long de la thèse : il fût auprès de moi un aîné avisé qui su me guider à travers les méandres marécageux de l'optimisation stochastique. Je voudrais aussi saluer Nadia Oudjane et Arnaud Lenoir du département OSIRIS d'EDF R&D pour leurs discussions mathématiques et les échanges instructifs que nous avons pu avoir autour de l'application des méthodes d'optimisation dans le monde de l'énergie. De surcroît, je ne peux que tirer humblement mon chapeau aux anciens doctorants de l'équipe optimisation et systèmes pour avoir éclairé le chemin devant moi. Je salue aussi Pierre Haessig dont la thèse a été un sujet d'inspiration. Enfin, je remercie Lionel et Nicolas pour avoir su me convaincre de faire une thèse à une époque où le monde de la recherche était encore pour moi une vaste contrée inexplorée.

J'aimerais remercier la communauté open-source pour avoir développé les outils informatiques sans lesquels cette thèse n'aurait été que l'ombre d'elle-même. Je remercie notamment l'ensemble des développeurs ayant contribués au projet GNU/Linux, ainsi que Donald Knuth pour avoir développé un outil aussi performant que LaTeX. Par ailleurs, j'adresse ma vive reconnaissance à l'ensemble de la communauté gravitant autour du langage Julia. Je pense en particulier à la communauté JuliaOpt : un grand merci aux développeurs de JuMP sans qui la plupart des algorithmes présentés ci-après n'auraient pu être implémentés aussi rapidement. Je remercie notamment Oscar Dowson et Benoît Legat pour les nombreuses et fructueuses discussions que nous avons pu avoir au sujet de de l'algorithme SDDP.

Je remercie l'ensemble du personnel du CERMICS pour l'atmosphère studieuse de ce laboratoire. Je remercie notamment Isabelle et Fatna pour leur aide précieuse en matière d'administration, ainsi que Frédéric Meunier et Bernard Lapeyre pour m'avoir permis d'enseigner dans leurs cours respectifs. Mes pensées vont aussi à l'ensemble des doctorants, du 2e comme du 3e étage. Je pense notamment à Alexandre, Étienne, Marion, Grégoire, Julien, sans oublier mon frère de thèse Henri.

Du côté Bienvenuë, je souhaite remercier chaleureusement l'ensemble des camarades transitants pour m'avoir accompagnés au cours de cette thèse. Je remercie Romain pour nos exercices prospectifs communs, Emmanuel pour ses suggestions musicales éclairées, Alessio pour son calme inébranlable, Mathieu A. pour savoir rester en décalage. Je remercie dans la même veine les lurons Pierre et Pimpin, ainsi que Marina pour ses judicieux conseils d'ex-doctorante chevronnée. Enfin, j'adresse une pensée spéciale pour mon compagnon de bureau Tristan : que ce soit autour d'une bière ou d'un terminal, nous aurons passé nos trois années de thèse à nous épauler mutuellement.

J'adresse une pensée à l'ensemble des amis qui m'ont accompagné durant ces trois années. Je remercie Claire, Hella et Mathieu B. pour les dîners et nos débats transdisciplinaires. Je remercie

Jean, avec qui il est possible de filtrer le post-modernisme par ce bon vieux Kalman. Je remercie Mike pour les pigeons, les picons et pour son indéfectible soutien pendant la rédaction de cette thèse. Je pense aussi à Clément, Hélène, Jean Bernard, Julien, Loïc, Ricardo et Thibaut pour les nombreuses soirées passées ensemble. J'ai une pensée particulière pour Paul, fidèle compagnon de route, pour Grégoire et nos nombreuses discussions à tiroir ainsi que pour Thomas qui a toujours su rester à mes côtés au cours de mes nombreuses élucubrations aériennes.

Finalement, je souhaite remercier du fond du coeur ma famille pour toute l'aide qu'elle a pu m'apporter pendant mes nombreuses années d'études. Je remercie mes frères Benoît et Vincent pour leur soutien fraternel. Enfin, et surtout, je remercie mon père pour avoir ouvert mon horizon, et ma mère pour m'avoir donné la force de garder le cap.

Contents

Contents detailed

Chapter 1.

Introduction (version française)

Contents

Cette thèse se situe à l'intersection entre l'optimisation mathématique et le domaine de l'énergie. Pour commencer, nous donnons des éléments de contexte autour du travail présenté dans ce manuscrit, puis nous présentons une revue concernant les systèmes de gestion optimisés de l'énergie ainsi qu'une revue de littérature sur les méthodes d'optimisation stochastique. Enfin, nous détaillons la structure du manuscrit.

1.1. Contexte

Une fois le courant continu (défendu par Edison) remplacé par le courant alternatif (défendu par Tesla), la structure des réseaux électriques a évolué d'une structure décentralisée — avec des centrales situées proches des consommateurs — vers une structure centralisée — avec des centrales de plus en plus puissantes situées à l'écart des villes. Cette centralisation s'est poursuivie pendant le 20^e siècle, avec l'extension des réseaux de transmission de l'électricité.

Avec la diffusion croissante des énergies renouvelables, le débat entre réseau centralisé et réseau décentralisé refait surface. Un premier groupe considère que l'extension croissante des réseaux — jusqu'à des échelles continentales — permettra de distribuer plus efficacement l'électricité vers les consommateurs finaux — tout en permettant de mutualiser les risques. D'un autre côté, un deuxième groupe pense qu'il faudrait revenir à un réseau plus décentralisé, où l'énergie est consommée localement, directement sur le lieu de production. Ceci permettrait, selon eux, de diminuer les pertes induites par le transport de l'électricité sur de longues distances, tout en ayant une structure plus résiliente en cas de panne.

Nous étudions ici des petits réseaux de distribution, tels que défendus par le deuxième groupe. Ces réseaux rassemblent un ensemble de consommateurs et de producteurs pouvant produire leur propre énergie (par exemple via des panneaux solaires). De tels réseaux sont appelés *microgrids*, pour les distinguer des *smart grids*. L'étude de tels réseaux décentralisés connaît un regain d'interêt

en France depuis un changement legislatif paru en janvier 2017, permettant à différents producteurs locaux de mutualiser leurs moyens de production. Malgré leur petite taille, ces microgrids urbaines restent des systèmes complexes sujets à de nombreux aléas (demandes locales, productions renouvelables) pouvant rassembler un nombre potentiellement importants de systèmes physiques différents.

Ce travail s'inscrit au sein d'une collaboration entre Efficacity — institut français pour la transition énergétique — et le CERMICS (Centre d'Enseignement et de Recherche en Mathématiques et Calcul Scientifique), le laboratoire de mathématiques appliquées de l'école des Ponts. Cette collaboration cherche plus particulièrement à étudier la gestion optimale des microgrids urbaines.

Nos contributions principales portent avant tout sur la gestion des incertitudes au sein des microgrids urbaines, afin d'améliorer la résilience de tels systèmes. Nous étudions les aspects mathématiques posés par la gestion optimale des microgrids. Nous étudions dans un premier temps des microgrids de petite taille, dont le réseau regroupe une seule maison, avant d'étendre ce travail à des microgrids de plus grande taille, rassemblant jusqu'à une cinquantaine de maisons. Nous cherchons à utiliser des méthodes d'optimisation stochastique pour gérer le système de façon optimale.

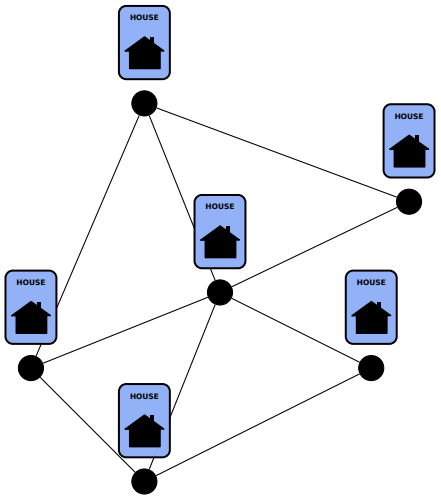


Figure 1.1.: Autoconsommation collective à l'échelle du quartier

1.2. Gestion optimale des microgrids

Un nombre conséquent de travaux ont été consacrés récemment à la gestion optimale des flux énergétiques dans les microgrids. Nous détaillons ci-après les différents défis posés par la gestion optimale de tels systèmes, en détaillant les travaux de référence sur le sujet.

1.2.1. Gestion de l'énergie

Nous nous intéressons avant tout à l'*energy management system* (EMS) des microgrids urbaines. À un instant donné, l'EMS doit gérer les différents flux énergétiques transitant au sein de la microgrid tout en cherchant à minimiser un certain nombre de coûts opérationnels. Les flux énergétiques optimaux calculés par l'EMS sont ensuite utilisés comme référence par un contrôle plus fin qui doit gérer de manière effective les différentes puissances électriques au sein de la microgrid.

Les principaux défis posés par la gestion optimale des EMS sont les suivants.

- L'EMS ne connaît pas à l'avance les futures réalisations des incertitudes (que ce soit les demandes ou les productions renouvelables), ce qui complique le calcul des flux optimaux permettant une adéquation parfaite entre la production et la demande.
- Plus les incertitudes sont importantes, plus la microgrid est sujette à un risque de panne. L'EMS doit prendre des décisions *robustes* par rapport aux événements extrêmes (par exemple une demande électrique élevée).
- Les microgrids peuvent rassembler une dizaine de stocks, potentiellement de nature hétérogènes. Les problèmes peuvent ainsi avoir une dimension très élevée, ce qui complique la résolution par les solveurs d'optimisation classiques.
- Les systèmes énergétiques sont naturellement complexes. La production d'énergie peut ne pas être modulable, ou les différents équipements peuvent avoir des contraintes opérationnelles complexes.

1.2.2. Optimisation de la gestion de l'énergie

La gestion optimale des microgrids a soulevé beaucoup d'intérêt récemment. Dans (Olivares et al., 2014), les auteurs donnent ainsi une revue détaillée de l'optimisation de la gestion des flux énergétiques au sein des microgrids. Les microgrids sont actuellement surtout utilisées pour gérer des réseaux isolés ou des systèmes auto-suffisants (Olivares et al., 2011)-(Heymann et al., 2015). Même si les approches stochastiques ou robustes sont de plus en plus étudiées, les méthodes de référence pour le contrôle des EMS restent des méthodes déterministes, comme l'algorithme *Model Predictive Control* (MPC) (Garcia et al., 1989).

En dehors des systèmes isolés, le contrôle des microgrids urbaines est aussi un champ de recherche en pleine extension. Les systèmes peuvent être très hétérogènes, MPC étant aussi bien utilisé pour contrôler des bâtiments résidentiels (Oldewurtel et al., 2012) que pour contrôler des bâtiments de bureaux (Lamoudi, 2012). Dans (Parisio et al., 2015), les auteurs utilisent le même algorithme MPC pour contrôler cette fois-ci un ensemble de bâtiments résidentiels, cette étude ayant été étendue à des groupements conséquents de maisons par des méthodes de décomposition-coordination (Pflaum et al., 2014).

Gérer les incertitudes par l'optimisation stochastique. À l'échelle locale, les demandes électriques et les productions peuvent être très variables, d'autant plus que les microgrids peuvent produire une grande part de leur énergie par des moyens renouvelables, par nature intermittents. Le développement récent des méthodes de prévision probabiliste des incertitudes (Morales et al., 2013) a conduit à un intérêt croissant pour appliquer les méthodes d'optimisation stochastique à la gestion des microgrids. Dans (Oldewurtel, 2011), les auteurs utilisent ainsi une version stochastique de l'algorithme MPC pour gérer le système en considérant différents scénarios d'incertitudes. D'autres approches utilisent des méthodes d'optimisation robuste pour gérer les incertitudes en considérant des scénarios correspondant à un pire des cas. Nous faisons référence à (Paridari et al., 2016) pour un cas d'étude portant sur des bâtiments résidentiels, et à (Wytock et al., 2017) pour l'application d'une version robuste de MPC à la gestion de bâtiments intelligents.

Avec l'actuel engouement pour les méthodes d'apprentissage automatique, les méthodes d'apprentissage par renforcement commencent aussi à être étudiées pour le contrôle des EMS. Nous faisons référence à (Ernst et al., 2009) pour une comparaison de MPC avec des méthodes d'apprentissage par renforcement.

En dehors de la gestion des microgrids, les méthodes d'optimisation stochastique ont été utilisées par ailleurs pour la gestion des systèmes énergétiques (De Lara et al., 2014). Historiquement, l'optimisation stochastique a d'abord été appliquée à la gestion de vallées hydrauliques (Pereira and Pinto, 1991). D'autres applications ont ensuite été trouvées, comme l'intégration optimale de

l'énergie éolienne (Haessig, 2014) ou encore la gestion de microgrids isolées (Heymann et al., 2016). Les méthodes d'affectation de production ont aussi pendant longtemps été un cas d'application classique de l'optimisation stochastique (Carpentier et al., 1996). Plus globalement, nous faisons référence à (Wallace and Fleten, 2003) pour une revue concernant l'application des méthodes d'optimisation stochastique aux systèmes énergétiques.

1.3. À propos de l'optimisation stochastique

Les méthodes d'optimisation stochastique cherchent à minimiser un critère dont l'expression dépend explicitement de variables aléatoires. Ces méthodes se situent à l'intersection entre l'optimisation et la théorie des probabilités.

Il existe plusieurs manières de prendre des décisions dans un cadre incertain. Dans l'optimisation en boucle ouverte, le preneur de décision prend ses décisions une fois pour toute, sans aucun recours possible par rapport aux incertitudes futures. Au contraire, les méthodes d'optimisation en boucle fermée cherchent à adapter leurs décisions au fur et à mesure de la réalisation des incertitudes.

Dans les EMS, la plupart des problèmes se formulent comme un problème d'optimisation stochastique à plusieurs pas de temps (Carpentier et al., 2015), où les incertitudes arrivent à différent instants. Nous nous concentrons ici sur la résolution de tels problèmes d'optimisation.

1.3.1. Programmation Stochastique

Les méthodes de programmation stochastique modélisent les futures réalisations des incertitudes comme un arbre de scénarios, où chaque nœud représente la réalisation d'un aléa donné. Une décision optimale est affectée à chacun des nœuds de l'arbre, de manière à minimiser un certain critère. Nous faisons référence à (Shapiro et al., 2009) et à (Kall et al., 1994) pour une introduction plus large de ces méthodes.

Cependant, le nombre de scénarios et de noeuds peut augmenter de manière exponentielle avec le nombre de pas de temps. Les méthodes de réduction de scénarios (Dupačová et al., 2003)-(Heitsch and Römisch, 2003) ou de décomposition (Rockafellar and Wets, 1991a) peuvent permettre de réduire la complexité de la résolution.

1.3.2. Contrôle optimal stochastique

Au contraire des méthodes de programmation stochastique, les méthodes de contrôle optimal stochastique se formulent en fixant un certain horizon (possiblement infini) ainsi qu'un ensemble de contrôles, d'états et d'incertitudes. Si les incertitudes sont indépendantes, pas de temps par pas de temps, un résultat connu permet de résoudre le problème par programmation dynamique (Bellman, 1957) en calculant un ensemble de fonctions valeurs (les fonctions de Bellman) de manière récursive. Une fois ces fonctions valeurs obtenues, nous pouvons construire une politique de contrôle optimal pour chaque pas de temps. Nous faisons référence à (Bertsekas, 2012)-(Puterman, 1994) pour une description des équations de la programmation dynamique dans un contexte stochastique, en temps discret. Ici, nous nous plaçons avant tout dans le formalisme introduit dans (Carpentier et al., 2015) pour décrire des problèmes de contrôle optimal stochastique.

Nous détaillons dans la suite les principaux algorithmes pouvant être utilisés pour résoudre les équations de Bellman.

Stochastic Dynamic Programming (SDP). SDP est un algorithme classique pour calculer les fonctions de Bellman de manière récursive. L'espace des états est discrétisé de façon à obtenir un nombre fini de points à explorer, ce qui autorise le calcul d'une approximation des fonctions de Bellman par recherche exhaustive. Cette procédure est équivalente à modéliser le problème originel comme un processus décisionnel de Markov (Puterman, 1994).

Stochastic Dual Dynamic Programming (SDDP). SDDP est un algorithme qui utilise des résultats classiques en analyse convexe permettant d'approximer toute fonction convexe comme un supremum de fonctions d'appui affines.

SDDP calcule itérativement une approximation des fonctions de Bellman comme un supremum de fonctions affines. À chaque itération, l'algorithme calcule de manière *forward* un ensemble de trajectoires où il serait judicieux de raffiner l'approximation courante. Ensuite, SDDP améliore l'approximation des fonctions de Bellman de manière *backward*, en calculant un sous-gradient en chacun des points des trajectoires précédemment calculées pendant la phase *forward*. Ce sous-gradient permet ensuite de construire une nouvelle borne inférieure affine.

Nous nous référons à (Van Slyke and Wets, 1969) pour l'idée originelle résidant derrière SDDP, et à (Pereira and Pinto, 1991) pour une première description de l'algorithme SDDP en tant que tel. Une preuve de convergence dans le cas linéaire existe (Philpott and Guan, 2008), preuve qui a ensuite été étendue dans le cas convexe (Girardeau et al., 2014). Nous faisons référence à (Shapiro, 2011) pour une description récente de l'algorithme SDDP. Des travaux ont récemment étendu SDDP à des problèmes avec variables entières (Zou et al., 2017).

Autres méthodes de résolution.

Approximate Dynamic Programming (ADP). Tandis que SDDP approxime les fonctions valeurs comme un supremum de fonctions affines, ADP approxime les fonctions valeurs par d'autres classses de fonctions, appropriées au problème étudié.

Nous faisons référence à (Bertsekas and Tsitsiklis, 1996)-(Powell, 2007) pour une description de l'algorithme ADP.

Reinforcement learning (RL). Les précédents algorithmes (de SDP à ADP) ont besoin d'une fonction dynamique pour décrire l'évolution temporelle des variables d'état. Au contraire, les méthodes d'apprentissage par renforcement fonctionnent sans dynamique connu. Elles considèrent uniquement un ensemble d'échantillons qui associent un ensemble de paires d'états et de décisions à des gains, ces échantillons pouvant être observés en ligne ou hors ligne. L'apprentissage par renforcement approxime alors des fonctions valeurs qui associent chaque paire état-décision à un coût donné.

Nous faisons référence à (Sutton and Barto, 1998) pour une revue des méthodes d'apprentissage par renforcement.

Méthodes particulaires. Tandis que les algorithmes précédents requièrent le calcul (exact ou approximé) de fonctions valeurs, d'autres algorithmes utilisent quant à eux des principes variationnels — comme le principe du maximum de Pontryagin (Pontryagin, 1962) — pour calculer directement les décisions optimales satisfaisant les conditions d'optimalité au premier ordre.

Nous faisons référence à (Dallagi, 2007) pour une application du principe du maximum de Pontryagin aux problèmes d'optimisation stochastique à plusieurs pas de temps, ainsi qu'à (Carpentier et al., 2015) pour une présentation plus large des outils théoriques sur lesquels s'appuient les méthodes particulaires.

1.3.3. Décomposition des problèmes d'optimisation stochastique à plusieurs pas de temps.

La performance des algorithmes reposant sur la résolution des équations de Bellman est directement liée au nombre de dimensions de l'état sous-jacent. Ces algorithmes sont confrontés à la malédiction de la dimension : plus la dimension de cet état est grand, plus la résolution des équations de Bellman devient (exponentiellement) difficile.

Pour pallier ce problème, il est possible, dans certains cas, de décomposer le problème originel en sous-problèmes de plus petite taille. L'objectif devient alors de coordonner les sous-problèmes entre eux via une variable de coordination (par exemple un prix ou une resource) jusqu'à retrouver l'optimum global du problème originel. Nous faisons référence à Carpentier and Cohen (2017) pour une description générique des méthodes de décomposition-coordination.

La décomposition des problèmes d'optimisation stochastique à plusieurs pas de temps est un champ de recherche encore actif. La principale difficulté réside dans le fait que, dans ce cas, les variables de coordination deviennent des processus stochastiques (possédant éventuellement une certaine dynamique), compliquant ainsi la résolution des sous-problèmes par programmation dynamique. L'idée devient alors d'approximer les processus de coordination en utilisant un autre processus dont la dynamique est connue (Barty et al., 2010a). Cette idée a été appliquée avec succès à la résolution de problèmes d'optimisation stochastique à plusieurs pas de temps de grande taille, par exemple des problèmes d'affectation de la production (Girardeau, 2010) ou des problèmes de gestion de vallées hydrauliques (Alais, 2013). Des développements théoriques concernant les méthodes de décomposition stochastique peuvent être trouvés dans Leclère (2014).

1.4. Contributions

Nous détaillons maintenant les principales contributions de ce manuscrit. La thèse est divisée en trois parties.

Décomposition temporelle pour l'optimisation de la gestion de microgrid domestique. Dans la première partie, nous appliquons les méthodes d'optimisation stochastique à plusieurs pas de temps à la gestion de microgrids de petite taille. Nous cherchons à obtenir les stratégies de gestion optimale afin de prendre des décisions en ligne au sein du système de gestion de l'énergie.

- Dans le Chapitre 2, nous introduisons un cadre générique pour classifier différentes stratégies de gestion en ligne. Nous distinguons deux classes de stratégies en ligne. Les stratégies reposant sur une fonction valeur minimisent un problème d'optimisation avec un horizon d'un pas de temps afin d'obtenir une décision en ligne. Cette méthode voit le futur uniquement à travers la fonction valeur donnée en entrée. Au contraire, les méthodes *lookahead* minimisent directement un problème d'optimisation à plusieurs pas de temps. Dans ce premier chapitre, nous définissons par ailleurs formellement la variable d'état comme une réduction des décisions et des incertitudes précédentes. Enfin, nous proposons une méthode d'évaluation générique pour comparer différentes stratégies entre elles.

- Dans le Chapitre 3, nous présentons les modèles physiques utilisés pour construire les problèmes d'optimisation dans les chapitres 4 et 5. Ces modèles physiques sont un compromis entre précision physique et solvabilité du problème d'optimisation sous-jacent. Nous distinguons les modèles purement physiques (pour les stockages et les unités de production) des modèles statistiques utilisés pour représenter les aléas futurs.

- Dans le Chapitre 4, nous considérons une maison équipée avec une centrale de micro-cogénération et une batterie. Nous modélisons le problème de gestion de l'énergie comme un problème d'optimisation stochastique à plusieurs pas de temps, et nous comparons trois stratégies en ligne — une heuristique, une stratégie reposant sur Model Predictive Control et une stratégie reposant sur la programmation dynamique.

- Dans le Chapitre 5, nous considérons un nouveau problème de gestion d'une microgrid résidentielle, cette fois-ci équipée avec des panneaux solaires et une batterie. Contrairement au Chapitre 4, nous nous intéressons ici avant tout à la modélisation des aléas dans la conception des stratégies en ligne.

Décomposition temporelle et spatiale de problèmes d'optimisation de grande taille. Tandis que la première partie traite surtout de problèmes de petite taille, la seconde partie cherche à étendre les algorithmes présentés durant cette première partie à des problèmes stochastiques de grande taille.

- Dans le Chapitre 6, nous présentons deux méthodes de décomposition utilisant respectivement des prix ou des resources comme variables de coordination. Ces méthodes permettent d'obtenir des bornes inférieures et supérieures pour le problème originel. Sous des hypothèses adéquates, nous pouvons résoudre les sous-problèmes (obtenus par décomposition du problème global) par programmation dynamique. Nous pouvons prouver de surcroît que les sommes des fonctions de Bellman locales permettent de borner inférieurement et supérieurement les fonctions de Bellman globales, et ce à chaque pas de temps.

- Dans le Chapitre 7, nous appliquons les méthodes de décomposition stochastique présentées dans le Chapitre 6 à la gestion de microgrids de grande taille, pouvant rassembler jusqu'à 48 bâtiments. Nous illustrons numériquement l'intérêt des algorithmes de décomposition en comparant les algorithmes de décomposition par les prix et par les ressources avec une version de SDDP qui résout globalement les équations de Bellman du problème initial.

- Dans le Chapitre 8, nous étudions un autre problème portant cette fois-ci sur la gestion de vallées hydrauliques de grande taille. Nous présentons une nouvelle étude numérique, comparant cette fois-ci la décomposition par les prix avec l'algorithme SDDP.

Contributions à l'algorithme Stochastic Dual Dynamic Programming. Enfin, nous abordons dans la dernière partie des considérations portant essentiellement sur l'algorithme *Stochastic Dual Dynamic Programming* (SDDP).

- Dans le Chapitre 9, nous présentons une nouvelle méthode pour calculer une borne supérieure déterministe pour des problèmes d'optimisation stochastique à plusieurs pas de temps, en exploitant la dualité au sens de Fenchel. Cette méthode permet d'obtenir un critère d'arrêt déterministe pour SDDP. Nous illustrons ces considérations théoriques avec des résultats numériques montrant la pertinence de la méthode proposée.

- Dans le Chapitre 10, nous incluons un complément pour le Chapitre 9 en présentant une preuve de convergence pour la nouvelle version de SDDP présentée dans le chapitre précédent.

Chapter 2.

Introduction

Contents

By applying mathematical optimization methods to energy systems, this thesis is at a crossroad between two domains. We first give in Section 2.1 some elements to situate the context of the work presented in this document, and then present overviews of energy management systems in Section 2.2 and of stochastic optimization methods in Section 2.3. Then, we will detail in Section 2.4 the structure of this dissertation.

2.1. Context

Once Tesla's AC power overcame Edison's DC power at the end of the 19th century, the electricity grid moved from an assembly of decentralized producing units to an interconnection of large centralized units. The transmission grids have pursued their centralization further during the 20th century, as transmission networks extended.

However, with the recent democratization of renewable energies, the feud between centralized and decentralized grids is regaining interest. Whereas some argue that large regional networks would allow to dispatch more efficiently the renewable energies to final consumers — as back-up costs would be mutualized across a large panel of consumers — others reply that it would be more effective to consume the energy directly where it is produced. In fact, decentralized distribution grids would allow to decrease the energy losses — occurring when electricity is transported through large distance — and would allow to build a more resilient grid being less sensitive to transmission outages.

Here, we will consider small networks of electricity users having their own local sources of supply. Such small networks are called *microgrids*, to distinguish them from larger *smart grids*. Urban microgrids gain interest recently since a change in the French legislation that allows local producers to share energy between themselves, as depicted in Figure 2.1. Urban microgrids remain complex systems confronted with external uncertainties (local demands, renewable energy production) and gathering potentially large amount of heterogeneous stocks. This work is part of a

collaboration between Efficacity — a French institute for the energy transition — and CERMICS — a mathematical laboratory — to study the optimal management of urban microgrids.

Our main contributions lie in the management of uncertainties to improve the resilience of microgrid systems. As explained in §2.2, we focus on the optimal management of urban microgrids from a mathematical point of view. We will first tackle small scale microgrids, with a single building, and extend our study to large-scale microgrids gathering up to 48 buildings. We will adapt some existing stochastic optimization algorithms, presented in §2.3, to control effectively urban microgrids. In parallel, we will extend the decomposition algorithms to other energy systems, such as large scale dams-valleys.

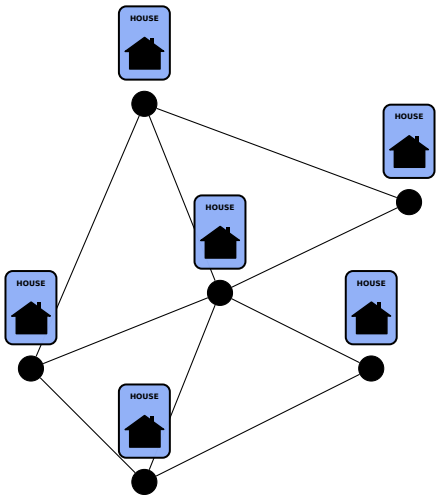


Figure 2.1.: Collective self-consumption at district level

2.2. Management of microgrids

Lots of works have been devoted recently to the design of energy management systems in microgrids. We detail hereafter the different challenges that arise in energy management systems, and then point out a survey concerning the optimization of the control of energy management systems.

2.2.1. Energy management systems

We focus here on the energy management system (EMS) of local microgrids. At a given moment, the energy management system manages the different energy flows in the microgrid, so as to minimize the operational costs. Then, the energy flows returned by the energy management system are used as a reference by a primary controller managing the power flows inside the microgrid.

The main challenges that fall out in the optimization of EMS are the following.

- The EMS cannot know the exact realization of the future energy demands and energy production, thus increasing the difficulty to ensure the adequacy between production and demand.
- The more uncertainties, the more likely failures become. The design of EMS must be *robust* to extreme events.
- Microgrids can gather dozens of heterogeneous stocks, giving large-scale optimization problems. Classic optimization algorithms may become irrelevant to solve problems of this size.

- Energy systems are complex. Sometimes, the energy production cannot be modulated, or devices might have complex operational constraints. These constraints may introduce uncommon optimization approaches to deal with integer or binary variables, thus increasing the complexity of the resolution.

2.2.2. Optimization of energy management systems.

The design of EMS for microgrids has raised much interest in recent years. In Olivares et al. (2014), the authors give a survey of application of optimization methods to the design of EMS. Even if stochastic methods are gaining interest, the reference method remains the well-known Model Predictive Control (MPC) (Garcia et al., 1989) and deterministic methods to control EMS. The main application of microgrids remains for insulated and self-sufficient systems. We refer to Olivares et al. (2011) for an application of MPC in the EMS of an insulated microgrid. In Heymann et al. (2015) the authors study an insulated microgrid in Chile and compute optimal solutions for the EMS by solving the deterministic Hamilton-Jacobi-Bellman equations.

Apart of insulated microgrids, the control of residential microgrids is also a vivid research area. We refer to Oldewurtel et al. (2012) for an application of MPC to residential buildings and to Lamoudi (2012) for an application of MPC to office buildings. In Parisio et al. (2015), the authors extend MPC to control a panel of residential buildings. In a similar manner, Pflaum et al. (2014) applied decomposition methods to decompose the resolution of MPC.

Tackling uncertainties with stochastic optimization. At local scale, electrical demands and productions are highly variable, especially as microgrids are expected to absorb renewable energies. The recent development in probabilistic forecasts (Morales et al., 2013) leads to pay a growing attention to stochastic optimization approaches. In Oldewurtel (2011), the author presents a Stochastic MPC scheme relying on stochastic programming to handle different forecast scenarios. In Olivares et al. (2015), the authors use stochastic unit-commitment to compute the setpoints and then use MPC as a second level controller to refine online the dispatch of the unit-commitment algorithm. Other approaches use robust optimization algorithms to handle uncertainties by considering worst-case scenarios. We refer to Paridari et al. (2016) for a given use case in residential buildings, and to Wytock et al. (2017) for an application of scenario-based robust MPC to the management of smart-home.

With the growing interest in machine learning, reinforcement learning algorithms are also investigated for the control of EMS. We refer to Ernst et al. (2009) for a comparison of MPC with reinforcement learning methods.

Aside from microgrid management, stochastic optimization has found numerous applications in energy systems (De Lara et al., 2014). Historically, stochastic optimization has been widely applied to dams-valley management (Pereira and Pinto, 1991). Other applications have arisen recently, such as integration of wind energy and storage (Haessig, 2014) or insulated microgrids management (Heymann et al., 2016). Apart from microgrid management, other applications arise in the resolution of unit-commitment problems (Carpentier et al., 1996) or hydro-thermal scheduling (Philpott and De Matos, 2012). We refer to Wallace and Fleten (2003) for a review of applications of stochastic optimization in energy systems.

2.3. Background on stochastic optimization

Stochastic optimization methods aim at minimizing a criterion over random variables. Such methods are at the intersection between optimization and probability theory.

There exists two manners to combine decisions with uncertainties in stochastic optimization. In open-loop stochastic optimization, the decision maker computes his decision once and for all, by considering a single stage and compute a solution without recourse w.r.t. the future realization of

uncertainties. On the contrary, in closed-loop stochastic optimization the decision-maker is able to adapt his decisions to previous realization of uncertainties.

In energy management systems, most problems formulate as multistage stochastic optimization problems (Carpentier et al., 2015) where uncertainties arise at different time steps. We focus hereafter on multistage stochastic optimization resolution methods.

2.3.1. Stochastic Programming

Stochastic programming encodes the realization of the future uncertainties with a finite scenario tree. Then, it aims to find the optimal decisions attached to every node in the tree, so as to minimize the average cost. We refer to Shapiro et al. (2009) and Kall et al. (1994) for a broader introduction.

However, the number of scenarios and nodes grows exponentially with the number of time steps. It pays to use scenario reduction algorithms (Dupačová et al., 2003)-(Heitsch and Römisch, 2003) or decomposition algorithms (Rockafellar and Wets, 1991a) to reduce the numerical burden.

2.3.2. Stochastic Optimal Control

On the contrary of stochastic programming, Stochastic Optimal Control (SOC) is a class of problems formulated with time, controls, states and uncertainties. Provided that the uncertainties are stagewise independent, the problem satisfies the recursive Dynamic Programming equations (Bellman, 1957) that allow to compute a set of Bellman functions in a backward manner. Once these Bellman functions are obtained, we are able to compute the optimal decisions at each time step. We refer to Bertsekas (2012)-Puterman (1994) for a description of the theory lying behind the resolution of Dynamic Programming equations in discrete time. We follow all along this thesis the formalism of Carpentier et al. (2015) to describe multistage stochastic optimization problems.

In the sequel, we detail the main algorithms devoted to the resolution of the Bellman recursive equations.

Stochastic Dynamic Programming (SDP). SDP is a classical method to compute the Bellman functions backward in time. In the algorithmic version, one discretizes the state space to obtain a finite number of features and then computes the (approximate) Bellman equations by exhaustive search. This is equivalent to model the original problem as a Markov decision process (MDP) (Puterman, 1994) before solving it exhaustively.

Stochastic Dual Dynamic Programming (SDDP). SDDP is an algorithm that use a classical results in convex analysis stating that any convex function can be approximated below by a supremum of affine functions. SDDP applies to convex multistage problems.

SDDP computes iteratively an approximation of the Bellman functions as a supremum of affine functions. At each iteration, it computes forward a set of trajectories where it would be wise to refine the approximated value functions. Then, SDDP refines its approximated Bellman value functions during a backward pass, by computing the subgradients of each incumbent state.

We refer to Van Slyke and Wets (1969) for the original idea lying behind SDDP, and to Pereira and Pinto (1991) for a first description of the SDDP algorithm. A proof of convergence in the linear case was given in Philpott and Guan (2008) and extended in the convex case in Girardeau et al. (2014). We refer to Shapiro (2011) for an up-to-date description of the SDDP algorithm. Recent works extend the SDDP algorithm to the integer case (Zou et al., 2017).

Other resolution methods.

Approximate Dynamic Programming (ADP). Whereas SDDP approximates the value functions as supremum of affine functions, ADP approximates the value functions with generic functions having suitable properties for the problem under study.

We refer to (Bertsekas and Tsitsiklis, 1996)-(Powell, 2007) for a comprehensive description of Approximate Dynamic Programming algorithms.

Reinforcement learning (RL). All previous algorithms (from SDP to ADP) rely on a dynamics to describe the evolution of the state variables. On the contrary, reinforcement learning methods do not require any dynamics. They only handle a set of samples that associate decisions to rewards, these samples being observed offline or online. Then, the RL approximates a value function that associates each peer of state and decision to a given cost.

We refer to Sutton and Barto (1998) for an exhaustive review of reinforcement learning methods.

Particle methods. Whereas the previous algorithms rely on the computation of some cost-to-go, some other algorithms use variational principles — as the Pontryagin maximum principle (Pontryagin, 1962) — to compute the optimal decisions satisfying the first order optimality conditions, without need of any cost-to-go.

We refer to Dallagi (2007) for an application of the Pontryagin maximum principle to multistage stochastic optimization methods, and to Carpentier et al. (2015) for a broader description of the theory lying behind the so-called particle methods.

2.3.3. Decomposition of multistage stochastic optimization problems.

The main drawback of the algorithms relying on the resolution of the Bellman recursive equations is that they are confronted by the curse of dimensionality: the larger the dimension of the state, the more cumbersome the resolution becomes.

To overcome this issue, there are cases where the original problem can be decomposed in smaller subproblems with a more tractable state size. Then, the algorithm coordinates the subproblems via a given coordination variable (e.g. a price or a ressource) till a global optimum is found. We refer to Carpentier and Cohen (2017) for a generic description of decomposition-coordination methods.

The decomposition of multistage stochastic optimization problem remains under study. The main difficulty lies in the fact that the coordination variable are stochastic processes possibly exhibiting a given dynamics thus complicating the resolution of the local subproblems by Dynamic Programming. The idea is then to approximate the coordination variable with another process, whose dynamics is known (Barty et al., 2010a). This idea was successfully applied to the resolution of multistage stochastic optimization problem, starting from a unit-commitment problem (Girardeau, 2010) to a dams-valley management problem (Alais, 2013). Theoretical insights concerning stochastic decomposition methods are given in Leclère (2014).

2.4. Contributions

We now emphasize the different contributions presented in this manuscript. The manuscript comprises three parts.

Time decomposition in optimization and management of home microgrids. In the first part, we focus on the application of stochastic optimization algorithms to the management of small-scale microgrids (e.g. residential microgrids). We aim at finding optimal policies to take online decisions in energy management systems.

- In Chapter 3, we introduce a common framework to frame online policies relying on stochastic optimization methods. We distinguish two classes of online policies. The *cost-to-go policies*

minimize a one-step optimization problem to yield an online decision, the future being hidden inside a given cost-to-go. The *lookahead policies* solve instead a multistage optimization problem. We define the state as a reduction of a history gathering past decisions and uncertainties at any moment. Eventually, we propose a generic assessment method to compare different stochastic optimization policies.

- In Chapter 4, we present the physical models we use to build optimization problems. Such models are a trade-off between the physical accuracy and the tractability of the resolution by optimization algorithms. We distinguish the physical models (for stocks and producing units) from the statistical models used to represent the future uncertainties and from the economic model to represent costs.

- In Chapter 5, we consider a residential building equipped with a micro-combined Heat-and-Power generator and a battery. We frame the energy management system problem as a multistage stochastic optimization problem and compare three online policies (a heuristic, MPC and SDDP policies).

- In Chapter 6, we consider another residential microgrid problem, this time being equipped with solar panels and a battery. Comparing to Chapter 5, we improve the modeling of uncertainties in the design of the online policies. Then, we compare SDDP with MPC.

Mixing time and spatial decomposition in large-scale optimization problems. Whereas the first part deals with small-scale systems, the second part deals with large-scale stochastic systems.

- In Chapter 7, we present price and resource decomposition methods, to compute decomposed upper and lower bounds for the original problem. Under some assumptions, we are able to solve the decomposed subproblems by Dynamic Programming. We then prove that the sums of local Bellman value functions give lower and upper bounds for the global Bellman value functions.

- In Chapter 8, we apply the decomposition methods introduced in Chapter 7 to the management of a large-scale microgrid, gathering up to 48 interconnected buildings. We emphasize the relevance of the decomposition algorithms with numerical results comparing price and resource decomposition with a version of SDDP that tackles the resolution of Dynamic Programming equations globally.

- In Chapter 9, we study large scale dams-valley problems and present another numerical comparison of price decomposition methods with SDDP.

Contributions to Stochastic Dual Dynamic Programming. Eventually, the third part deals with more theoretical considerations concerning the Stochastic Dual Dynamic Programming algorithm.

- In Chapter 10, we introduce a novel method to compute a deterministic upper-bound of a multistage stochastic optimization problem, by exploiting Fenchel duality. That allows to design a deterministic stopping criterion for SDDP. We illustrate these theoretical results with numerical results showing the effectiveness of the method.

- In Chapter 11, we give a complement for Chapter 10 with a proof of convergence of the so-called abstract SDDP algorithm.

Part I.

Time decomposition in optimization and management of home microgrids

Abstract. In this first part, we focus on the control of small scale units, such as encountered in residential microgrids. The main challenge of the energy management system is to ensure, at least cost, that supply matches demand for all time while handling the inherent uncertainties of such systems (demand, renewable energy production). Hereafter, we use optimization algorithms to design *online policies* that will handle the available information to take a decision at a given moment. We will compare different methods to design policies, and provide benchmarks on two use-cases.
We first describe in Chapter 3 a generic framework to design online policies, and introduce a classification of different existing methods to frame online policies. Then, we detail in Chapter 4 the physical modelling of the devices we look at in residential microgrids, as well as a statistical modelling of the different uncertainties we usually encounter (e.g. the energy demands and the renewable productions). Then, we present two case studies. In Chapter 5 we handle a residential microgrid equipped with a micro-Combined Head and Power generator (μCHP) — producing electricity and heat — and a battery. We compare different online policies and show the effectiveness of optimization algorithms to compute optimal policies by comparing the performance of each policy over one year. In Chapter 6, we examine a new case study consisting of another residential microgrid, this time equipped with solar panels and a battery. We put emphasis on the online information used by the online policies and give a benchmark for three representative days in a year.

Chapter 3.

A template to design online policies for multistage stochastic optimization problems

Contents

3.1. Introduction

In multistage stochastic optimization problems, the decision maker makes, at every stage, a decision that depends at most upon past uncertainties. Once a decision taken, he has to wait till the next realization of uncertainty to take a new decision.

Such problems naturally arise in the management of energy systems (see De Lara et al. (2014)) where systems are often confronted to external uncertainties (e.g. renewable productions or electrical demands). Mathematically, they formulate as multistage stochastic optimization problems whose solutions are indexed both by stages and by uncertainties. However, the resolution of such problems proves to be difficult. On the one hand, stochastic programming methods (see Shapiro et al. (2009)) handles the resolution by writing a scenario tree corresponding to realizations of all uncertainties, hence potentially (very) large. On the other hand, stochastic control methods rely on the Dynamic Programming principle and look for solutions as feedback on previous history, but are naturally confronted by the *curse of dimensionality*. We refer to Bellman (1957)-Bertsekas (2012)-Puterman (1994) for an overview of stochastic dynamic programming methods.

Because exact solutions are out of reach, there is ongoing interest in the design of *approximate* resolution algorithms designed to tackle the shortfalls of the different *exact* resolution methods. In (Bertsekas, 2012, Chapter 6), a whole chapter is dedicated to *approximate dynamic programming*

methods. This work was detailed in Bertsekas (2005b) where the emphasis is put on the well-known Model Predictive Control. More recently, Powell (2014) proposes a framework to categorize different resolution algorithms in four different classes, with a stress being put on approximate dynamic programming and reinforcement learning methods. The common point between these two works is that the authors do not solve the original multistage problem all in once but choose instead to take decisions with *online policies* that are refined as time goes on and uncertainties accumulate.

We follow the approach introduced in Carpentier et al. (2018a) and formulate multistage stochastic programming over a *history* space in Section 3.2. We introduce two classes of history feedback policies: the so-called cost-to-go policies and lookahead policies. In Section 3.3, we focus on lookahead policies and frame three existing algorithms in this class. In Section 3.4, we describe the cost-to-go methods and detail the offline computation of cost-to-go. Finally, we present in Section 3.5 a method to compare different online policies together, in a fair manner.

3.2. Multistage stochastic optimization problems

We reconsider in §3.2.1 the framework introduced in Carpentier et al. (2018a) to frame multistage stochastic optimization problems with history. This framework allows to write explicitly the solution of multistage stochastic problems via Bellman recursive equations. Then, we define formally a state as a reduction of the history in §3.2.2, and describe the compatibility assumptions existing between the history and the state process. Eventually, we introduce in §3.2.3 two schemes to describe online policies: the cost-to-go policies and the lookahead policies.

3.2.1. Stochastic dynamic programming with history feedback policies

Consider the time span $\{0, 1, 2 \dots, T-1, T\}$, with *horizon* $T \in \mathbb{N}^*$. At the end of the time interval $[t-1, t[$, an uncertainty variable w_t is produced. Then, at the beginning of the time interval $[t, t+1[$, a decision-maker takes a decision u_t. The interplay between uncertainty and decision is as follows

$$w_0 \rightsquigarrow u_0 \rightsquigarrow w_1 \rightsquigarrow u_1 \rightsquigarrow \quad \dots \quad \rightsquigarrow w_{T-1} \rightsquigarrow u_{T-1} \rightsquigarrow w_T \ .$$

We present a mathematical formalism to handle such type of problems.

3.2.1.1. Histories, feedback and flows

We first define the basic and the composite spaces that we will need to formulate multistage stochastic optimization problems. Then, we introduce a class of solutions called history feedback and the associated notion of flow.

Histories and history spaces. For each time $t = 0, 1, 2 \dots, T-1$, the decision u_t takes its values in a measurable set $\mathbb{U}_t$ equipped with a σ-field $\mathcal{U}_t$. For each time $t = 0, 1, 2 \dots, T$, the uncertainty w_t takes its values in a measurable set $\mathbb{W}_t$ equipped with a σ-field $\mathcal{W}_t$.

For $t = 0, 1, 2 \dots, T$, we define the *history space* $\mathbb{H}_t$ equipped with the *history field* $\mathcal{H}_t$ by

$$\mathbb{H}_t = \mathbb{W}_0 \times \prod_{s=0}^{t-1} (\mathbb{U}_s \times \mathbb{W}_{s+1}) \text{ and } \mathcal{H}_t = \mathcal{W}_0 \otimes \bigotimes_{s=0}^{t-1} (\mathcal{U}_s \otimes \mathcal{W}_{s+1}) \ , \ \ t = 0, 1, 2 \dots, T \ , \tag{3.1}$$

with the particular case $\mathbb{H}_0 = \mathbb{W}_0$, $\mathcal{H}_0 = \mathcal{W}_0$. The notation $\bigotimes$ denotes the usual product between

σ-fields. A generic element $h_t \in \mathbb{H}_t$ is called a *history*:

$$h_t = (w_0, (u_s, w_{s+1})_{s=0,\dots,t-1}) = (w_0, u_0, w_1, u_1, w_2, \dots, u_{t-2}, w_{t-1}, u_{t-1}, w_t) \in \mathbb{H}_t \ . \tag{3.2a}$$

We introduce the notations

$$\mathbb{W}_{r:t} = \prod_{s=r}^{t} \mathbb{W}_s \ , \ \ 0 \le r \le t \le T \ , \tag{3.2b}$$

$$\mathbb{U}_{r:t} = \prod_{s=r}^{t} \mathbb{U}_s \ , \ \ 0 \le r \le t \le T-1 \ , \tag{3.2c}$$

$$\mathbb{H}_{r:t} = \prod_{s=r-1}^{t-1} (\mathbb{U}_s \times \mathbb{W}_{s+1}) = \mathbb{U}_{r-1} \times \mathbb{W}_r \times \cdots \times \mathbb{U}_{t-1} \times \mathbb{W}_t \ , \ \ 1 \le r \le t \le T \ . \tag{3.2d}$$

Let $0 \le r \le s \le t \le T$. From a history $h_t \in \mathbb{H}_t$, we extract the $(r\,{:}\,s)$-*history uncertainty part*

$$[h_t]^{\mathbb{W}}_{r:s} = (w_r, \dots, w_s) = w_{r:s} \in \mathbb{W}_{r:s} \ , \ \ 0 \le r \le s \le t \ , \tag{3.2e}$$

the $(r\,{:}\,s)$-*history control part* (notice that the indices are special)

$$[h_t]^{\mathbb{U}}_{r:s} = (u_{r-1}, \dots, u_{s-1}) = u_{r-1:s-1} \in \mathbb{U}_{r-1:s-1} \ , \ \ 1 \le r \le s \le t \ , \tag{3.2f}$$

and the $(r\,{:}\,s)$-*history*

$$[h_t]_{r:s} = (u_{r-1}, w_r, \dots, u_{s-1}, w_s) = h_{r:s} \in \mathbb{H}_{r:s} \ , \ \ 1 \le r \le s \le t \ , \tag{3.2g}$$

so that we obtain, for $0 \le r+1 \le s \le t$,

$$h_t = (\underbrace{w_0, u_0, w_1, \dots, u_{r-1}, w_r}_{h_r} \underbrace{u_r, w_{r+1}, \dots, u_{t-2}, w_{t-1}, u_{t-1}, w_t}_{h_{r+1:t}}) = (h_r, h_{r+1:t}) \ . \tag{3.2h}$$

Feedbacks and flows. Let r and t be given such that $0 \le r \le t \le T$.

History feedback policies. When $0 \le r \le t \le T-1$, we define a $(r\,{:}\,t)$-*history feedback policy* as a sequence $\{\gamma_s\}_{s=r,\dots,t}$ of measurable mappings

$$\gamma_s : (\mathbb{H}_s, \mathcal{H}_s) \to (\mathbb{U}_s, \mathcal{U}_s) \ . \tag{3.3}$$

We call $\Gamma_{r:t}$ the set of $(r\,{:}\,t)$-history feedback policies.

Flows. When $0 \le r < t \le T$, for a $(r\,{:}\,t-1)$-history feedback $\gamma = \{\gamma_s\}_{s=r,\dots,t-1} \in \Gamma_{r:t-1}$, we define the *flow* $\Phi^{\gamma}_{r:t}$ by

$$\Phi^{\gamma}_{r:t} : \mathbb{H}_r \times \mathbb{W}_{r+1:t} \to \mathbb{H}_t \tag{3.4a}$$

$$(h_r, w_{r+1:t}) \mapsto (h_r, \gamma_r(h_r), w_{r+1}, \gamma_{r+1}(h_r, \gamma_r(h_r), w_{r+1}), w_{r+2}, \cdots, u_{t-1}, w_t) \ , \tag{3.4b}$$

that is,

$$\Phi^{\gamma}_{r:t}(h_r, w_{r+1:t}) = (h_r, u_r, w_{r+1}, u_{r+1}, w_{r+2}, \dots, u_{t-1}, w_t) \ , \tag{3.4c}$$

with $h_s = (h_r, u_r, w_{r+1}, \dots, u_{s-1}, w_s)$, $r < s \leq t$ and $u_s = \gamma_s(h_s)$, $r < s \leq t-1$. When $0 \leq r = t \leq T$, we put

$$\Phi^{\gamma}_{r:r} : \mathbb{H}_r \to \mathbb{H}_r \ , \ \ h_r \mapsto h_r \ . \tag{3.4d}$$

With this convention, the expression $\Phi^{\gamma}_{r:t}$ makes sense when $0 \leq r \leq t \leq T$ for a $(r\,{:}\,t-1)$-history feedback $\gamma = \{\gamma_s\}_{s=r,\dots,t-1} \in \Gamma_{r:t-1}$ (when $r = t$, no $(r\,{:}\,r-1)$-history feedback exists, but none is needed).

The mapping $\Phi^{\gamma}_{r:t}$ gives the history at time t as a function of the initial history h_r at time r and of the history feedback policies $\{\gamma_s\}_{s=r,\dots,t-1} \in \Gamma_{r:t-1}$. An immediate consequence of this definition are the two following *flow properties*:

$$\Phi^{\gamma}_{r:t+1}(h_r, w_{r+1:t+1}) = \Big(\Phi^{\gamma}_{r:t}(h_r, w_{r+1:t}), \gamma_t\big(\Phi^{\gamma}_{r:t}(h_r, w_{r+1:t})\big), w_{t+1}\Big) \ , \ \ 0 \leq r \leq t \leq T-1 \ , \tag{3.5a}$$

$$\Phi^{\gamma}_{r:t}(h_r, w_{r+1:t}) = \Phi^{\gamma}_{r+1:t}\big((h_r, \gamma_r(h_r), w_{r+1}), w_{r+2:t}\big) \ , \ \ 0 \leq r < t \leq T \ . \tag{3.5b}$$

3.2.1.2. Optimization with stochastic kernels

In the sequel, we suppose we are given a sequence of stochastic kernels. Then, given a history feedback and a sequence of stochastic kernels from partial histories to uncertainties, we will build a new sequence of stochastic kernels, but from partial histories to sequences of uncertainties. With this construction, we are able to introduce a family of optimization problems with stochastic kernels. Then, we show how such problems can be solved by stochastic dynamic programming.

In what follows, we say that a function is *numerical* if it takes its values in $[-\infty, +\infty]$ (also called *extended* or *extended real-valued* function) (Loève, 1977).

Stochastic Kernels.

Definition of stochastic kernels. Let $(\mathbb{X}, \mathcal{X})$ and $(\mathbb{Y}, \mathcal{Y})$ be two measurable spaces. A *stochastic kernel* from $(\mathbb{X}, \mathcal{X})$ to $(\mathbb{Y}, \mathcal{Y})$ is a mapping $\rho : \mathbb{X} \times \mathcal{Y} \to [0, 1]$ such that

- for any $Y \in \mathcal{Y}$, $\rho(\cdot, Y)$ is $\mathcal{X}$-measurable;
- for any $x \in \mathbb{X}$, $\rho(x, \cdot)$ is a probability measure on $\mathcal{Y}$.

By a slight abuse of notation, a stochastic kernel (on $\mathbb{Y}$ knowing $\mathbb{X}$) is also denoted as a mapping $\rho : \mathbb{X} \to \Delta(\mathbb{Y})$ from the measurable space $(\mathbb{X}, \mathcal{X})$ towards the space $\Delta(\mathbb{Y})$ of probability measures over $\mathcal{Y}$, with the property that the function $x \in \mathbb{X} \mapsto \int_Y \rho(x, \mathrm{d}y)$ is measurable for any $Y \in \mathcal{Y}$.

Building new stochastic kernels from history feedback policies and stochastic kernels.

Definition 3.2.1. *Let r and t be given such that $0 \leq r \leq t \leq T$.*

- *For $0 \leq r < t \leq T$, let be*
  1. *a $(r\,{:}\,t-1)$-history feedback $\gamma = \{\gamma_s\}_{s=r,\dots,t-1} \in \Gamma_{r:t-1}$,*
  2. *a family $\{\rho_{s-1:s}\}_{r+1 \leq s \leq t}$ of stochastic kernels*

$$\rho_{s-1:s} : \mathbb{H}_{s-1} \times \mathcal{W}_s \to [0, 1] \ , \ \ s = r+1, \dots, t \ , \tag{3.6}$$

that we note also

$$\rho_{s-1:s} : \mathbb{H}_{s-1} \to \Delta(\mathbb{W}_s) \ , \ \ s = r+1, \dots, t \ . \tag{3.7}$$

We define a stochastic kernel

$$\rho^{\gamma}_{r:t} : \mathbb{H}_r \to \Delta(\mathbb{H}_t) \tag{3.8a}$$

by, for any $\varphi : \mathbb{H}_t \to [0, +\infty]$, measurable nonnegative numerical function,[1]

$$\int_{\mathbb{H}_t} \varphi(h'_r, h'_{r+1:t}) \rho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_t) = \int_{\mathbb{W}_{r+1:t}} \varphi\big(\Phi^{\gamma}_{r:t}(h_r, w_{r+1:t})\big) \prod_{s=r+1}^{t} \rho_{s-1:s}\big(\Phi^{\gamma}_{r:s-1}(h_r, w_{r+1:s-1}), \mathrm{d}w_s\big) \ . \tag{3.8b}$$

- *When $0 \leq r = t \leq T$, we define*

$$\rho^{\gamma}_{r:r} : \mathbb{H}_r \to \Delta(\mathbb{H}_r) \ , \ \ \rho^{\gamma}_{r:r}(h_r, \mathrm{d}h'_r) = \delta_{h_r}(\mathrm{d}h'_r) \ . \tag{3.8c}$$

The stochastic kernels $\rho^{\gamma}_{r:t}$ on $\mathbb{H}_t$, given by (3.8), are of the form

$$\rho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_t) = \rho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_r \mathrm{d}h'_{r+1:t}) = \delta_{h_r}(\mathrm{d}h'_r) \otimes \varrho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_{r+1:t}) \ , \tag{3.9}$$

where, for each $h_r \in \mathbb{H}_r$, the probability distribution $\varrho^{\gamma}_{r:t}(h_r, \mathrm{d}h'_{r+1:t})$ only charges the histories visited by the flow from $r+1$ to t.

Proposition 3.2.2 (Flow property)**.** *The family $\{\rho^{\gamma}_{s:t}\}_{r \leq s \leq t}$ of stochastic kernels. given in Definition 3.2.1, has the flow property, that is, for $s < t$,*

$$\rho^{\gamma}_{s:t}(h_s, \mathrm{d}h'_t) = \int_{\mathbb{W}_{s+1}} \rho_{s:s+1}\big(h_s, \mathrm{d}w_{s+1}\big) \rho^{\gamma}_{s+1:t}\Big(\big(h_s, \gamma_s(h_s), w_{s+1}\big), \mathrm{d}h'_t\Big) \ . \tag{3.10}$$

Family of optimization problems with stochastic kernels. To build a family of optimization problems over the time span $\{0, \ldots, T-1\}$, we need two ingredients:

- a family $\{\rho_{s-1:s}\}_{1 \leq s \leq T}$ of stochastic kernels

$$\rho_{s-1:s} : \mathbb{H}_{s-1} \to \Delta(\mathbb{W}_s) \ , \ \ s = 1, \ldots, T \ , \tag{3.11}$$

- a numerical function, playing the role of a cost to be minimized,

$$j : \mathbb{H}_T \to [0, +\infty] \ , \tag{3.12}$$

  assumed to be nonnegative[2] and measurable with respect to the field $\mathcal{H}_T$.

We define, for any $\{\gamma_s\}_{s=t,\ldots,T-1} \in \Gamma_{t:T-1}$,

$$V^{\gamma}_t(h_t) = \int_{\mathbb{H}_T} j(h'_T) \rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \ , \ \ \forall h_t \in \mathbb{H}_t \ , \tag{3.13}$$

[1] We could also consider any $\varphi : \mathbb{H}_t \to \mathbb{R}$, measurable bounded function, or measurable and uniformly bounded below function. However, for the sake of simplicity, we will deal in the sequel with measurable nonnegative numerical functions.

[2] See Footnote 1. When $j(h_T) = +\infty$, this materializes joint constraints between uncertainties and controls.

where $\rho^{\gamma}_{t:T}$ is defined by (3.8). We consider the family of optimization problems, indexed by $t = 0, \dots, T-1$ and parameterized by $h_t \in \mathbb{H}_t$:

$$\inf_{\gamma_{t:T-1} \in \Gamma_{t:T-1}} \int_{\mathbb{H}_T} j(h'_T) \rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \; . \tag{3.14}$$

For all $t = 0, \dots, T-1$, we define the minimum value of Problem (3.14) by

$$V_t(h_t) = \inf_{\gamma_{t:T-1} \in \Gamma_{t:T-1}} \int_{\mathbb{H}_T} j(h'_T) \rho^{\gamma}_{t:T}(h_t, \mathrm{d}h'_T) \tag{3.15a}$$

$$= \inf_{\gamma_{t:T-1} \in \Gamma_{t:T-1}} V^{\gamma}_t(h_t) \; , \quad \forall h_t \in \mathbb{H}_t \; , \tag{3.15b}$$

and we also define

$$V_T(h_T) = j(h_T) \; , \quad \forall h_T \in \mathbb{H}_T \; . \tag{3.15c}$$

The last notation is consistent with (3.14) by the definition (3.8c) of the stochastic kernel $\rho^{\gamma}_{T:T}$. The numerical function $V_t : \mathbb{H}_t \to [0, +\infty]$ is called *value function* at time t.

Resolution by Stochastic Dynamic Programming. Now, we show that the value functions in (3.15) are *Bellman functions*, in that they are solution of the Bellman or Dynamic Programming equation.

The following two assumptions will be made throughout the whole chapter.

Assumption 3.2.3 (Measurable function)**.** *For all $t = 0, \dots, T-1$ and for all nonnegative measurable numerical function $\varphi : \mathbb{H}_{t+1} \to [0, +\infty]$, the numerical function*

$$h_t \mapsto \inf_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \varphi(h_t, u_t, w_{t+1}) \rho_{t:t+1}(h_t, dw_{t+1}) \tag{3.16}$$

is measurable[3] *from $(\mathbb{H}_t, \mathcal{H}_t)$ to $[0, +\infty]$.*

Assumption 3.2.4 (Measurable selection)**.** *For all $t = 0, \dots, T-1$, there exists a* measurable selection,[4] *that is, a measurable mapping*

$$\gamma^{\sharp}_t : (\mathbb{H}_t, \mathcal{H}_t) \to (\mathbb{U}_t, \mathcal{U}_t) \tag{3.17a}$$

such that

$$\gamma^{\sharp}_t(h_t) \in \operatorname*{arg\,min}_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} V_{t+1}(h_t, u_t, w_{t+1}) \rho_{t:t+1}(h_t, \mathrm{d}w_{t+1}) \; , \tag{3.17b}$$

where the numerical function V_{t+1} is given by (3.15)*.*

Bellman operators. For $t = 0, \dots, T$, let $\mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t)$ be the space of nonnegative measurable numerical functions over $\mathbb{H}_t$.

Definition 3.2.5. *For $t = 0, \dots, T-1$, we define the* Bellman operator

$$\mathcal{B}_{t+1:t} : \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1}) \to \mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t) \tag{3.18a}$$

[3] This is a delicate issue, treated in Bertsekas and Shreve (1996).

[4] See Bertsekas and Shreve (1996) and Rockafellar and Wets (1998) for a precise definition of a measurable selection.

such that, for all $\varphi \in \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1})$ *and for all* $h_t \in \mathbb{H}_t$,

$$\big(\mathcal{B}_{t+1:t}\varphi\big)(h_t) = \inf_{u_t \in \mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \varphi(h_t, u_t, w_{t+1}) \rho_{t:t+1}(h_t, dw_{t+1}) \; . \tag{3.18b}$$

Since $\varphi \in \mathbb{L}^0_+(\mathbb{H}_{t+1}, \mathcal{H}_{t+1})$, we have that $\mathcal{B}_{t+1:t}\varphi$ is a well defined nonnegative numerical function and, by Assumption 3.2.3, we know that $\mathcal{B}_{t+1:t}\varphi$ is a measurable numerical function, hence belongs to $\mathbb{L}^0_+(\mathbb{H}_t, \mathcal{H}_t)$.

Bellman equation and optimal history feedback policies. We are able to state the main result of §3.2.1.2, that is, a Dynamic Programming equation without any independence between the uncertainties.

Theorem 3.2.6. *The value functions in* (3.15) *satisfy the* Bellman equation*, or* (Stochastic) Dynamic Programming equation

$$V_T = j \; , \tag{3.19a}$$
$$V_t = \mathcal{B}_{t+1:t} V_{t+1} \text{ for } t = T-1, \dots, 0 \; . \tag{3.19b}$$

Moreover, a solution to any Problem (3.14) *— that is, whatever the index* $t = 0, \dots, T-1$ *and the parameter* $h_t \in \mathbb{H}_t$ *— is any history feedback* $\gamma^\sharp = \big\{\gamma_s^\sharp\big\}_{s=t,\dots,T-1}$ *defined by the collection of mappings* $\gamma_s^\sharp$ *in* (3.17).

Proof. We refer to Carpentier et al. (2018a) for the proof. □

3.2.2. Compressing history in a state process

It is possible that there exists a state reduction $x_t = \theta_t(h_t)$ carrying enough information to compute the Bellman value functions (3.19).

History and reduction mappings. For $t \in \{0, .., T\}$, let $\mathbb{X}_t$ be a measurable set equipped with a σ-field $\mathcal{X}_t$. We suppose that there exists measurable reduction mappings

$$\theta_t : (\mathbb{H}_t, \mathcal{H}_t) \to (\mathbb{X}_t, \mathcal{X}_t) \; , \;\; \forall t \in \{0, .., T\} \; , \tag{3.20a}$$

and measurable dynamics

$$f_{t:t+1} : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{X}_{t+1} \; , \;\; \forall t \in \{0, .., T-1\} \; , \tag{3.20b}$$

such that for all $t \in \{0, .., T-1\}$

$$\theta_{t+1}(h_t, u_t, w_{t+1}) = f_{t:t+1}\big(\theta_t(h_t), u_t, w_{t+1}\big) \; , \;\; \forall (h_t, u_t, w_{t+1}) \in \mathbb{H}_{t+1} \; . \tag{3.20c}$$

We define formally the state $x_t \in \mathbb{X}_t$ as

$$x_t = \theta_t(h_t) \; . \tag{3.20d}$$

Figure 3.1 illustrates the link between the history h_t and the state x_t across the different time-steps.

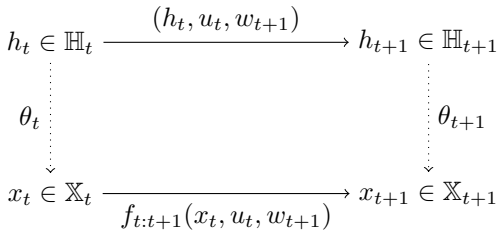


Figure 3.1.: Linking history h_t with the state x_t.

Compatibility of states with stochastic kernels. Let $\{\rho_{t:t+1}\}_{t\in\{0,..,T-1\}}$ be the family of stochastic kernels as in Equation (3.11).

Definition 3.2.7. *The state reductions* $\{\theta_t\}_{t\in\{0,..,T\}}$ *in* (3.20a) *are* compatible *with the family of stochastic kernels* $\{\rho_{t:t+1}\}_{t\in\{0,..,T-1\}}$ *in* (3.11) *if there exists reduced stochastic kernels*

$$\overline{\rho}_{t:t+1} : \mathbb{X}_t \to \Delta(\mathbb{W}_{t+1}) \,, \tag{3.21}$$

such that for all $t \in \{0, .., T-1\}$ *we have*

$$\rho_{t:t+1}(h_t, dw_{t+1}) = \overline{\rho}_{t:t+1}\big(\theta_t(h_t), dw_{t+1}\big) \,, \quad \forall h_t \in \mathbb{H}_t \,. \tag{3.22}$$

Reduced Bellman operators. Let $\{\theta_t\}_{t\in\{0,..,T\}}$ be a family of state reduction mappings compatible with the stochastic kernels $\{\rho_{t:t+1}\}_{t\in\{0,..,T-1\}}$. Then, using (Carpentier et al., 2018a, Theorem 2), we know that there exists, for all time $t \in \{0, .., T-1\}$, a *reduced Bellman operator*

$$\overline{\mathcal{B}}_{t+1:t} : \mathbb{L}^0(\mathbb{X}_{t+1}, \mathcal{X}_{t+1}) \to \mathbb{L}^0(\mathbb{X}_t, \mathcal{X}_t) \,, \tag{3.23}$$

such that for any real valued function $\overline{\psi}_{t+1} : \mathbb{X}_{t+1} \to \mathbb{R}$, we have

$$(\overline{\mathcal{B}}_{t+1:t}\overline{\psi}_{t+1}) \circ \theta_{t+1} = \mathcal{B}_{t+1:t}(\overline{\psi}_{t+1} \circ \theta_{t+1}) \,, \tag{3.24}$$

where $\mathcal{B}_{t+1:t}$ is the Bellman operator defined in Equation (3.18).

3.2.3. A template to design online policies

The exact resolution of Bellman equations (3.19) is generally out of reach, as is the exact computation of the optimal Bellman policies $\{\gamma_t^\sharp\}_{t\in\{0,..,T-1\}}$ given by Theorem 3.2.6. Practitioners use Equations (3.17) and (3.19) as *templates* to design *online* policies $\{\gamma_t\}_{t\in\{0,..,T-1\}}$.

3.2.3.1. Cost-to-go policies

Cost-to-go policies use Equation (3.17) as a template to compute a decision u_t at time t. At time $t \in \{0, .., T-1\}$, cost-to-go policies write as solution of a one-stage problem:

$$\gamma_t(h_t) \in \operatorname*{arg\,min}_{u_t\in\mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \widetilde{V}_{t+1}(h_t, u_t, w_{t+1})\widetilde{\rho}_{t:t+1}(h_t, dw_{t+1}) \,. \tag{3.25}$$

Cost-to-go policies have two ingredients: a cost-to-go $\widetilde{V}_{t+1} : \mathbb{H}_{t+1} \to \mathbb{R}$ and a stochastic kernel $\widetilde{\rho}_{t:t+1} : \mathbb{H}_t \to \Delta(\mathbb{W}_{t+1})$.

3.2.3.2. Lookahead policies

Whereas cost-to-go policies take inspiration from Equation (3.17), *lookahead policies* use directly Equation (3.15) as a template. Lookahead policies write as solution of a multistage problem:

$$\begin{aligned}\gamma_t(h_t) \in \underset{u_t \in \widetilde{\mathbb{U}}_t}{\arg\min} \min_{\widetilde{\gamma}_{t+1:\widetilde{T}-1}} & \int_{\mathbb{H}_{\widetilde{T}}} \widetilde{V}_{\widetilde{T}}(h_{\widetilde{T}}) \widetilde{\rho}^{\widetilde{\gamma}}_{t:\widetilde{T}}(h_t, \mathrm{d}h_{\widetilde{T}}) \\ \text{s.t. } & h_{\widetilde{T}} = (h_t, u_t, w_{t+1}, \widetilde{\gamma}_{t+1}(h_{t+1}), w_{t+2}, \cdots, \widetilde{\gamma}_{t+1}(h_{\widetilde{T}-1}), w_{\widetilde{T}}) \ , \\ & \widetilde{\gamma}_{t+1:\widetilde{T}-1} \in \widetilde{\Gamma}_{t+1:\widetilde{T}-1} \ .\end{aligned} \tag{3.26}$$

We detail hereafter the different ingredients appearing in Equation (3.26). The procedure is as follows.

- Set an *horizon* $\widetilde{T} \in \{t+1, .., T\}$.
- Set a future cost-to-go $\widetilde{V}_{\widetilde{T}} : \mathbb{H}_{\widetilde{T}} \to \mathbb{R}$. We note that by setting $\widetilde{T} = T$ and $\widetilde{V}_{\widetilde{T}} = j$, we recover the original setting introduced in (3.15).
- Set two sequences of σ-fields:
  1. a sequence of *noise* fields $\{\widetilde{\mathcal{W}}_s\}_{s \in \{t+1, .., \widetilde{T}\}}$;
  2. a sequence of *lookahead* fields $\{\widetilde{\mathcal{H}}_s\}_{s \in \{t+1, .., \widetilde{T}-1\}}$.

  By definition, stochastic kernels (3.8) depend on two σ-fields: the σ-field that restrain the measurability w.r.t. the first argument (here corresponding to the history h_t) and the σ-field defining the input space of the second argument (that is, the sets we are able to measure with the stochastic kernel). Thus, we are able to use the two sequences of lookahead fields and noise fields to define accordingly each stochastic kernel $\widetilde{\rho}_{s:s+1}$ in $\widetilde{\rho}^{\gamma}_{t:\widetilde{T}}$ (3.8b), such as

$$\widetilde{\rho}_{s:s+1} : \mathbb{H}_s \times \widetilde{\mathcal{W}}_{s+1} \to [0, 1] \ , \ \ \forall s \in \{t, .., \widetilde{T} - 1\} \ , \tag{3.27}$$

  where, for all $W \in \widetilde{\mathcal{W}}_{s+1}$, $\widetilde{\rho}_{s:s+1}(\cdot, W)$ is measurable w.r.t. $\widetilde{\mathcal{H}}_s$, and for all $h_s \in \mathbb{H}_s$, $\widetilde{\rho}_{s:s+1}(h_s, \cdot)$ is a probability measure on $\widetilde{\mathcal{W}}_{s+1}$.
- Set a space of admissible history feedback policies $\widetilde{\Gamma}_{t+1:\widetilde{T}-1}$ as

$$\widetilde{\Gamma}_{t+1:\widetilde{T}-1} = \left\{ (\widetilde{\gamma}_{t+1}, \cdots, \widetilde{\gamma}_{\widetilde{T}-1}) \mid \widetilde{\gamma}_s : (\mathbb{H}_s, \widetilde{\mathcal{I}}_s) \to (\mathbb{U}_s, \mathcal{U}_s) \ , \ \ \forall s \in \{t+1, .., \widetilde{T}-1\} \right\} , \tag{3.28}$$

  where $\{\widetilde{\mathcal{I}}_s\}_{s \in \{t+1, .., \widetilde{T}-1\}}$ is a sequence of *information* σ-fields. To ensure that the admissible history feedbacks γ_s in $\widetilde{\Gamma}_{t+1:\widetilde{T}-1}$ are compatible with the stochastic kernels (3.27), we impose that

$$\widetilde{\mathcal{I}}_s \subset \widetilde{\mathcal{H}}_s \ , \ \ \forall s \in \{t+1, .., \widetilde{T}-1\} \ . \tag{3.29}$$

Discussion

We introduced two classes of online policies: the *cost-to-go* based policies and the *lookahead* policies. One salient difference between these two templates is the time span considered. The distinction between cost-to-go policies and lookahead is similar to the distinction between *explicit cost-to-go approximation* and *implicit cost-to-go approximation* introduced in Bertsekas (2005b).

3.3. A template for lookahead policies

In Sect. 3.2, we have presented an overview of the different ingredients required to solve stochastic optimization problems and have introduced two templates to design *online* policies: the class of *cost-to-go* policies and the class of *lookahead* policies. We focus in this section on lookahead policies. We first depict the general structure of lookahead policies in §3.3.1, and then frame three classical algorithms (Model Predictive Control, Open-Loop Feedback Control and Stochastic Programming) with the lookahead template in §3.3.2.

3.3.1. Design of lookahead policies

We detailed in §3.2.3 the four ingredients of lookahead policies.

1. A horizon $\widetilde{T} \in \{t+1, .., T\}$.
2. A final cost-to-go $\widetilde{V}_{\widetilde{T}} : \mathbb{H}_{\widetilde{T}} \to \mathbb{R}$ measurable w.r.t. $\widetilde{\mathcal{H}}_{\widetilde{T}}$.
3. A sequence of σ-fields $\{\widetilde{\mathcal{W}}_s\}_{s\in\{t+1,..,\widetilde{T}\}}$: and a sequence of lookahead fields $\{\widetilde{\mathcal{H}}_s\}_{s\in\{t+1,..,\widetilde{T}\}}$ parameterizing the measurability of the stochastic kernels $\{\widetilde{\rho}_{s:s+1}\}_{s\in\{t+1,..,\widetilde{T}\}}$

$$\widetilde{\rho}_{s:s+1} : \mathbb{H}_s \times \widetilde{\mathcal{W}}_{s+1} \to [0,1] \ , \ \ s \in \{t+1, .., \widetilde{T}-1\} \ , \tag{3.30}$$

   such that for all $s \in \{t+1, .., \widetilde{T}-1\}$ and $W \in \widetilde{\mathcal{W}}_{s+1}$, $h_s \to \widetilde{\rho}_{s:s+1}(h_s, W)$ is $\widetilde{\mathcal{H}}_s$-measurable.
4. A sequence of information fields $\widetilde{\mathcal{I}}_{t+1:\widetilde{T}} = \{\widetilde{\mathcal{I}}_s\}_{s\in\{t+1,..,\widetilde{T}-1\}}$ parameterizing the admissible history feedbacks

$$\gamma_s : (\mathbb{H}_s, \widetilde{\mathcal{I}}_s) \to (\mathbb{U}_s, \mathcal{U}_s) \ , \ \ s = t+1, \cdots, \widetilde{T}-1 \ . \tag{3.31}$$

We now discuss these four ingredients more in detail.

Choosing the horizon. The horizon $\widetilde{T}$ belongs to $\{t+1, .., T\}$.

Choosing the stochastic kernels family. The sequence of lookahead fields $\widetilde{\mathcal{H}}_{t+1:\widetilde{T}}$ and the associated stochastic kernels $\{\widetilde{\rho}_{s:s+1}\}_{s\in\{t+1,..,\widetilde{T}\}}$ models the *views* of the decision-maker regarding the uncertainties after state $t+1$. We present hereunder three classes of stochastic kernels.

Constant noise process. The noise fields write

$$\widetilde{\mathcal{W}}_s = \{\emptyset, \sigma(\{\overline{w}_s\}), \mathbb{W}_s\} \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ . \tag{3.32a}$$

and the lookahead fields $\{\widetilde{\mathcal{H}}_s\}_{s\in\{t+1,..,\widetilde{T}-1\}}$ write accordingly

$$\widetilde{\mathcal{H}}_s^c = \bigotimes_{r=t}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \widetilde{\mathcal{W}}_{r+1} \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ . \tag{3.32b}$$

By doing so, we disregard information from previous controls because we only consider the control σ-fields $\{\emptyset, \mathbb{U}_r\}$. The stochastic kernels defined with the noise fields (3.32) are able to attach a probability only to the singletons $\{\overline{w}_s\}_{s\in\{t+1,..,\widetilde{T}-1\}}$.

Finite scenario tree. This class extends the constant case by considering finite noise fields

$$\widetilde{\mathcal{W}}_s = \{\emptyset, \sigma(\{\overline{w}_s^1\}, \cdots, \{\overline{w}_s^N\}), \mathbb{W}_s\} \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ , \tag{3.33a}$$

from which we build the corresponding lookahead fields

$$\widetilde{\mathcal{H}}_s^d = \bigotimes_{r=t}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \widetilde{\mathcal{W}}_{r+1} \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ . \tag{3.33b}$$

The lookahead fields (3.33b) disregard information from previous controls.

Generic noise scenario. In this case, the noise fields $\widetilde{\mathcal{W}}_s$ are not necessarily finite and can encode any information structure for the noise. The lookahead fields write

$$\widetilde{\mathcal{H}}_s^w = \bigotimes_{r=t}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \mathcal{W}_{r+1} \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ . \tag{3.34}$$

Again, we disregard in (3.34) information from previous decisions.

Choosing the set of policies. Let $\widetilde{\mathcal{W}}_{t+1}, \cdots, \widetilde{\mathcal{W}}_{\widetilde{T}}$ be noise fields and $\widetilde{\mathcal{H}}_{t+1}, \cdots, \widetilde{\mathcal{H}}_{\widetilde{T}}$ lookahead fields. We introduce the information fields $\widetilde{\mathcal{I}}_{t+1}, \cdots, \widetilde{\mathcal{I}}_{\widetilde{T}-1}$ to define admissible history feedback policies. We introduce two different information structures.

- *Open-loop policies* handle as information fields:

$$\widetilde{\mathcal{I}}_s = \bigotimes_{r=t}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \bigotimes_{r=t+1}^{s} \{\emptyset, \mathbb{W}_r\} \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ . \tag{3.35}$$

  Any policy $\widetilde{\gamma}_s : (\mathbb{H}_s, \widetilde{\mathcal{I}}_s) \to (\mathbb{U}_s, \mathcal{U}_s)$ is a constant policy.

- *Closed-loop policies* take advantage of all information available at time t:

$$\widetilde{\mathcal{I}}_s = \widetilde{\mathcal{H}}_s \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ . \tag{3.36}$$

Choosing a final cost-to-go. Lookahead policies deal with any final cost-to-go $\widetilde{V}_{\widetilde{T}} : \mathbb{H}_{\widetilde{T}} \to \mathbb{R}$.

3.3.2. Classical lookahead methods

The design of lookahead policies depend on two ingredients: the σ-fields $\widetilde{\mathcal{H}}_{t+1:\widetilde{T}}$ used to restrain the stochastic kernels $\widetilde{\rho}_{s:s+1}$ in Equation (3.30) and the σ-fields $\widetilde{\mathcal{I}}_{t+1:\widetilde{T}-1}$ used to restrain the measurability of policies $\gamma_{t+1:\widetilde{T}-1}$ in Equation (3.31).

We enumerate different combinations of stochastic kernels and policies spaces in Table 3.1. We discuss further the different possible choices detailing Model Predictive Control (MPC), Open-loop Feedback Control (OLFC) and Stochastic Programming (SP) in the sequel.

		Open-loop	Closed-loop
	$\widetilde{\mathcal{I}}_s$ / $\widetilde{\mathcal{H}}_s$	$\bigotimes_{r=t}^{s-1}\{\emptyset,\mathbb{U}_r\} \otimes \bigotimes_{r=t+1}^{s}\{\emptyset,\mathbb{W}_r\}$	$\widetilde{\mathcal{H}}_s$
Constant	(3.32) ($\bigotimes_{r=t}^{s-1}\{\emptyset,\mathbb{U}_r\} \otimes \{\emptyset, \sigma(\{\overline{w}_{r+1}\}), \mathbb{W}_{r+1}\}$)	MPC	$\emptyset$
Scenario	(3.33b) ($\bigotimes_{r=t}^{s-1}\{\emptyset,\mathbb{U}_r\} \otimes \widetilde{\mathcal{W}}_{r+1}$)	OLFC	SP
Generic	(3.34) ($\bigotimes_{r=t}^{s-1}\{\emptyset,\mathbb{U}_r\} \otimes \mathcal{W}_r$)	OLFC	?

Table 3.1.: Classification of online policies as function of stochastic kernels and policy spaces

3.3.2.1. Model Predictive Control

Model Predictive Control (MPC) (see Garcia et al. (1989) and Bertsekas (2005b)) is a well-known algorithm that tackles uncertainties by using deterministic forecasts.

Ingredients.

- *Horizon.* $\widetilde{T} \in \{t+1, .., T\}$ is given.
- *Stochastic kernels.* MPC selects the *constant information* noise and lookahead fields (3.32):

$$\widetilde{\mathcal{H}}_s^c = \bigotimes_{r=t}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \bigotimes_{r=t+1}^{s} \left\{\emptyset, \sigma(\{\overline{w}_r\}), \mathbb{W}_r\right\} , \ \ \forall s \in \{t+1, .., \widetilde{T}\} .$$

  Then, MPC takes stochastic kernels that are Dirac measures on a given value $\overline{w}_{s+1} \in \mathbb{W}_{s+1}$:

$$\widetilde{\rho}_{s:s+1}(h_s, \cdot) = \delta_{\overline{w}_{s+1}}(\cdot) \ , \forall s \in \{t, .., \widetilde{T} - 1\} . \tag{3.37}$$

- *Policy space.* We use open-loop policies (3.35) $\widetilde{\gamma}_s : (\mathbb{H}_s, \widetilde{\mathcal{I}}_s) \to (\mathbb{U}_s, \mathcal{U}_s)$, where

$$\widetilde{\mathcal{I}}_s = \bigotimes_{r=t+1}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \bigotimes_{r=t+1}^{s} \{\emptyset, \mathbb{W}_r\} , \ \ \forall s \in \{t+1, .., \widetilde{T}\} , \tag{3.38}$$

  so that the history feedback policies can be identified with constant values $\{u_s\}_{s=t+1,\dots,\widetilde{T}}$.

- *Final cost.* The final cost is given.

We depict the information scheme used by MPC in Figure 3.2.

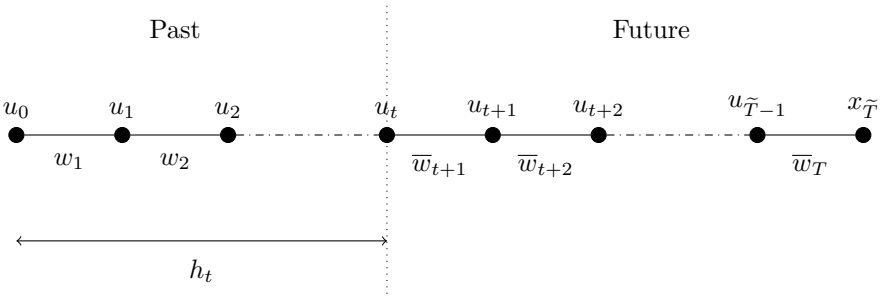


Figure 3.2.: MPC envisages the future with a deterministic forecast

Online problem formulation. Problem (3.26) rewrites

$$\gamma_t(h_t) \in \operatorname*{arg\,min}_{u_t \in \mathbb{U}_t} \min_{u_{t+1:\widetilde{T}-1} \in \mathbb{U}_{t+1:\widetilde{T}-1}} \widetilde{V}_{\widetilde{T}}(h_t, u_t, \overline{w}_{t+1}, \dots, u_{\widetilde{T}-1}, \overline{w}_{\widetilde{T}}) . \tag{3.39}$$

Problem (3.39) is a deterministic optimization problem, possibly solvable by proper mathematical programming methods.

3.3.2.2. Open Loop Feedback Control

Open Loop Feedback Control (OLFC) (see (Bertsekas, 2012, Chapter 6)) is another well-known control method. We detail hereby the features of OLFC.

Ingredients

- *Horizon.* $\widetilde{T} \in \{t+1, .., T\}$ is given.
- *Stochastic kernels.* OLFC selects *generic* noise and lookahead fields (3.34), with

$$\widetilde{\mathcal{H}}_s^w = \bigotimes_{r=t}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \widetilde{\mathcal{W}}_r \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ .$$

  The stochastic kernels $\{\widetilde{\rho}_{s:s+1}\}_{s \in \{t+1,..,\widetilde{T}-1\}}$ do not depend on previous decisions. There exist stochastic kernels $\overline{\rho}_{s:s+1} : \mathbb{W}_{t+1:s} \times \widetilde{\mathcal{W}}_s \to [0,1]$ such that

$$\widetilde{\rho}_{s:s+1}(h_s, \cdot) = \overline{\rho}_{s:s+1}([h_s]_{t+1:s}^{\mathbb{W}}, \cdot) \ , \ \ \forall s \in \{t+1, .., \widetilde{T}-1\} \ , \tag{3.40}$$

  where $[h_s]_{t+1:s}^{\mathbb{W}}$ is the history uncertainty part defined in Equation (3.2e):

$$[h_s]_{t+1:s}^{\mathbb{W}} = (w_{t+1}, \cdots, w_s) \ .$$

- *Policy space.* OLFC uses open-loop policies (3.35), that is, $\widetilde{\gamma}_s : (\mathbb{H}_s, \widetilde{\mathcal{I}}_s) \to (\mathbb{U}_s, \mathcal{U}_s)$ with

$$\widetilde{\mathcal{I}}_s = \bigotimes_{r=t+1}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \bigotimes_{r=t+1}^{s} \{\emptyset, \mathbb{W}_r\} \ , \ \ \forall s \in \{t+1, .., \widetilde{T}\} \ ,$$

  so that any history feedback policies $\widetilde{\gamma}_{t+1}, \cdots, \widetilde{\gamma}_{\widetilde{T}-1}$ can be identified with a sequence of constant values $\{u_s\}_{s \in \{t+1,..,\widetilde{T}-1\}}$.

- *Final cost.* The final cost is given.

We depict OLFC information scheme in Figure 3.3.

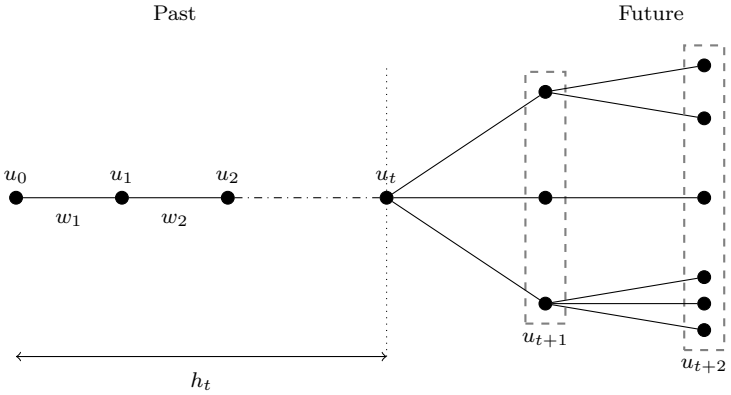


Figure 3.3.: OLFC depicts the future as a collection of scenarios (in this picture, the kernels are discrete so that the noise structure is a tree)

Online problem formulation. Problem (3.26) becomes

$$\gamma_t(h_t) \in \operatorname*{arg\,min}_{u_t \in \mathbb{U}_t} \min_{u_{t+1:\widetilde{T}-1} \in \mathbb{U}_{t+1:\widetilde{T}-1}} \int_{\mathbb{W}_{t+1:\widetilde{T}}} \widetilde{V}_{\widetilde{T}}(h_t, u_t, w_{t+1}, \ldots, u_{\widetilde{T}-1}, w_{\widetilde{T}}) \ \widetilde{\rho}_{t+1:\widetilde{T}}^{u_{t+1:\widetilde{T}-1}}(h_t, dw_{t+1:\widetilde{T}}) \ . \tag{3.41}$$

3.3.2.3. Stochastic Programming

Stochastic Programming (Shapiro et al., 2009) makes use of finite partitions to model the noise fields $\{\widetilde{\mathcal{W}}_s\}_{s\in\{t+1,..,\widetilde{T}\}}$. Then, it introduces stochastic kernels compatible with the finite partitions, thus enconding a scenario tree. A decision is attached to every node of the tree. The procedure has the following features.

Ingredients

- *Horizon.* $\widetilde{T} \in \{t+1,..,T\}$ is given.
- *Stochastic kernels.* Stochastic Programming selects *finite* noise and lookahead fields (3.33b), with
$$\widetilde{\mathcal{H}}_s^d = \bigotimes_{r=t}^{s-1} \{\emptyset, \mathbb{U}_r\} \otimes \widetilde{\mathcal{W}}_{t+1:s} \ , \ \ \forall s \in \{t+1,..,\widetilde{T}\} \ .$$
Every stochastic kernel has *discrete support* depending on previous noises:
$$\widetilde{\rho}_{s:s+1}(h_s, \cdot) = \sum_{i=1}^{S} \pi_i([h_s]_{t+1:s}^{\mathbb{W}}) \delta_{w_i([h_s]_{t+1:s}^{\mathbb{W}})}(\cdot) \ , \tag{3.42}$$
where $[h_s]_{t+1:s}^{\mathbb{W}}$ is the history uncertainty part defined in Equation (3.2e):
$$[h_s]_{t+1:s}^{\mathbb{W}} = (w_{t+1}, \cdots, w_s) \ .$$
- *Policy space.* Stochastic Programming considers closed-loop policies as in Equation (3.36), a that is,
$$\widetilde{\mathcal{I}}_s = \widetilde{\mathcal{H}}_s^d \ , \ \ \forall s \in \{t+1,..,\widetilde{T}-1\} \ . \tag{3.43}$$
Thus, the policies depend only on previous uncertainties.
- *Final cost.* The final cost is given.

We depict Stochastic Programming information structure in Figure 3.4.

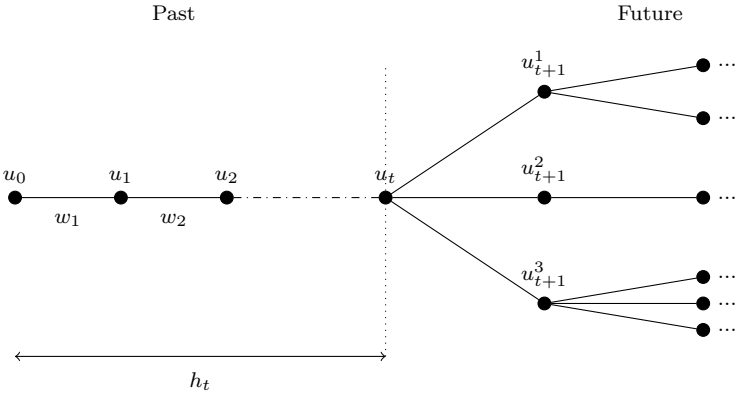


Figure 3.4.: Stochastic Programming represents the future as an arborescent tree with a decision at each node.

Discussion

We have detailed in this section a general structure for lookahead policies and have presented Stochastic Programming method as a generalization of Open-Loop Feedback Control, itself a generalization of the Model Predictive Control algorithm.

Lookahead algorithms correspond to the *implicit cost-to-go approximation* presented in Bertsekas (2005b), where the future cost $\widetilde{V}_{t+1}$ is estimated online by restricting the set of admissible policies γ_s. Most algorithms that rely on *forecasts* are lookahead algorithms. For instance, we are able to frame the Stochastic Model Predictive Control algorithm either as a Open-Loop Feedback Control or as Stochastic Programming method, depending on the information structures that define the set of future policies.

In the following section, we will focus on the cost-to-go policies, corresponding to the *explicit cost-to-go approximation* of Bertsekas (2005b).

3.4. A template for cost-to-go policies

We focus in this section on *cost-to-go* policies (3.25). We first describe the generic template of cost-to-go policies in §3.4.1 and then frame in §3.4.2 three classical algorithms (Stochastic Dynamic Programming, Stochastic Dual Dynamic Programming and Approximate Dynamic Programming) with the template of cost-to-go policies.

3.4.1. A general sketch to design cost-to-go policies

We saw in §3.2.3 that the two ingredients of cost-to-go policies are:

1. a stochastic kernel $\widetilde{\rho}_{t:t+1} : \mathbb{H}_t \to \Delta(\mathbb{W}_{t+1})$;

2. a cost-to-go $\widetilde{V}_{t+1} : \mathbb{H}_{t+1} \to \mathbb{R}$.

The first ingredient has been discussed in the description of lookahead policies (see §3.3), and the procedure to design the stochastic kernel is similar for cost-to-go policies. However, cost-to-go policies require an explicit cost-to-go $\widetilde{V}_{t+1} : \mathbb{H}_{t+1} \to \mathbb{R}$ as input.

In Bertsekas (2005b), two solutions are investigated to compute *explicitly* such a family of cost-to-go $\{\widetilde{V}_{t+1}\}_{t\in\{0,..,T\}}$:

a) Compute *online* cost-to-go with a parametric approximation tuned by some *heuristic* or *systematic* methods. Such methods encompass those used in the reinforcement learning community.

b) Solve *offline* backward recursive equations mimicking the Bellman equations (3.19), that is, choosing Bellman operators $\widetilde{\mathcal{B}}_{t+1:t}$ such that

$$\widetilde{V}_t = \widetilde{\mathcal{B}}_{t+1:t}\widetilde{V}_{t+1} \ , \ \ \forall\, t = T-1, \cdots, 0 \ , \tag{3.44}$$

with a given $\widetilde{V}_T$. Then, we plug the cost-to-go $\widetilde{V}_{t+1}$ given by (3.44) in Equation (3.25).

We choose here to focus on the second method. The choice of the pseudo-Bellman operators $\widetilde{\mathcal{B}}_{t+1:t}$ in the resolution of Equation (3.44) frames the cost-to-go algorithm.

3.4.2. Classical cost-to-go methods

We focus here on the resolution of the recursive Equation (3.44) with proper operators $\widetilde{\mathcal{B}}_{t+1:t}$. We will show that we are able to frame well-known algorithms (Stochastic Dynamic Programming, Stochastic Dual Dynamic Programming and Approximate Dynamic Programming) only by describing their corresponding operators $\widetilde{\mathcal{B}}_{t+1:t}$.

3.4.2.1. Obtaining Bellman-like equations for cost-to-go

The design of the operator $\widetilde{\mathcal{B}}_{t+1:t}$ is done in three steps.

Choosing offline stochastic kernels. First, we choose *offline stochastic kernels* $\{\widetilde{\rho}^{of}_{t:t+1}\}_{t\in\{0,..,T-1\}}$ such that

$$\widetilde{\rho}^{of}_{t:t+1} : \mathbb{H}_t \to \Delta(\mathbb{W}_{t+1}) \ , \ \ \forall t \in \{0,..,T-1\} \ . \tag{3.45}$$

Choosing the Bellman backward operator. Then, we use the stochastic kernels in Equation (3.45) to define the corresponding *offline Bellman operators* $\mathcal{B}^{of}_{t+1:t} : \mathbb{L}^0(\mathbb{H}_{t+1}, \mathcal{H}_{t+1}) \to \mathbb{L}^0(\mathbb{H}_t, \mathcal{H}_t)$ defined, for all t, for all $\varphi \in \mathbb{L}^0(\mathbb{H}_{t+1}, \mathcal{H}_{t+1})$ and for all $h_t \in \mathbb{H}_t$, by:

$$\big(\mathcal{B}^{of}_{t+1:t}\varphi\big)(h_t) = \inf_{u_t\in\mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \varphi(h_t, u_t, w_{t+1}) \widetilde{\rho}^{of}_{t:t+1}(h_t, dw_{t+1}) \ . \tag{3.46}$$

We note that the functions defined recursively as $V^{of}_t = \mathcal{B}^{of}_{t+1:t} V^{of}_{t+1}$ satisfy the Bellman recursive equations (3.19) associated to the stochastic kernels $\{\widetilde{\rho}^{of}_{t:t+1}\}_{t\in\{0,..,T-1\}}$. However, the numerical computation of the offline Bellman functions $\{V^{of}_t\}_{t\in\{0,..,T\}}$ is generally out of reach.

Choosing an operator domain. To ease the resolution of the backward recursive equations (3.44), we introduce the *operator domain*

$$\widetilde{\mathfrak{V}}_t \subset \mathbb{L}^0(\mathbb{H}_t, \mathcal{H}_t) \ , \ \ \forall t \in \{0,..,T-1\} \ . \tag{3.47}$$

and aim to compute a sequence of value functions $\{\widetilde{V}_t\}_{t\in\{0,..,T\}}$ such that, for all $t \in \{0,..,T\}$, $V_t \in \widetilde{\mathfrak{V}}_t$. To do so, we look for Bellman operators in (3.44) such that

$$\widetilde{\mathcal{B}}_{t+1:t} : \widetilde{\mathfrak{V}}_{t+1} \to \widetilde{\mathfrak{V}}_t \ . \tag{3.48}$$

Then, we are able to compute the cost-to-go $\{\widetilde{V}_t\}_{t\in\{0,..,T\}}$ in a backward manner with the recursive equations (3.44).

We note that the operator domain $\widetilde{\mathfrak{V}}_t$ is often generated as a finite linear combination of basis functions $\Phi^1, \cdots, \Phi^N$, such as

$$\widetilde{\mathfrak{V}}_t = \mathrm{span}(\Phi^1, \cdots, \Phi^N) = \Big\{ \sum_{i=1}^{N} \gamma^i \Phi^i \mid \gamma^i \in \mathbb{R} \ , \ \ N \in \mathbb{N}^\star \Big\} \ . \tag{3.49}$$

3.4.2.2. Stochastic Dynamic Programming

In the algorithmic implementation of Stochastic Dynamic Programming (SDP) (see Bertsekas (2012)-Puterman (1994)) one usually approximates the offline Bellman operators (3.46) with *step functions*. If we denote by $H_1, \cdots, H_N$ the subsets (atoms) of a partition of the space $\mathbb{H}_{t+1}$, the

operator domain $\widetilde{\mathfrak{V}}_{t+1}$ (3.47) writes

$$\widetilde{\mathfrak{V}}_{t+1} = \left\{ V : \mathbb{H}_{t+1} \to \mathbb{R} \mid \exists (\gamma_k)_{k\in\{1,..,N\}} \ , \ \gamma_k \in \mathbb{R} \ , \ \text{s.t. } V = \sum_{k=1}^{N} \gamma_k \mathbf{1}_{H_k} \right\} , \tag{3.50}$$

that is,

$$\widetilde{\mathfrak{V}}_{t+1} = \text{span}(\mathbf{1}_{H_1}, \cdots, \mathbf{1}_{H_N}) ,$$

where $\mathbf{1}_{H_k}$ is the indicator function of the subset H_k:

$$\mathbf{1}_{H_k}(x) = \begin{cases} 1 \text{ if } x \in H_k \\ 0 \text{ otherwise.} \end{cases} \tag{3.51}$$

Then, the SDP algorithm computes backward a set of value functions using a discretization of the history space $\mathbb{H}_t$ with a grid $\{h_t^i\}_{i\in\{1,..,N\}} \in \mathbb{H}_t^N$ with size $N \in \mathbb{N}^\star$. Its procedure is as follows.

- We define the *trace operator*

$$\mathfrak{T}_t^N : \mathbb{L}_t^0(\mathbb{H}_t, \mathcal{H}_t) \to \mathbb{H}_t^N \times \mathbb{R}^N , \tag{3.52}$$

  as the operator that associates the finite dimensional representation $\{(h_t^i, \phi_t(h_t^i))\}_{i\in\{1,..,N\}}$ of a function $\phi_t \in \mathbb{L}^0(\mathbb{H}_t, \mathcal{H}_t)$. Stochastic Dynamic Programming apply the trace operator $\mathfrak{T}_t^N$ to the offline Bellman operator $\mathcal{B}_{t+1:t}^{of}$ (3.46).

- We define the *regression-interpolation* operator

$$\mathfrak{R}_{\widetilde{\mathfrak{V}}_t} : \mathbb{H}_t^N \times \mathbb{R}^N \to \widetilde{\mathfrak{V}}_t \tag{3.53}$$

  as the operator that interpolates the values in the grid with a function in the operator domain $\widetilde{\mathfrak{V}}_t$.

We compute $\widetilde{V}_t$ in (3.44) as

$$\widetilde{V}_t = (\mathfrak{R}_{\widetilde{\mathfrak{V}}_t} \circ \mathfrak{T}_t^N \circ \mathcal{B}_{t+1:t}^{of}) \widetilde{V}_{t+1} . \tag{3.54}$$

In Equation (3.44), the backward operators $\widetilde{\mathcal{B}}_{t+1:t} : \widetilde{\mathfrak{V}}_{t+1} \to \widetilde{\mathfrak{V}}_t$ are given by

$$\widetilde{\mathcal{B}}_{t+1:t} = \mathfrak{R}_{\widetilde{\mathfrak{V}}_t} \circ \mathfrak{T}_t^N \circ \mathcal{B}_{t+1:t}^{of} \ , \ \forall t \in \{0, .., T-1\} . \tag{3.55}$$

3.4.2.3. Stochastic Dual Dynamic Programming

Stochastic Dual Dynamic Programming (SDDP) mixes Stochastic Dynamic Programming with convex analysis. The idea behind SDDP was first introduced in Van Slyke and Wets (1969) and extended further in Pereira and Pinto (1991). We refer to Shapiro (2011) for a recent overview of the algorithm. We detail hereafter the SDDP procedure in the history framework.

SDDP takes advantage of a well known result in convex analysis that allows to represent any convex l.s.c. function as a supremum of affine functions. We suppose that the future cost-to-go $\widetilde{V}_{t+1}$ in (3.44) is convex w.r.t. h_{t+1}. Then, SDDP approximates *iteratively* the offline Bellman operators (3.46) by a supremum of *affine functions*. The operator domain (3.47) writes

$$\widetilde{\mathfrak{V}}_{t+1} = \left\{ V : \mathbb{H}_{t+1} \to \mathbb{R} \mid \exists (\lambda^i, \beta^i)_{i\in\{1,..,N\}} \ , \ \text{such that } V(h) = \max_{i=1\cdots N} \left\langle \lambda^i , h \right\rangle + \beta^i \right\} . \tag{3.56}$$

SDDP computes the approximation of the Bellman value functions defined in (3.44) iteratively, by running an alternation of *forward* and *backward* passes as follows. At iteration k, we suppose

given a family of value functions $\{\widetilde{V}_t^{(k)}\}_{t\in\{0,..,T\}}$. Then, SDDP refines the value functions as follows:

- *Forward pass:* Let $w^{(k)} = (w_0^{(k)}, \cdots, w_T^{(k)}) \in \mathbb{W}_0 \times \cdots \times \mathbb{W}_T$ be a noise scenario. SDDP computes a history trajectory $h^{(k)} = (h_0^{(k)}, \cdots, h_T^{(k)})$ along the scenario $w^{(k)}$ by using the cost-to-go policies corresponding to Equation (3.25). For time $t = 0, \cdots, T-1$, do

$$\gamma_t^{(k)}(h_t^{(k)}) \in \operatorname*{arg\,min}_{u_t \in \widetilde{\mathbb{U}}_t} \int_{\mathbb{W}_{t+1}} \widetilde{V}_{t+1}^{(k)}(h_t^{(k)}, u_t, w_{t+1}) \widetilde{\rho}_{t:t+1}(h_t^{(k)}, dw_{t+1}) \ . \tag{3.57}$$

  and set $h_{t+1}^{(k)} = (h_t^{(k)}, \gamma_t^{(k)}(h_t^{(k)}), w_{t+1}^{(k)})$.

- *Backward pass:* Then, SDDP refines the approximation along the history trajectory $h^{(k)}$, in a backward manner. Let $\widetilde{V}_T^{(k+1)} = K$. For time $t = T-1, \cdots, 0$, the algorithm computes a new affine function at point h_t^k, being a minorant of $\mathcal{B}_{t+1:t}^{of}(\widetilde{V}_{t+1}^{(k+1)})$ with a linearization operator $\mathfrak{S}_t$, and set

$$\widetilde{V}_t^{(k+1)} = \max\left\{\widetilde{V}_t^{(k)},\ \mathfrak{S}_t \circ \mathcal{B}_{t+1:t}^{of}(\widetilde{V}_{t+1}^{(k+1)})(h_t^k)\right\} \ . \tag{3.58}$$

We observe that at iteration k, the update of the value function (3.58) writes as a maximum of the previous value function and an affine function. Thus, the value function $\widetilde{V}_t^{(k+1)}$ remains piecewise linear, provided that the previous value function $\widetilde{V}_t^{(k)}$ was piecewise linear.

3.4.2.4. Approximate Dynamic Programming

Approximate Dynamic Programming (ADP) is an algorithm that approximates the cost-to-go by a sequence of basis functions. We refer to Bertsekas and Tsitsiklis (1996) and Powell (2007) for extensive overviews of ADP. We describe hereafter ADP in the history framework.

Approximate Dynamic Programming encodes the offline Bellman operator $\mathcal{B}_{t+1:t}^{of}$ in Equation (3.46) by a *parametrized* operator domain $\widetilde{\mathfrak{V}}_t$ (3.47) spanned by a functional basis $\phi^1, \cdots, \phi^N$:

$$\widetilde{\mathfrak{V}}_{t+1} = \operatorname{span}\{\phi_{t+1}^1, \cdots, \phi_{t+1}^N\} \ . \tag{3.59}$$

The functions $\phi_{t+1}^k : \mathbb{H}_{t+1} \to \mathbb{R}$ form a finite basis (e.g. spline or quadratic functions).

Then, cost-to-go are computed backward, in the following manner:

$$\widetilde{V}_t = \operatorname{proj}_{\widetilde{\mathfrak{V}}_t} \circ \mathcal{B}_{t+1:t}^{of}(\widetilde{V}_{t+1}) \ , \tag{3.60}$$

with, for any $\phi \in \mathbb{L}^0(\mathbb{H}_t, \mathcal{H}_t)$,

$$\operatorname{proj}_{\widetilde{\mathfrak{V}}_t}(\phi) \in \operatorname*{arg\,min}_{f \in \widetilde{\mathfrak{V}}_t} \|f - \phi\|^2 \ . \tag{3.61}$$

The projection computes parameters $\{\gamma_t^k\}_{k\in\{1,..,N\}}$ parameterizing the cost-to-go $\widetilde{V}_t$ in the operator domain $\widetilde{\mathfrak{V}}_t = \operatorname{span}\{\phi_t^1, \cdots, \phi_t^N\}$ as

$$\widetilde{V}_t(h_t) = \sum_{k=1}^{N} \gamma_t^k \phi_t^k(h_t) \ , \ \ \forall h_t \in \mathbb{H}_t \ . \tag{3.62}$$

In Equation (3.44), the backward operators $\widetilde{\mathcal{B}}_{t+1:t} : \widetilde{\mathfrak{V}}_{t+1} \to \widetilde{\mathfrak{V}}_t$ are such that

$$\widetilde{\mathcal{B}}_{t+1:t} = \mathrm{proj}_{\widetilde{\mathfrak{V}}_t} \circ \mathcal{B}^{of}_{t+1:t} \ . \tag{3.63}$$

3.4.2.5. Difference between offline algorithms

The only difference between the algorithms that compute a set of cost-to-go offline are the backward operators $\widetilde{\mathcal{B}}_t$ used. The different choices are summarized in Table 3.2.

Algo	$\widetilde{\mathcal{B}}_{t+1:t}$	
SDP	$\mathfrak{R}_{\widetilde{\mathfrak{V}}_t} \circ \mathfrak{T}_t^N \circ \mathcal{B}^{of}_{t+1:t}$	Dicretize then interpolate
SDDP	$\sup \circ\, \mathfrak{S}_t^N \circ \mathcal{B}^{of}_{t+1:t}$	Linearize then take supremum
ADP	$\mathrm{proj}_{\widetilde{\mathfrak{V}}_t} \circ \mathcal{B}^{of}_{t+1:t}$	Project directly

Table 3.2.: The approximate Bellman operators of SDP, SDDP and ADP.

Extension to the state case. Using the reduced Bellman operators $\{\overline{\mathcal{B}}_{t+1:t}\}_{t\in\{0,..,T-1\}}$ defined in Equation (3.24), we are able to adapt the different cost-to-go algorithms detailed in §3.4.2 to the state case.

By using the notation introduced in §3.2.2, the offline Bellman operators (3.46) rewrite, for any $\phi \in \mathbb{L}^0(\mathbb{X}_{t+1}, \mathcal{X}_{t+1})$, for all $x_t \in \mathbb{X}_t$, as

$$\big(\overline{\mathcal{B}}^{of}_{t+1:t}\varphi\big)(x_t) = \inf_{u_t\in\mathbb{U}_t} \int_{\mathbb{W}_{t+1}} \varphi(f_{t:t+1}(x_t, u_t, w_{t+1}))\overline{\rho}^{of}_{t:t+1}(x_t, dw_{t+1}) \ . \tag{3.64}$$

Then, by defining operator domains adapted to the state functional classes

$$\overline{\mathfrak{V}}_t \subset \mathbb{L}^0(\mathbb{X}_t, \mathcal{X}_t) \ , \ \ \forall t \in \{0, .., T\} \ , \tag{3.65}$$

we are able to adapt the different algorithms presented in §3.4.2 in a straightforward manner.

3.5. Assessment of online policies

Policies $\gamma_t : \mathbb{H}_t \to \mathbb{U}_t$ are mappings that take as argument, at each time t, the history $h_t \in \mathbb{H}_t$ to return a decision $u_t \in \mathbb{U}_t$. Depending on the ingredients chosen as explained in §3.2.3, the performances of policies differ. We detail hereafter a procedure to compare the performances of different policies.

3.5.1. Simulating the flow induced by a policy along a scenario

Let $\gamma = (\gamma_0, \cdots, \gamma_{T-1}) \in \Gamma_{0:T-1}$ be a family of history feedback policies, and $w^s = (w_0^s, \cdots, w_T^s) \in \mathbb{W}_0 \times \cdots \times \mathbb{W}_T$ be an uncertainty scenario.

The simulator computes stage by stage the value of the *flow* $\Phi^\gamma_{0:T}$ (cf Equation (3.4)) along scenario w^s:

$$\Phi^\gamma_{0:T}(w_0^s, \cdots, w_T^s) = \big(w_0^s, \gamma_0(w_0^s), w_1^s, \gamma(w_0^s, \gamma_0(w_0^s), w_1^s), \cdots, \gamma_{T-1}(h_{T-1}^s), w_T^s\big) = h_T^s \in \mathbb{H}_T \ . \tag{3.66}$$

Then, the cost of policy γ along scenario w^s is

$$j(h_T^s) = j \circ \Phi^\gamma_{0:T}(w^s) \ , \tag{3.67}$$

Algorithm 3.1: Simulation

Data: Policy γ, assessment scenario w^s
Result: Flow $\Phi^{\gamma}_{0:T}(w^s)$
$h^s_0 = w^s_0$;
for $t \in \{0, .., T-1\}$ **do**
  $u^s_t = \gamma_t(h^s_t)$;
  $h^s_{t+1} = (h^s_t, u^s_t, w^s_{t+1})$;
end
Return the flow $(w^s_0, u^s_0, w^s_1, u^s_1, \cdots, u^s_{T-1}, w^s_T)$

where $j : \mathbb{H}_T \to \mathbb{R}$ is the cost to minimize, as in Equation (3.12).

3.5.2. Comparing policies

Once the flow of a given policy γ computed, we can attach a cost to the policy γ that allows to compare it with other policies.

3.5.2.1. Assessment of a single policy

By computing the flow $\Phi^{\gamma}_{0:T}$ for a given policy γ along a *single* scenario w^s, we are able to associate a cost $j \circ \Phi^{\gamma}_{0:T}(w^s)$.

By iterating the procedure along *multiple* scenarios $w^1, \cdots, w^N$ we are able to obtain different samples $j(h^1_T), \cdots, j(h^N_T)$ usable to estimate the expected value of the policy costs. We describe the assessment procedure in Algorithm 3.2.

Algorithm 3.2: Assessment of a policy γ with N scenarios

Data: Policy γ, assessment scenarios $w^1, \cdots, w^N$
Result: Samples $\big(j(h^1_T), \cdots, j(h^N_T)\big)$
for $w^s \in (w^1, \cdots, w^N)$ **do**
  $j(h^s_T) = j \circ \Phi^{\gamma}_{0:T}(w^s)$;
end

The assessment procedure yields a cost vector $(j(h^1_T), \cdots, j(h^N_T)) \in \mathbb{R}^N$. With this, we can assess the policy γ with different metrics $\mu^N : \mathbb{R}^N \to \mathbb{R}$ — e.g. the average, the median, the variance, etc — to obtain a real value characterizing the performance of the policy:

$$\mu^N\big(j(h^1_T), \cdots, j(h^N_T)\big) \in \mathbb{R} \, .$$

Law of Large Number and Central Limit Theorem. We now consider the special case where μ^N is the sample average

$$\mu^N(x^1, \cdots, x^N) = \frac{1}{N} \sum_{i=1}^{N} x^i \, . \tag{3.68}$$

As the number of simulation N grows, we get closer and closer to the expected value of the policy's performance by using the law of large numbers. The procedure is as follows.

1. Draw a large number N of independent scenarios w^s.

2. Compute the sample cost average

$$m = \frac{1}{N}\sum_{s=0}^{N} j \circ \Phi_{0:T}^{\gamma}(w^s) \ , \tag{3.69}$$

and the sample cost standard deviation

$$\sigma^2 = \frac{1}{N-1}\sum_{s=0}^{N} (j \circ \Phi_{0:T}^{\gamma}(w^s) - m)^2 \ . \tag{3.70}$$

3. Use the Central Limit Theorem to obtain a confidence interval for $\mathbb{E}\big[j \circ \Phi_{0:T}^{\gamma}(\boldsymbol{H}_T)\big]$.

Figure 3.5 describes the whole assessment procedure applied to a given policy γ.

$$\begin{array}{ccccc} & & w^s \in \mathbb{W} & \longleftarrow & (w^1, \cdots, w^N) \in \mathbb{W}^N \\ & & \downarrow & & \vdots \\ \gamma_{0:T-1} \in \Gamma_{0:T-1} & \longrightarrow & j \circ \Phi_{0:T}^{\gamma}(w^s) \in \mathbb{R} & \longrightarrow & \mu\big((j \circ \Phi_{0:T}^{\gamma}(w^s))_{s=1,\cdots,N}\big) \in \mathbb{R} \end{array}$$

Figure 3.5.: Assessment procedure of a policy with N scenarios

3.5.2.2. Comparing different policies

We are able to assess S different policies $\gamma^1, \cdots, \gamma^S$ along the same set of scenarios $w^1, \cdots, w^N$ and compare them altogether with the same metrics μ^N. The best policy is the policy that yields the minimum cost in the vector

$$\Big(\mu^N\big[(j \circ \Phi_{0:T-1}^{\gamma^s}(w^i))_{i\in\{1,..,N\}}\big]\Big)_{s\in\{1,..,S\}} \ . \tag{3.71}$$

Usually, the decision maker uses a sample $\mathfrak{W}^{opt} = \{w_{opt}^1, \cdots, w_{opt}^M\}$ of uncertainty scenarios to design the online policies (for instance to infer the probability laws of the future uncertainties). We call the scenarios in $\mathfrak{W}^{opt}$ *optimization scenarios*, whereas the scenarios $\mathfrak{W}^{ass} = \{w^1, \cdots, w^N\}$ used during the assessment procedure in §3.5.2.1 are called *assessment scenarios*.

To ensure that the comparison of different policies remains fair, we must ensure that the set of optimization scenarios is distinct from the set of assessment scenarios. The assessment procedure is *out-of-sample* as soon as

$$\mathfrak{W}^{opt} \cap \mathfrak{W}^{ass} = \emptyset \ . \tag{3.72}$$

3.6. Discussion

We have presented in this chapter two classes of online history feedback policies: the *lookahead policies* in Sect. 3.3 and the *cost-to-go policies* in Sect. 3.4. Then, we have described in Sect. 3.5 a method to compare the performances of different history feedback policies. We sketch hereafter two other classifications — the classifications introduced in Bertsekas (2005b) and in Powell (2014) — and compare them with our own taxonomy. We note that our classification is written with a generic information structure depending on history, whereas the two other classifications rely on a state process.

A comparison with Bertsekas' classification. In Bertsekas (2005b), the author introduced a unifying suboptimal control framework, in which rollout algorithms and Model Predictive Control stand as a special case of Approximate Dynamic Programming methods. The author framed existing algorithms in two main classes as follows.

- *Explicit cost-to-go* methods compute offline a sequence of cost-to-go $\{\widetilde{V}_t\}$ and solve online a problem similar to (3.25). The author outlines two main classes of algorithms among these methods:
  - Get cost-to-go $\widetilde{V}_t$ by applying Dynamic Programming on a simpler problem. This is equivalent to define an operator $\widetilde{\mathcal{B}}_{t+1:t}$ mimicking the Bellman operators of the original problem.
  - Use a parametric approximation to approximate the cost-to-go $\widetilde{V}_t$, then tune the approximation by some systematic methods.
- *Implicit cost-to-go* methods compute online a cost-to-go by solving a problem similar to (3.26). The author distinguishes three classes of algorithms:
  - Rollout algorithm computes Problem (3.26) by using suboptimal/heuristic policies $\gamma_{t+1}, \cdots, \gamma_{\widetilde{T}-1}$ up to a given horizon $\widetilde{T}$.
  - Open-loop feedback control (OLFC) solves Problem (3.26) with open-loop policies.
  - Model Predictive Control (MPC) is a variant of OLFC where the future scenario is deterministic. MPC is equivalent to use constant stochastic kernels in Problem (3.26).

A comparison with Powell's classification. We exhibit hereafter the link existing between the cost-to-go and lookahead policies we introduced earlier and the classification introduced by Powell (2014).

Powell stated that there exists four classes of algorithms to design online policies. We frame these four classes as cost-to-go methods or lookahead methods.

1. The *policy functions approximation (PFA)* directly parameterizes the online policy γ_t with a fixed rule or a parametric model depending on history. For instance, γ_t may encode a PI discrete controller or a (s,S) policy.

2. The *cost function approximation (CFA)* minimizes a parametrized cost as follows

$$\gamma_t(h_t) \in \underset{u_t \in \widetilde{\mathbb{U}}_t}{\arg\min} \int_{\mathbb{W}_{t+1}} \big(\sum_{f \in \widetilde{\mathfrak{Y}}} \gamma^f \Phi_t^f(h_t, u_t, w_{t+1}) \big) \rho_{t+1}(h_t, dw_{t+1}) \,, \tag{3.73}$$

   with $\{\Phi^f\}_{f \in \widetilde{\mathfrak{Y}}}$ a set of basis functions. That corresponds to the Approximate Dynamic Programming cost-to-go policies, which use a parametric approximation of the cost-to-go $\widetilde{V}_{t+1} : \mathbb{H}_{t+1} \to \mathbb{R}$, as described in §3.4.2.4.

3. The *value function approximation (VFA)* approximates the cost-to-go $\widetilde{V}_{t+1}$ and defines the corresponding online policies as

$$\gamma_t(h_t) \in \underset{u_t \in \widetilde{\mathbb{U}}_t}{\arg\min} \int_{\mathbb{W}_{t+1}} \widetilde{V}_{t+1}(h_{t+1}) \rho_{t+1}(h_t, dw_{t+1}) \,. \tag{3.74}$$

   This is exactly the template of cost-to-go policies introduced in Sect. 3.4.

4. The *lookahead policies* optimize a multistage problem over a given horizon $\widetilde{T}$, yielding exactly the template of the lookahead policies described in Sect. 3.3.

We recover Powell's framework, with the exception of the *policy function approximations* scheme.

Powell described also different ingredients to frame online policies. We hereby establish a link between his ingredients and the parameters introduced earlier in Sect. 3.2.3:

- *Limiting the horizon:* that is equivalent to set a horizon $\widetilde{T} \leq T$.
- *State aggregation:* that equate to choose properly a reduction mapping θ_t, as in §3.2.2.
- *Outcome aggregation or sampling:* that corresponds to a given parameterization of the stochastic kernels $\widetilde{\rho}_{t+1}(h_t, \cdot)$.

Chapter 4.

Background on the modelling of energy flows and stocks in microgrids

Contents

4.1. Introduction

A microgrid is a local energy network that produces part of its energy and controls its own demand. Because of the different stocks and interconnections, such systems are often complex to control. Furthermore, at local scale, electrical demands and weather conditions (heat demand and renewable energy production) are highly variable and hard to predict; their stochastic nature adds uncertainty to the system.

We distinguish the *electrical system* from the *thermal system*. Depending on the microgrid's configuration, these two systems can be independent or coupled. The electrical system assembles the different devices using electricity as energy source, whereas the thermal system combines the devices using heat as energy. In France, the heating consumption account for up to 63% of the energy consumption in the residential sector (ADEME (2017)) Thus, the choice of the heating technology (electrical or thermal heaters) deeply impacts the costs structure in microgrids.

In this chapter, we depict the physical equations governing local microgrids. These equations are naturally written in continuous time. Such physical models will be at the heart of the decision process used by the Energy Management System (EMS). Uncertainties are viewed as exogenous variables impacting the different elements of the microgrid. We make use of statistical methods to model such uncertainties as random variables in the optimization model.

This chapter is organized as follows. First, we describe the uncertainties w we are facing in §4.2, then the different controls u we can act upon to control microgrids in §4.3, and eventually we detail the different stocks x in the microgrids in §4.4.

4.2. Modelling uncertainties

Local microgrids are naturally confronted by two kind of uncertainties. First, the weather impacts the microgrids with *outer uncertainties*: wind and solar radiation impact the production of local renewable productions, outdoor temperature affects the inner temperatures in buildings. Second, the energy demands are *inner uncertainties*, depending on the behavior of the different inhabitants. The outer and inner uncertainties do not have the same nature, and rely on different statistical models.

We first begin to describe the outer uncertainties, and then focus on inner uncertainties.

4.2.1. Weather conditions as outer uncertainties

We view weather conditions as outer uncertainties. Generally, we rely on weather forecast to predict the future evolution of the outdoor temperature and the solar irradiation. Most forecasts give the evolution of weather as deterministic values. However, probabilistic forecasts are gaining interest as they give a more complete representation of the uncertainties (Morales et al., 2013).

4.2.1.1. Solar radiation

The solar radiation impacts different systems inside the microgrid: it heats the buildings and it can be recovered by a solar panel to produce energy.

The solar radiation depends on the date (radiation is higher during summer than during winter), the hour in the day, the geoposition of the microgrid (high latitudes receive less radiation than low latitudes) and the nebulosity (the cloud coverage).

Apart from nebulosity, all parameters are deterministic. However, the cloud coverage is hardly predictable, especially during windy days. That is why we choose to model the solar radiations as an outer uncertainty.

The global horizontal irradiation is function of two kinds of radiations: the direct radiation Φ^b (b for *beam*) and the diffuse radiation Φ^d (d for *diffuse*). The direct radiation arrives to the surface considered and depends on the position of the sun in the sky, whereas the diffuse radiation results from the diffusion of the sun lightning through the sky. The more cloudy is the weather, the higher the diffusion. We refer to Annex 4.5.3 for a broader description of the computation of solar irradiation.

Figure 4.1 displays the evolution of temperature and solar radiation for one week in summer.

4.2.1.2. Outdoor temperature

With the thermal losses through windows and walls, the outdoor — or outer — temperature θ^e affects the temperatures of the different buildings. If we consider short time horizons (less than two days) the temperature's forecasts are relatively accurate enough, and the buildings' thermal inertia limits the impact of forecasts' errors. Thus, we do not consider the outdoor temperature as an uncertainty, but rather as an exogeneous parameter with known value.

4.2.2. Energy demands as inner uncertainties

Energy demands depend mainly on the occupancy of the inhabitants. On the contrary to outer uncertainties, we do not have forecasts available to predict the future demands.

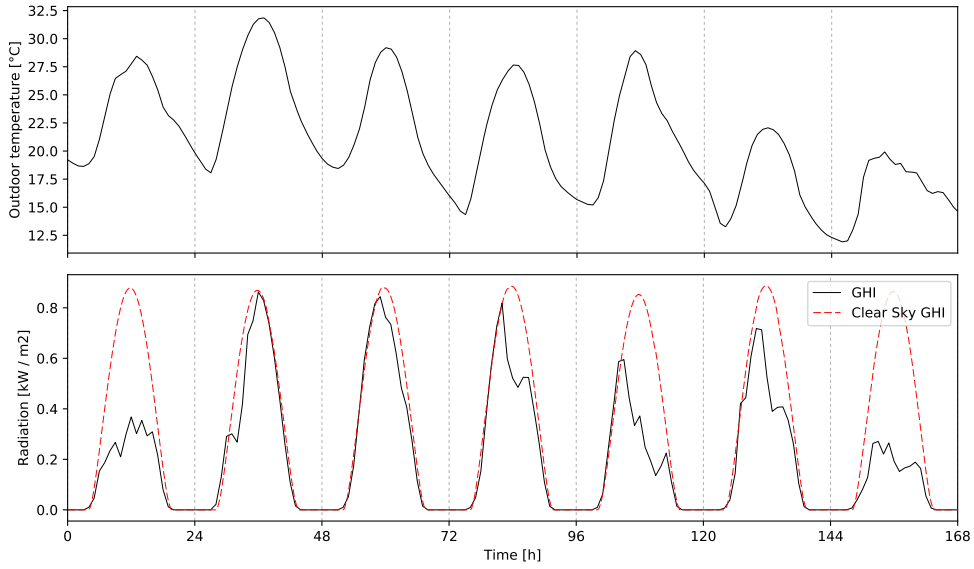


Figure 4.1.: Evolution of outdoor temperature and solar radiation between July 20th and July 27th, 2015 in Orly, France.

4.2.2.1. Modelling occupancy and demands with Markov Chain

In local microgrid, the two main energy demands are the *electrical demand* (gathering the lightning load, the consumption of the different electronic devices and other home appliances) and the *domestic hot water demand* (DHW) (the consumption of hot water in the bathroom or the washing machine). These demands depend on the number of inhabitants inside the building.

In Page et al. (2008), the authors present a probabilistic modelling of the occupancy N^{occ} via an inhomogeneous Markov chain, whose probability weights are estimated from real data. In Baetens and Saelens (2016), the authors extend this procedure to generate electrical and thermal loads from an occupancy Markov chain. Their Markov chains are calibrated with data issued from Belgium buildings. They developed an open-source library, `StRoBe` [1], to generate demand scenarios in a flexible manner.

We use the library `StRoBe` to generate scenarios for electrical and thermal loads. We modify the library to take as input the number of inhabitants and the sociological profile, the number of scenarios to generate, and the number of days to consider.

Figure 4.2 displays 10 scenarios of demands during one week, sampled at a 15mn time step. We observe peaks in electrical demands during mornings and evenings, and lower demands during night. The change of pattern during the fifth and sixth days is due to the week-end.

4.2.2.2. Fitting probability laws on demands scenarios

We now suppose given a set of N demand scenarios, corresponding to previous history of demands. We aim at fitting a statistical model to these scenarios.

Identifying marginal laws. We first present a model that disregards the time correlation between the different time steps and identifies discrete marginal probability distributions. At a given moment, we suppose available N samples $w^1, \cdots, w^N$.

[1] Available at `https://github.com/open-ideas/StROBe`

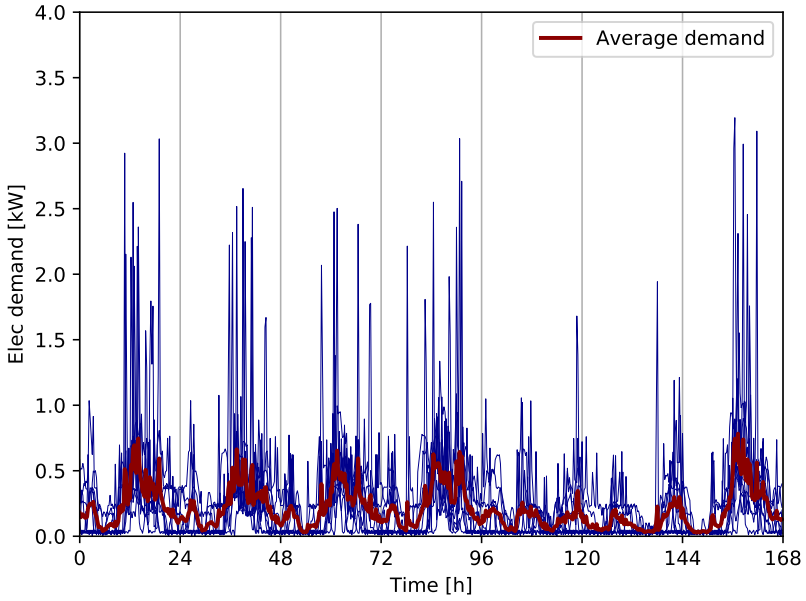


Figure 4.2.: Displaying 10 scenarios of electrical demand for one week, at 15mn time step

A first model looks at the discrete distribution corresponding to the N samples:

$$\mu(\cdot) = \frac{1}{N}\sum_{s=1}^{N} \delta_{w^s}(\cdot)\ , \tag{4.1}$$

with δ_{w^s} the Dirac measure centered at w^s. However, that could lead to large size distribution as N is potentially large.

We use quantization algorithms to reduce the size N of the discrete probability distribution (4.1). Let $1 \leq S \leq N$ be a quantization size. The quantization algorithm finds a partition $\Xi = (\Xi_1, \cdots, \Xi_S)$ of the N samples $(w^1, \cdots, w^N)$ as solution of the optimal quantization problem

$$\min_{\Xi} \quad \sum_{s=1}^{S} \left(\sum_{w^i \in \Xi_s} \left\| w^i - \widetilde{w}^s \right\|_2^2 \right)\ , \tag{4.2}$$

where $\widetilde{w}^s = \frac{1}{card(\Xi_s)} \sum_{w^i \in \Xi_s} w^i$ is the *centroid* of Ξ_s.

Problem (4.2) is NP-hard. We use as heuristic the Lloyd-Max quantization algorithm (Lloyd, 1982) (or *k-means algorithm*) to solve (4.2). This procedure may be viewed as a scenario reduction methods, as explained in Rujeerapaiboon et al. (2017).

Once the optimal partition obtained, we define the quantized distribution

$$\mu^q(\cdot) = \sum_{s=1}^{S} p_s \delta_{\widetilde{w}^s}(\cdot)\ , \tag{4.3}$$

where $\delta_{\widetilde{w}^s}$ is the Dirac measure at point $\widetilde{w}^s$ and $p_s = card(\Xi_s)/N$ is the associated probability weight.

Identifying a Markov Chain. Another approach would consist in fitting a Markov Chain to the optimization scenarios, so as to catch the time interdependence between to consecutive time steps. We refer to Bally et al. (2003) for a description of the procedure and to Löhndorf and Shapiro (2017) for an application to multistage stochastic programming.

4.3. Modelling production

We describe now the different devices used to produce energy inside microgrids. As we control these devices, their production will be part of the decision vector u.

4.3.1. Combined heat and power generator

A Combined Heat and Power Generator (CHP) is a generator that produces electricity with a thermal engine and recovers the heat produced during the process. Small-scale CHP, called micro-CHP or μCHP, are small generators suitable for use in local microgrids. The main technologies for μCHP are internal combustion engines, Stirling engines, and fuel cell technology (see Thomas (2008)). We refer to Bonabe de Rougé (2018) and to Parisio et al. (2015) for a description of the physical modelling of μCHP and the calibration of the parameters with experimental data. Here, we consider Stirling engine μCHP.

Let p^{gas} be the thermal power of the gas injected into the μCHP, and η_{chp} its yield. The power generated by the μCHP at a given time t is

$$p^{\text{chp}}(t) = \eta_{\text{chp}} \times p^{\text{gas}} \times y(t) \ , \tag{4.4a}$$

where $y(t) \in \{0, 1\}$ denotes if the μCHP is either ON or OFF. The gas is burnt to run the external combustion engine, and we produce an amount $f^{\text{ge}}(t)$ of electricity and $f^{\text{gh}}(t)$ of heat. The energy balance of the μCHP writes:

$$f^{\text{ge}}(t) + f^{\text{gh}}(t) = p^{\text{chp}}(t) \ . \tag{4.4b}$$

The Stirling technology does not allow to modulate the share of energy produced by the generator. If we denote $\gamma \in [0, 1]$ this share, Equation (4.4b) now becomes

$$f^{\text{ge}}(t) = \gamma \times p^{\text{chp}}(t) \ , \tag{4.4c}$$

$$f^{\text{gh}}(t) = (1 - \gamma) \times p^{\text{chp}}(t) \ . \tag{4.4d}$$

4.3.2. Solar panel

A photovoltaic system gathers photovoltaic solar cells that transform light into electricity. Let p^{pv} be the energy production of such panels; it depends linearly on the solar radiation Φ^{pv} received by the solar panel

$$p^{\text{pv}}(t) = \eta_s \times A_s \times \Phi^{\text{pv}}(t) \ , \tag{4.5}$$

where η_s is the yield of the solar panel (between 5% and 20%), A_s its surface in m^2.

The solar radiation Φ_t^{pv} is function of the direct and diffuse horizontal radiation

$$\Phi^{\text{pv}}(t) = g_{\text{pv}}\big(\Phi^b(t), \Phi^d(t)\big) \ , \tag{4.6}$$

with g_{pv} a function depending on the inclination and the orientation of the solar panel (see Annex 4.5.3). Thus, it is also greatly impacted by varying nebulosity.

Remark 4.3.1. *Some solar panel technologies allow to modulate the power generated by the solar panel. In this case, the EMS is able to control the power output of the solar panel within a given*

range:

$$\Phi^{pv}(t) = \eta_s \times A_s \times \Phi^{pv}(t) \times u^m(t) \ , \tag{4.7}$$

with $u^m(t) \in [\underline{u}^m, \overline{u}^m] \subset [0,1]$ *the modulation control.* ♢

4.3.3. Thermal boiler

The thermal boiler burns gas to generate heat. We model the thermal boiler with a balance equation stating that the power of the thermal boiler is proportional to the power of gas burnt by the boiler:

$$p^{\text{burn}}(t) = \eta^{\text{burn}} \times p^{\text{gas}}(t) \ , \tag{4.8}$$

with $p^{\text{gas}}(t) \in [0, \overline{p}^{\text{gas}}]$ an adjustable power and η^{burn} a given yield, that we suppose independent of the outer temperature and pressure.

4.4. Modelling storage

After uncertainties and production, we now describe the different stocks used to store energy in local microgrids. The storage's dynamics introduce a time coupling. The stocks are gathered inside a state x.

4.4.1. Electrical battery

There exists different technologies to store energy: water storage, flywheels, chemical batteries. Here, we consider only electro-chemical batteries. Two technologies dominate the market: lead-acid batteries and lithium-ion batteries. The former is widely used in industrial context, where the size of the battery is of importance. The later has gathered a lot of attention recently with the democratization of energy storage in electronic appliances and with the rise of interest in electrical cars and domestic batteries. We consider here only lithium-ion batteries.

Dynamics. The evolution of the energy stored in the battery $b(t)$ is modeled with a stock dynamic:

$$\frac{db}{dt} = \rho_c (f^{\text{b}}(t))^+ - \frac{1}{\rho_d} (f^{\text{b}}(t))^- \ , \tag{4.9}$$

with ρ_c and ρ_d being the charge and discharge efficiency and $f^{\text{b}}(t)$ denoting the power exchange with the battery. We use the convention $f^+ = \max(0, f)$ and $f^- = \max(0, -f)$.

Constraints. A battery has a maximal allowable charge that depends upon the technology and the age of the battery. With lithium-ion battery, it is recommended to ensure that the energy stored inside the battery ranges between 30% and 90% of the maximal nominal charge. We write this constraint as:

$$\underline{b} \leq b(t) \leq \overline{b} \ . \tag{4.10a}$$

As we cannot withdraw an infinite power from the battery at time t, we bound the power exchanged with the battery:

$$-\underline{f}^{\text{b}} \leq f^{\text{b}}(t) \leq \overline{f}^{\text{b}} \ . \tag{4.10b}$$

Remark 4.4.1. *Some models consider battery aging, as the nominal charge of a chemical battery decreases with the number of cycles. We refer to Haessig et al. (2015) for a broader introduction to that subject, which is not treated here.* ♢

4.4.2. Hot water tank

The thermal system uses a hot water tank to store heat. We use here a simplified modelling by considering that the temperature of the hot water inside the tank is homogeneous.

Dynamic. We use a simple linear model for the hot water tank. At time t, we denote by $\theta^h(t)$ the temperature inside the hot water tank. We suppose that this temperature is homogeneous, that is, no stratification occurs inside the tank.

At time t, we define the energy $h(t)$ stored inside the tank as the difference between the tank's temperature $\theta^h(t)$ and a reference temperature θ^{ref}

$$h(t) = \rho V_h c_p \big(\theta^h(t) - \theta^{ref}\big) \ , \tag{4.11}$$

where V_h is the tank's volume, c_p the calorific capacity of water and ρ the density of water. The energy $h(t)$ is bounded:

$$0 \leq h(t) \leq \overline{h} \ . \tag{4.12}$$

The enthalpy balance equation is written

$$\frac{dh}{dt} = q^{\text{in}}(t) - q^{\text{out}}(t) - k^h A^h \big(\theta^h(t) - \theta^{env}\big) \ , \tag{4.13}$$

where $q^{\text{in}}(t)$ is the power injected inside the tank, $q^{\text{out}}(t)$ the output power to satisfy the different heating needs, and the term $k^h A^h(\theta^h(t) - \theta^{env})$ stands for the heat losses due to conduction through the hot water tank surface A^h, with k^h a convection coefficient and θ^{env} the temperature of the tank environment.

A more accurate representation would model the stratification inside the hot water tank. However, this would greatly increase the number of states in the system, rendering the numerical resolution more cumbersome. We refer to Schütz et al. (2015) and Beeker et al. (2016) for discussions about the impact of the tank's modelling on the performance of the control algorithms. The first approach uses a finite layers model to model the stratification of temperatures inside the tank, at the expense of having as many state variables as layers. The second approach models electrical hot water tank as a mix between a hot fluid and a cold fluid. This model considers three state variables: the temperature of the hot water, the temperature of the cold water and the position of the interface between these two fluids (the hot water is always above cold water). However, such model introduces a Boolean variables to avoid simultaneous thermal exchanges between the hot fluid and the cold fluid. That is why we decide to simplify the model and use a single state variable to characterize the energy stored inside the tank. This is equivalent to consider that temperatures are homogeneous inside the tank.

4.4.3. Thermal envelope

Heaters ensure that the indoor temperature remains greater than a given set-point. The power injected through the heaters heats the rooms and compensates the energy losses that occur through the walls and windows. There exist two main methods to model the thermal dynamics of a house.

- *Electrical analogy* methods assimilate the thermal system to an electrical circuit: one views temperatures as voltages, thermal flows as currents thermal stocks and walls as capacitors and resistances. We refer to Berthou (2013) for an application of such models to the control of buildings.

- *Reduction methods* (singular uncertainties, identification methods, balanced truncation) identifies a linear model on experimental data. The use of this kind of methods in optimal control is studied in Malisani (2012).

To compute efficiently the heating needs, we introduce a model of the house's thermal envelope based upon an electrical analogy (see Figure 4.3 for an illustration). Berthou (2013) shows that a model with 6 resistances and 2 capacitors (also called R6C2 model) is a good compromise between complexity and accuracy for such a modelling. Such model considers two state variables at each time t:

- the inner temperature $\theta^{\mathrm{i}}(t)$,
- the walls' temperature $\theta^{\mathrm{w}}(t)$,

and it assumes that the inner (resp. the wall) temperature is homogeneous inside the building (resp. the walls).

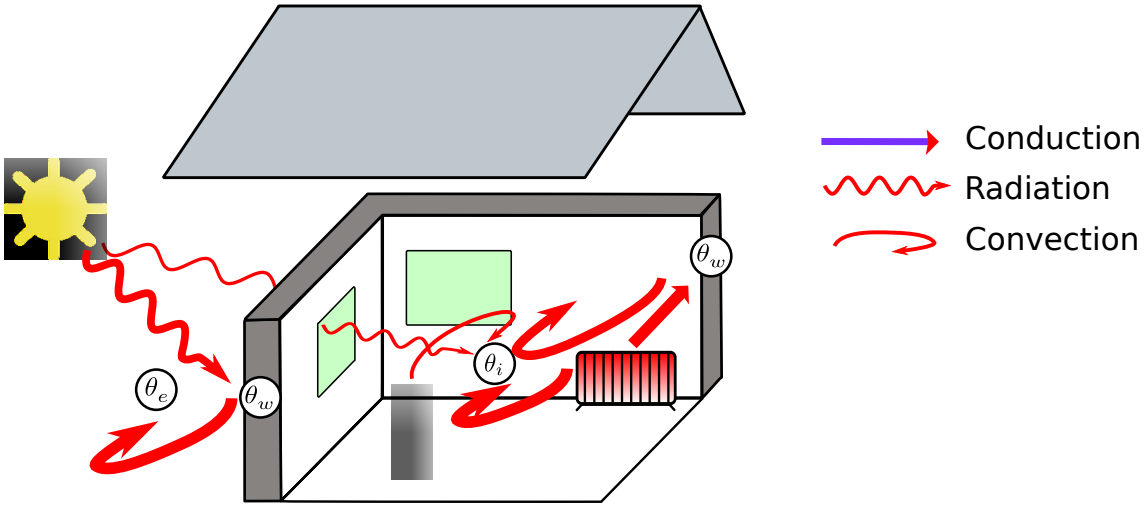


Figure 4.3.: Different energy exchanges inside a building

R_i	inner convection resistance
R_s	resistance of isolation
R_m	wall's resistance
R_e	external convection resistance
R_v	ventilation's resistance
R_w	windows' resistance
C_i	inner capacitance
C_w	wall's capacitance

Table 4.1.: Thermal resistances and capacitors

We denote by f^h the thermal flow injected into the heaters. A proportion γ of this power is dissipated through the wall by conduction, and a proportion $1-\gamma$ is dissipated by convection in the main room. We give in Equation (4.14) the continuous equations of the R6C2 model corresponding

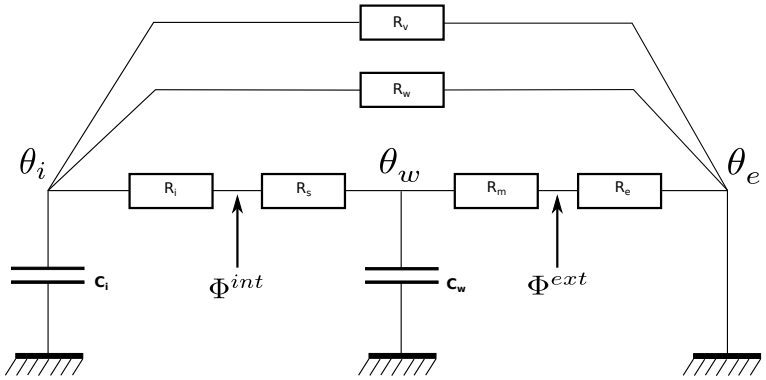


Figure 4.4.: Modelling of the thermal behavior of a house with a R6C2 analogy

to Figure 4.4:

$$C_w \frac{d\theta^{\mathrm{w}}}{dt} = \underbrace{\frac{\theta^{\mathrm{i}}(t) - \theta^{\mathrm{w}}(t)}{R_i + R_s}}_{\substack{\text{Exchange} \\ \text{Indoor/Wall}}} + \underbrace{\frac{\theta^{e}(t) - \theta^{\mathrm{w}}(t)}{R_m + R_e}}_{\substack{\text{Exchange} \\ \text{Outdoor/Wall}}} + \underbrace{\gamma f^h(t)}_{\text{Heater}} + \underbrace{\frac{R_i}{R_i + R_s}\Phi^{\mathrm{int}}(t)}_{\substack{\text{Radiation} \\ \text{through windows}}} + \underbrace{\frac{R_e}{R_e + R_m}\Phi^{\mathrm{ext}}(t)}_{\substack{\text{Radiation} \\ \text{through wall}}}, \quad (4.14a)$$

$$C_i \frac{d\theta^{\mathrm{i}}}{dt} = \underbrace{\frac{\theta^{\mathrm{w}}(t) - \theta^{\mathrm{i}}(t)}{R_i + R_s}}_{\substack{\text{Exchange} \\ \text{Indoor/Wall}}} + \underbrace{\frac{\theta^{e}(t) - \theta^{\mathrm{i}}(t)}{R_v}}_{\text{Ventilation}} + \underbrace{\frac{\theta^{e}(t) - \theta^{\mathrm{i}}(t)}{R_w}}_{\text{Windows}} + \underbrace{(1-\gamma) f^h(t)}_{\text{Heater}} + \underbrace{\frac{R_s}{R_i + R_s}\Phi^{\mathrm{int}}(t)}_{\substack{\text{Radiation} \\ \text{through windows}}}. \quad (4.14b)$$

The parameters used in these equations are given in Table 4.1. Φ^{int} and Φ^{ext} account for the solar heating through the windows and the walls, and depend on the direct and diffuse radiations:

$$\begin{cases} \Phi^{\mathrm{int}}(t) = g_b^{\mathrm{int}}\big(\Phi^{\mathrm{b}}(t), \Phi^{d}(t)\big) \\ \Phi^{\mathrm{ext}}(t) = g_b^{\mathrm{ext}}\big(\Phi^{\mathrm{b}}(t), \Phi^{d}(t)\big) \end{cases} \quad (4.15)$$

where g_b^{int} and g_b^{ext} are two functions depending on the size of the building, its orientation and the proportion of windows (see Annex 4.5.3).

Remark 4.4.2. *The drawback of such models is that the coefficients in* (4.14) *must be calibrated to correspond to the description of the thermal envelope of a house. Furthermore, these coefficients have to be recalibrated to account for the evolution of the thermal envelope (for instance, the model is sensitive to weather's wetness; a wet wall does not have the same behavior as a dry wall). We refer to Berthou (2013) for an analysis of models calibration.* ◊

4.5. Discussion

We described in this chapter the physical and uncertainties models used to described local microgrids. We write the physical equations in continuous time. In Chapter 5 and Chapter 6, we will use these models to design optimization models written in discrete time.

In Chapter 5, we present a first case study. The system is a domestic microgrid, equipped with a micro combined heat and power generator (μCHP). We compare the performance of optimization algorithms with existing approaches and show the benefit of optimization methods to control such

systems. Then, we compare two different optimization algorithms: the so-called model predictive control (MPC) and a stochastic optimization algorithm based upon SDDP.

In Chapter 6, we focus on a second case study. The system produces its own energy with solar panels rather than a μCHP. This chapter encompasses a comparison of the two stochastic optimization algorithms we introduced earlier in Chapter 3, and quantifies the sensitivity of these two algorithms w.r.t. the level of uncertainty.

Appendix

We detail in this section the different numerical values used in the models previously introduced. We thank Romain Bonabe de Rougé for his help concerning the calibration of models.

4.5.1. R6C2 model

We give in Table 4.2 the parameters of the different R6C2 model. RT12, RT05 and RT88 are French specification for buildings.

	RT 12	RT 05	RT 88
U_{wall}	0.130	0.280	0.821
U_{window}	1.226	2.040	2.285
C_i	2953500	2587720	5852180
C_w	77488500	88875200	82980800
G_s	3685	4690	3393
F_v	0.553	0.78	0.585

Table 4.2.: RT specifications for a single house

4.5.2. Specification of Stirling engine

We give in Table 4.3 the parameters corresponding to a Stirling CHP engine.

	Numerical values
P_{chp}	7 kW
η_{chp}	0.93
γ	0.13

Table 4.3.: Specification of Stirling engine

4.5.3. Models of solar irradiance

We now detail the irradiance model used to model the production of the solar panels and the solar heating inside the buildings.

The solar irradiance depends on the position of the sun in the sky, the current date, the atmospheric opacity (depending on pollution, humidity, etc.) and the nebulosity (depending on cloud cover).

Notation. We use in the next sections the notations introduced in Table 4.5.3.

Φ^g	GHI	Global Horizontal Irradiation
Φ^b	BHI	Beam (Direct) Horizontal Irradiation
Φ^d	DHI	Diffuse Horizontal Irradiation
$\overline{\Phi}^g$	GTI	Global Tilted Irradiation
$\overline{\Phi}^b$	BTI	Beam (Direct) Tilted Irradiation
$\overline{\Phi}^d$	DTI	Diffuse Tilted Irradiation

4.5.3.1. Beam horizontal irradiation

Let Φ^{atm} be the solar radiation at top of atmosphere. It depends on the distance between Earth and the sun, and is equal to

$$\Phi^{atm}(t) = G_{sc}\Big[1 + 0.0334\cos\Big(2\pi\frac{(n-4)}{T_{earth}}\Big)\Big] \ , \tag{4.16}$$

where G_{sc} is the solar constant, corresponding to the average solar irradiation at the top of the atmosphere — whose value is given by the black body irradiation of the sun — $T_{earth} \approx 365.25$ day is the duration of the year, n the day considered.

We recall that the *aphelion* arises around July 5th, and the *perihelion* between January 3rd and January 4th.

At the surface of the earth, the direct horizontal irradiation Φ^b is an attenuation of the solar radiation at top of atmosphere Φ^{atm}. The Kasten model gives a relation between these via the Beer-Lambert law:

$$\Phi^b(t) = \Phi^{atm}(t)\exp\big(-0.8662 \times T_L \times m_A \times \tau_R\big) \ , \tag{4.17}$$

where m_A is the atmospheric mass, τ_R the optical depth of the atmosphere and T_L the Linke turbidity factor. We detail hereafter each of these parameters.

Atmospheric mass. The atmospheric mass m_A is the ratio between the air mass crossed by the light beam and the air mass crossed by a light beam coming from the zenith down to a point at sea level. This value depends on the beam's orientation and the altitude of the current point. Kasten and Young (1989) give the relation:

$$m_A = \frac{p_{atm}}{p_0} \times \frac{1}{\cos\theta_z + K_0(\theta_0 - \theta_z)^{-\alpha}} \ , \tag{4.18}$$

where

- p_{atm} is the atmospheric pressure,
- p_0 is the reference pressure,
- θ_z is the zenith angle,
- $K_0 = 0.50572$,
- $\theta_0 = 96.07995°$,
- $\alpha = 1.6364$.

Optical depth. The optical depth characterizes the transparency degree of the layer crossed by the light beam. The higher the diffusion in the layer, the higher the optical depth. The optical depth τ_R satisfies:

$$d\tau_R = -\kappa\rho dz \ , \tag{4.19}$$

where κ is the opacity of the atmosphere at point z, and ρ the air density. If we denote by $\mathcal{L}$ the trajectory followed by the beam, we have:

$$\tau_R = \int_{\mathcal{L}} \kappa\rho dz \ . \tag{4.20}$$

Linke turbidity factor. This factor corresponds to the number of ideal atmospheres that would be crossed by the light beam to obtain the same attenuation as in the realistic atmosphere studied. This factor is given, and depends on the weather condition (see Table 4.4).

T_L	Type
2	dry and cold atmosphere
3	dry and warm atmosphere
4-6	wet and warm atmosphere
7	polluted atmosphere

Table 4.4.: Linke turbidity factor

4.5.3.2. Global horizontal irradiation

The clear-sky horizontal radiation Φ^g depends on the direct solar irradiation and the diffuse solar irradation:

$$\Phi^g = \Phi^b \cos\theta_z + \Phi^d \ , \tag{4.21}$$

with

- Φ^b the direct solar irradiation (BHI);
- Φ^d the diffuse solar irradiation (DHI) corresponding to the diffusion of light through atmosphere (it is consider isotropic if the sky is cloudy). The value of Φ^d is given by empirical relations (Noorian et al., 2008);
- θ_z is the angle between the sun direction and the normal vector of the horizontal surface.

Zenith angle. Let δ be Earth declination w.r.t. the ecliptic:

$$\delta = \delta_0 \times \sin\left(2\pi \frac{284+n}{365}\right) \ , \tag{4.22}$$

with $\delta_0 = 23.45°$.

We note ω the hour angle (between $-\pi$ and π) and ϕ the latitude. The zenith angle is given by the trigonometric relation:

$$\cos\theta_z = \cos\phi\cos\delta\cos\omega + \sin\phi\sin\delta \ . \tag{4.23}$$

4.5.3.3. Cloud cover solar radiation

The cloud cover solar radiation is a correction of the clear-sky solar radiation, taking into account the attenuation of light through the cloud.

The total irradiation Φ^{tot} is given by

$$\Phi^{tot} = \kappa \times \Phi^g \ , \tag{4.24}$$

where κ is the *cloud attenuation factor*.

Kasten gives a model to compute κ as function of the nebulosity N:

$$\kappa = (1 - \frac{3}{4}\left(\frac{N}{8}\right)^{3.4}) \ . \tag{4.25}$$

N is the proportion of the sky covered by clouds, between 0 and 8.

Other models (Bacher et al., 2009) rely on statistical model to estimate the attenuation κ.

4.5.3.4. Global tilted solar radiation

We now consider a tilted surface, with an angle β w.r.t. the horizontal. The global tilted solar radiation is:

$$\overline{\Phi}^g = \overline{\Phi}^b + \overline{\Phi}^d + \overline{\Phi}^r \ , \tag{4.26}$$

with $\overline{\Phi}^b$ the beam titled irradiation, $\overline{\Phi}^d$ the diffuse tilted irradiation and $\overline{\Phi}^r$ the reflected tilted irradiation.

4.5.3.5. Beam tilted irradiation

The beam tilted irradiation $\overline{\Phi}^b$ satisfies

$$\overline{\Phi}^b = \Phi^b \cos\theta \ , \tag{4.27}$$

with Φ^b the beam horizontal irradiation and θ the angle between the normal of the solar panel and the direction of the sun in the sky, satisfying:

$$\cos\theta = \cos\theta_z \cos\beta + \sin\theta_z \sin\beta \cos\left(\gamma_s - \gamma\right) \ , \tag{4.28}$$

with γ is the azimuth of the panel (that is, the angle w.r.t. the south).

4.5.3.6. Diffuse tilted irradiation

$\overline{\Phi}^d$ is the diffuse titled irradiation. Different models exist to compute $\overline{\Phi}^d$. We detail hereafter the Liu&Jordan model and the Perez model.

Liu&Jordan model. If the diffuse irradation is isotropic, the diffuse tilted irradiation is function of the diffuse horizontal irradiation:

$$\overline{\Phi}^d = \Phi^d \times \frac{1+\cos\beta}{2} \ . \tag{4.29}$$

Perez model. Perez model is considered as the reference model to compute the diffuse tilted radiation (Noorian et al., 2008). It writes

$$\overline{\Phi}^d = \Phi^d \Big[(1-F_1)\frac{1+\cos\beta}{2} + F1\frac{a}{b} + F_2 \sin\beta \Big] \ . \tag{4.30}$$

where F_1 is the coefficient of circumsolar irradiation, F_2 is the horizontal illumination coefficient, and a and b take into account the incidence angle of the sun:

$$\begin{cases} a = \max\left(0,\ \cos\theta\right) \\ b = \max\left(\cos 85^\circ,\ \cos\theta_z\right) \ . \end{cases} \tag{4.31}$$

4.5.3.7. Reflected radiation

$\overline{\Phi}^r$ is the irradiation reflected by the soil and depends on the albedo α_s of the surrounding surface:

$$\overline{\Phi}^r = \alpha_s \times \Phi^g \times \frac{1-\cos\beta}{2} \ . \tag{4.32}$$

Chapter 5.

Optimal management of a home microgrid with a CHP

Contents

5.1. Introduction

This chapter details a first case study, where we compare different methods to control the Energy Management System (EMS) of a domestic microgrid. We consider a building equipped with a micro-Combined Heat and Power generator (μCHP) and a battery. The heat produced by the μCHP is stored inside a hot water tank. Then, the heat is dispatched between thermal heaters — to heat the house — and the demand of domestic hot water. An auxiliary burner produces heat when the production of the μCHP is not sufficient to fulfill the thermal heating demand. The microgrid is connected to an distribution grid to import electricity when necessary.

We consider a horizon of one year. We suppose that the EMS updates its decisions every 15 minutes, yielding 35,040 time steps. The EMS views domestic hot water and electrical demands as uncertainties, whose future realizations are unknown. We aim at minimizing the operational cost of the microgrid.

The reference method to tackle uncertainties in microgrid is the well-known Model Predictive Control (MPC) algorithm. We first exhibit numerically that MPC gives better results than a (s, S) policy to manage stocks inside the microgrid. Then, we compare MPC with another algorithm, Stochastic Dual Dynamic Programming (SDDP). We show that MPC is almost as good as SDDP in terms of the out-of-sample performance.

The insertion of μCHP in local microgrids has been widely studied in the literature. We refer to Houwing et al. (2011) for an application of MPC to the control of a μCHP to improve the

μCHP's flexibility. An extension to the stochastic case was studied in Mohammadi et al. (2014) and in Liu et al. (2014), where stochastic programming based methods were applied to the control of microgrids equipped with batteries and μCHP. However, to the best of our knowledge, handling uncertainties with Dynamic Programming based methods such as SDDP has not been studied in the literature.

The author thanks Romain Bonabe de Rougé for his fruitful help concerning the modeling of the problem. We refer to Bonabe de Rougé (2018) for a broader description of the model and data used in this chapter.

The structure of this chapter is as follows. In §5.2 we describe the modeling of the system. Then, we present the resolution methods used to compute the optimal policies in §5.3. Eventually, we give in §5.4 numerical results, and we compare the different algorithms.

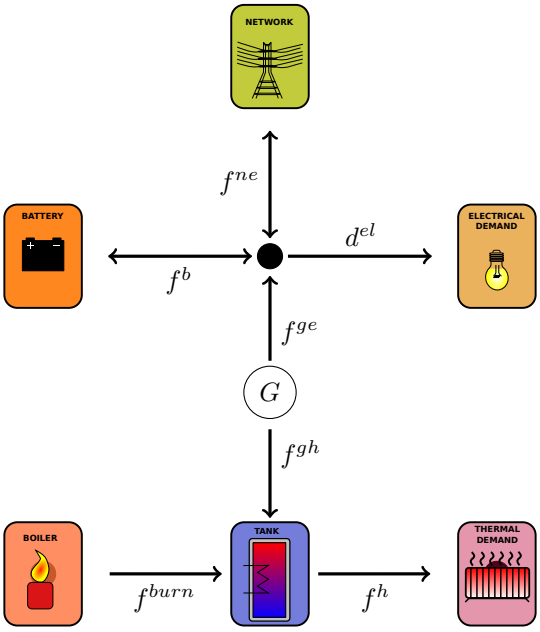


Figure 5.1.: The μCHP G connects together the thermal and electrical subsystems. The electrical system is equipped with a battery. The thermal system stores the heat produced by the μCHP with a hot water tank, and satisfies the heating and domestic hot water demands.

5.2. Problem statement

We describe in this section the optimization model corresponding to the local microgrid. We first write the electrical and thermal load balances in continuous time in §5.2.1 and then discretize the equations in §5.2.2.

5.2.1. Electrical and thermal load balance equations

The configuration of the microgrid is detailed in Figure 5.1. The domestic microgrid is equipped with a μCHP, denoted G, a battery, denoted B, and a hot water tank, denoted H. We consider also the house's thermal inertia. The continuous dynamics of these devices were described in §4.4. The balance equations of the electrical and the thermal systems are as follows.

Electrical load balance. The *electrical load balance* states that the production must equal the demands:

$$f^{\mathrm{ne}}(t) + f^{\mathrm{ge}}(t) = f^{\mathrm{b}}(t) + d^{\mathrm{el}}(t) \ . \tag{5.1}$$

We now comment the different terms. In the left hand side of Equation (5.1), the load produced consists of

- the import into the microgrid from the distribution network $f^{\mathrm{ne}}(t)$ (in kW),
- the electrical production of the μCHP $f^{\mathrm{ge}}(t)$ (in kW).

In the right hand side of Equation (5.1), the electrical demand is the sum of

- the power sent to the battery $f^{\mathrm{b}}(t)$ (in kW),
- the inflexible demands (lightning, cooking...), aggregated in a single demand $d^{\mathrm{el}}(t)$ (in kW).

Thermal load balance. We now detail the input and the output of the hot water tank at a given time t. The input of the tank is equal to

$$q^{\mathrm{in}}(t) = \beta^{\mathrm{gh}} f^{\mathrm{gh}}(t) + \beta^{\mathrm{burn}} f^{\mathrm{burn}}(t) \ , \tag{5.2a}$$

where f^{gh} is the heat produced by the μCHP and f^{burn} the heat produced by the auxiliary burner (we denote by β^{chp} and β^{burn} the transmission yields). The tank's output is

$$q^{\mathrm{out}}(t) = d^{\mathrm{hw}}(t) + f^{h}(t) \ , \tag{5.2b}$$

with d^{hw} the hot water demand and f^{h} the heating demand. All flows are given in kW.

5.2.2. Uncertainties, controls and states

We now take into account the different load balance equations stated in §5.2.1 to define a discrete time stochastic model. We use a fixed time step $\Delta T = 15$ mn, and we discretize the different continuous equations described in Chapter 4. We describe hereafter the uncertainties, the decisions and the states vectors we will consider.

We adopt the following convention for discrete processes: for $t \in \{0, 1, \cdots, T = \frac{T_0}{\Delta T}\}$, we set $x_t = x(t\Delta T)$. That is, x_t denotes the value of the variable x at the beginning of the interval $[t\Delta T, (t{+}1)\Delta T[$. Otherwise stated, we will denote by $[t, t{+}1[$ the continuous time interval $[t\Delta T, (t{+}1)\Delta T[$.

5.2.2.1. Modeling uncertainties as random variables

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space.

Due to the unpredictable nature of electrical and domestic hot water demands we model the uncertainties as random variables on the space $(\Omega, \mathcal{F}, \mathbb{P})$. We adopt the following convention: a random variable will be denoted by an uppercase bold letter $\boldsymbol{Z}$ and its realization will be denoted in lowercase $z = \boldsymbol{Z}(\omega)$ for $\omega \in \Omega$. For each discrete time step $t \in \{1, .., T\}$, we define the uncertainty vector $\boldsymbol{W}_t : \Omega \to \mathbb{W}_t$ as

$$\boldsymbol{W}_t = (\boldsymbol{D}_t^{\mathrm{el}}, \boldsymbol{D}_t^{\mathrm{hw}}) \ . \tag{5.3}$$

The uncertainty takes value in the set $\mathbb{W}_t = \mathbb{R}^2$.

Remark 5.2.1. *The solar radiations Φ^b, Φ^d and the external temperature θ^e (see §4.2.1) are supposed deterministic in this chapter.* ◊

5.2.2.2. Taking decisions to satisfy energy balances

The randomness of uncertainties contaminates the decision variables, as they will depend on previous history.

At time t, the EMS takes four decisions: how much energy to charge/discharge with the battery $\boldsymbol{F}_t^{\mathrm{b}}$, how much energy to heat the thermal heater $\boldsymbol{F}_t^{h}$, do we switch on the μCHP or not $\boldsymbol{Y}_t$ and how many energy we produce with the auxiliary burner $\boldsymbol{F}_t^{\mathrm{burn}}$. The decision vector writes

$$\boldsymbol{U}_t = (\boldsymbol{F}_t^{\mathrm{b}}, \boldsymbol{F}_t^{h}, \boldsymbol{F}_t^{\mathrm{burn}}) \ , \tag{5.4}$$

taking values in $\mathbb{U}_t = \mathbb{R}^3$. We consider the Boolean variable $\boldsymbol{Y}_t \in \{0, 1\}$ separately.

Electrical load balance. The load balance equation (5.1) rewrites in discrete time:

$$\boldsymbol{F}_{t+1}^{\mathrm{ne}} = \boldsymbol{D}_{t+1}^{\mathrm{el}} + \boldsymbol{F}_t^{\mathrm{b}} - \boldsymbol{F}_t^{\mathrm{ge}} \ . \tag{5.5}$$

That is, the importation from the distribution network $\boldsymbol{F}_{t+1}^{\mathrm{ne}}$ plays the role of the *recourse* variable to ensure that the production is equal to the demand whatever the realization of the uncertainty $\boldsymbol{D}_{t+1}^{\mathrm{el}}$ between t and $t+1$. The importation $\boldsymbol{F}_{t+1}^{\mathrm{ne}}$ is unconstrained, and depends linearly on the uncertainties $\boldsymbol{W}_{t+1} = (\boldsymbol{D}_{t+1}^{\mathrm{el}}, \boldsymbol{D}_{t+1}^{\mathrm{hw}})$ and on the decision $\boldsymbol{U}_t = (\boldsymbol{F}_t^{\mathrm{b}}, \boldsymbol{F}_t^{h}, \boldsymbol{F}_t^{\mathrm{burn}})$.

Thermal balance. In a discrete time setting, the thermal balance (5.2) rewrites

$$\boldsymbol{Q}_t^{\mathrm{in}} = \beta^{\mathrm{gh}} \boldsymbol{F}_t^{\mathrm{gh}} + \beta^{\mathrm{burn}} \boldsymbol{F}_t^{\mathrm{burn}} \ , \tag{5.6a}$$

for the input, and

$$\boldsymbol{Q}_{t+1}^{\mathrm{out}} = \boldsymbol{F}_t^{h} + \boldsymbol{D}_{t+1}^{\mathrm{hw}} \ , \tag{5.6b}$$

for the output, with $\boldsymbol{F}_t^{h}$ the heating need and $\boldsymbol{D}_{t+1}^{\mathrm{hw}}$ the hot water demand between time t and $t+1$.

5.2.2.3. States and dynamics

For time $t \in \{0, .., T\}$, the state gathers the stock in the battery $\boldsymbol{B}_t$, the level of the hot water tank $\boldsymbol{H}_t$ and the two temperatures of the thermal envelope $(\boldsymbol{\theta}_t^{\mathrm{w}}, \boldsymbol{\theta}_t^{\mathrm{i}})$. The state random variable $\boldsymbol{X}_t$ writes

$$\boldsymbol{X}_t = (\boldsymbol{B}_t, \boldsymbol{H}_t, \boldsymbol{\theta}_t^{\mathrm{w}}, \boldsymbol{\theta}_t^{\mathrm{i}}) \ , \tag{5.7}$$

taking values in $\mathbb{X}_t = \mathbb{R}^4$.

We saw in §4.4 that the stocks satisfy the ordinary differential equation

$$\frac{dx}{dt} = Ax(t) + Bu(t) + Cw(t) \ , \tag{5.8}$$

with A, B, C dynamics matrix corresponding to equations (4.9)-(4.13)-(4.14).

Thus, the state equation of the system writes:

$$\boldsymbol{X}_{t+1} = f_t\big(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{Y}_t, \boldsymbol{W}_{t+1}\big) = A_d \boldsymbol{X}_t + B_d \boldsymbol{U}_t + C_d \boldsymbol{W}_{t+1} \ , \tag{5.9}$$

with $f_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{Y}_t \times \mathbb{W}_{t+1} \to \mathbb{X}_{t+1}$ the linear dynamics corresponding to the integration of Equation (5.8) over a time ΔT. We have:

$$A_d = \exp(A\Delta T) \ , \ \ B_d = \int_0^{\Delta T} \exp(As) B ds \ , \ \ C_d = \int_0^{\Delta T} \exp(As) C ds \ . \tag{5.10}$$

5.2.2.4. Non-anticipativity constraints

The future realizations of uncertainties are unpredictable, as we do not know the future demands. The current decisions are functions only of the previous history, that is, the realization of uncertainties between time 0 up to time t. The so-called non-anticipativity constraint writes

$$\sigma(\boldsymbol{U}_t) \subset \mathcal{F}_t \ , \tag{5.11}$$

where $\sigma(\boldsymbol{U}_t)$ is the σ-algebra generated by $\boldsymbol{U}_t$ and

$$\mathcal{F}_t = \sigma(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_t) \ , \tag{5.12}$$

is the σ-algebra associated to the previous history $(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_t)$. If Constraint (5.11) holds true, the Doob lemma ensures that there exists a function π_t such that

$$\boldsymbol{U}_t = \pi_t(x_0, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_t) \ . \tag{5.13}$$

This is how we turn an (abstract) algebraic constraint (5.11) into a more practical functional constraint (5.13). The function π_t is an example of policy depending on previous history, as described in Section 3.2.

5.2.2.5. Bounds constraints

Using Equations (4.10a)—(4.12), we know that the stocks in the battery and in the hot water tank are bounded.

In Equation (5.9), the control $\boldsymbol{F}_t^{\mathrm{b}}$ must ensure that the next state $\boldsymbol{B}_{t+1}$ is admissible, that is, $\underline{b} \leq \boldsymbol{B}_{t+1} \leq \overline{b}$ by Equation (4.10a), which rewrites,

$$\underline{b} \leq \boldsymbol{B}_t + \Delta T \big[\rho_c (\boldsymbol{F}_t^{\mathrm{b}})^+ + \frac{1}{\rho_d}(\boldsymbol{F}_t^{\mathrm{b}})^-\big] \leq \overline{b} \ . \tag{5.14}$$

Thus, the constraints on $\boldsymbol{F}_t^{\mathrm{b}}$ depends on the stock $\boldsymbol{B}_t$. The same reasoning applies for the tank. Furthermore, we set bound constraints on controls, that is,

$$-\overline{f}^{\mathrm{b}} \leq \boldsymbol{F}_t^{\mathrm{b}} \leq \overline{f}^{\mathrm{b}} \ , \ \ 0 \leq \boldsymbol{F}_t^{h} \leq \overline{f}_t^{h} \ , \ \ 0 \leq \boldsymbol{F}_t^{\mathrm{burn}} \leq \overline{f}_t^{\mathrm{burn}} \ . \tag{5.15}$$

Finally the load-balance equation (6.5) also acts as a constraint on the controls. We gather constraints (5.14)-(5.15) in an admissible set depending on the current state $\boldsymbol{X}_t$:

$$\boldsymbol{U}_t \in \mathcal{U}_t^{ad}(\boldsymbol{X}_t) \ . \tag{5.16}$$

5.2.3. Writing down the optimization problem

We have stated in §5.2.2 the different physical equations governing the system. We now focus on the costs we consider in the optimization model.

5.2.3.1. Objective

We consider two kinds of costs: the operational cost — corresponding to the price we pay to import and produce energy — and a virtual discomfort cost, added to ensure that the temperatures inside the house remains higher than a given setpoint.

Energy costs. At time t, we pay:

- the cost to use gas in auxiliary burner: $p^g \times \boldsymbol{F}_{t+1}^{\text{burn}}$,
- the cost to use gas in μCHP generator: $p^{\text{chp}} \times \boldsymbol{Y}_t$.

The electricity bill corresponds to the difference between importation and exportation of electricity from the network:

$$-\underbrace{p_t^{\text{inj}} \max\{0, -\boldsymbol{F}_{t+1}^{\text{ne}}\}}_{\text{selling}} + \underbrace{p_t^{\text{el}} \max\{0, \boldsymbol{F}_{t+1}^{\text{ne}}\}}_{\text{buying}} .$$

Remark 5.2.2. *We assume that $p_t^{el} \geq p_t^{inj}$. Otherwise, it would become possible to import electricity at a given price and to sell it immediately at a greater price, which is economically inconsistent. As a consequence, the electricity costs is convex.* ◊

Cost of discomfort. We penalize the inner temperature if it is below a given setpoint $\bar{\theta}_t^{\text{i}}$, so as to ensure a minimal comfort for the inhabitants. The setpoint $\bar{\theta}_t^{\text{i}}$ depends on time t, and is fixed by the inhabitants according to their preference. The indoor temperature is penalized by the piecewise-linear function:

$$\kappa_{th}(\boldsymbol{\theta}_t^{\text{i}}) = p_t^{\text{th}} \times \max\{0, \bar{\theta}_t^{\text{i}} - \boldsymbol{\theta}_t^{\text{i}}\} , \tag{5.17}$$

where p_t^{th} is a virtual cost of discomfort.

Operational cost. The operational cost $L_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{Y}_t \times \mathbb{W}_{t+1} \to \mathbb{R}$ gathers all costs and writes at time t:

$$\begin{aligned} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{Y}_t, \boldsymbol{W}_{t+1}) = {} & p^{\text{chp}} \boldsymbol{Y}_t + p^g \boldsymbol{F}_t^{\text{burn}} \\ & - p_t^{\text{inj}} \max\{0, -\boldsymbol{F}_{t+1}^{\text{ne}}\} + p_t^{\text{el}} \max\{0, \boldsymbol{F}_{t+1}^{\text{ne}}\} + p_t^{\text{th}} \times \max\{0, \bar{\theta}_t^{\text{i}} - \boldsymbol{\theta}_t^{\text{i}}\} . \end{aligned} \tag{5.18}$$

Remark 5.2.3. *In Equation (5.18), we mix a virtual price p^{th} with real prices p^{chp}, p^g and p^{el}. The price p^{th} is chosen by the decision maker, so as to heat the house effectively. Another method would consist to use multi-criteria optimization to find the relative penalization of thermal comfort w.r.t. the energy costs.* ◊

5.2.3.2. Building an optimization problem

Now that we have made explicit the dynamics and the different costs, we write a global optimization problem:

$$V^\sharp = \min_{\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{Y}} \ \mathbb{E}\left[\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{Y}_t, \boldsymbol{W}_{t+1}) + K(\boldsymbol{X}_T) \right] , \tag{5.19a}$$

$$\text{s.t. } \boldsymbol{X}_{t+1} = f_t\big(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{Y}_t, \boldsymbol{W}_{t+1}\big) , \ \ \boldsymbol{X}_0 = x_0 , \tag{5.19b}$$

$$\boldsymbol{Y}_t \in \{0, 1\} , \tag{5.19c}$$

$$\boldsymbol{U}_t \in \mathcal{U}_t^{ad}(\boldsymbol{X}_t) , \tag{5.19d}$$

$$\sigma(\boldsymbol{U}_t) \subset \mathcal{F}_t . \tag{5.19e}$$

We suppose that measurability and integrability assumptions hold, so that the expression in (5.19a) makes sense. Problem (5.19) induces two challenges:

1. The Boolean constraint (5.19c) renders the problem non-convex.
2. The number T of stages is too large to keep Problem (5.19) numerically tractable. Indeed, we want to minimize the costs over one year at a 15 minutes time-step, giving $T = 35,040$ time-steps.

To render Problem (5.19) tractable, we propose to decompose it in time to obtain smaller subproblems.

5.2.3.3. Time decomposition

We split the initial problem in daily subproblems, as the system allows to fill the battery and the water tank completely between midnight and 6am. Thus, we obtain 365 subproblems. For all day $d \in \{0, \cdots, 364\}$, we define:

$$J_d(x^d) = \min_{\boldsymbol{X},\boldsymbol{U},\boldsymbol{Y}} \mathbb{E}\left[\sum_{t=96d}^{96(d+1)-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{Y}_t, \boldsymbol{W}_{t+1})\right], \tag{5.20a}$$

$$\text{s.t. } \boldsymbol{X}_{t+1} = f_t\big(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{Y}_t, \boldsymbol{W}_{t+1}\big), \ \ \boldsymbol{X}_{96d} = x^d, \tag{5.20b}$$

$$\boldsymbol{Y}_t \in \{0,1\}, \tag{5.20c}$$

$$\boldsymbol{U}_t \in \mathcal{U}_t^{ad}(\boldsymbol{X}_t), \tag{5.20d}$$

$$\sigma(\boldsymbol{U}_t) \subset \mathcal{F}_t. \tag{5.20e}$$

To ensure that Problem (5.20) remains coherent with Problem (5.19), we set

$$x^{d+1} = \mathbb{E}[\boldsymbol{X}_{96(d+1)}]. \tag{5.21}$$

To ease the notation, we define the daily horizon of day d:

$$T_d = 96(d+1). \tag{5.22}$$

We note that each day subproblem has 96 time steps and has not a final cost. The reason is that we are able to refill the tank between midnight and 6am, when the inhabitants start using the heating and the domestic hot water. In a similar manner, the battery can be refilled in four hours, thus avoiding the need to keep energy stocks at midnight.

By imposing constraint (5.21), we add a new constraint to Problem (5.19). Thus, we have

$$V^\sharp \leq \sum_{d=0}^{D} J_d(x^d), \ \ \forall (x^0, \cdots, x^D) \in \mathbb{X}^{D+1}, \tag{5.23}$$

where $D = 364$ and $V^\sharp$ is the optimal value of Problem (5.19).

In §5.3, we will present different algorithms to solve the different subproblems in (5.20), considering the coupling defined in the equation (5.21).

5.3. Resolution methods

We look for online policies $\pi_t : \mathbb{X}_0 \times \mathbb{W}_1 \times \cdots \times \mathbb{W}_t \to \mathbb{U}_t$ that map the available information at time t to a decision u_t, as introduced in Section 3.2. We present three algorithms to design such policies: the first based on a (s, S) policy, the second on Stochastic Dual Dynamic Programming (SDDP), and the third based on Model Predictive Control (MPC).

As Problems (5.20) are now decomposed day by day, the main challenge is to tackle the Boolean switch $\boldsymbol{Y}_t$.

In the following, we detail the different policies for a given day $d \in \{0, .., 364\}$.

5.3.1. (s, S) policy

The (s, S) policy is a simple heuristic, widely used to control the levels of stocks. The algorithm is the following.

- We consider the setpoint $\overline{\theta^{\mathrm{i}}}$ defined in (5.17) for the inner temperature.
- If the temperature is below this setpoint minus a given threshold $\Delta\theta^i$,

$$\boldsymbol{\theta}^{\mathrm{i}} \leq \overline{\theta^{\mathrm{i}}} - \Delta\theta^{\mathrm{i}} \ , \tag{5.24}$$

  then we set the heating to $\mathbf{F}_t^h = K_p \times (\theta^{\mathrm{i}} - \Delta\theta^{\mathrm{i}} - \boldsymbol{\theta}^{\mathrm{i}})$. K_p is a parameter of the algorithm that is hand tuned.
- If the level of the tank is below a given threshold $\underline{h}^s$, we switch on the μCHP ($\boldsymbol{Y}_t = 1$) till the tank is filled.
- We use the auxiliary burner as a recourse if the thermal tank is empty.

In the sequel, we will denote by π^{sS} this policy.

5.3.2. Model Predictive Control (MPC)

We reconsider the MPC online policy introduced in §3.3.2. Let $\overline{w}$ be a forecast for future uncertainties $\boldsymbol{W}_{t+1}, \cdots, \boldsymbol{W}_{T_d}$. The policy π_t^{mpc} writes at time $t \in \{0, .., T_d - 1\}$:

$$\pi_t^{\mathrm{mpc}}(x_t) \in \underset{x,u,y}{\arg\min} \Big[\sum_{j=t}^{T_d-1} L_j(x_j, u_j, y_j, \overline{w}_{j+1}) \Big] \ , \tag{5.25}$$

$$\begin{aligned} \text{s.t.} \quad x_{j+1} &= f_j(x_j, u_j, y_j, \overline{w}_{j+1}) \ , \\ y_j &\in \{0, 1\} \ , \\ u_j &\in \mathcal{U}_j^{ad}(x_j) \ , \end{aligned}$$

with $T_d = 96(d + 1) - 1$ the daily horizon of the subproblem corresponding to day d.

In Problem (5.25), dynamics f_t is linear and cost L_t is piecewise-linear for all t. However, due to the Boolean constraint (5.20c), the problem (5.25) is a Mixed-Integer Linear Problem (MILP), thus complicating the resolution.

5.3.3. Stochastic Dual Dynamic Programming (SDDP)

We reconsider the framework introduced in Section 3.3 to describe SDDP policy as a cost-to-go based policy. The procedure of SDDP has two distinct stages: an offline stage to compute value functions and an online stage to compute decisions on the fly.

5.3.3.1. Offline stage

We reconsider the notation introduced in Section 3.2. Let $\{\rho_{t+1}^{of}\}_{t\in\{0,..,T-1\}}$ be a family of stochastic kernels that do not depend on previous history

$$\rho_{t+1}^{of}(h_t, \cdot) = \mu_t^{of}(\cdot) \ , \ \ \forall t \in \{0, .., T\} \ , \tag{5.26}$$

where $\mu_t^{of} \in \Delta(\mathbb{W}_{t+1})$ is a discrete probability measure defined on $\mathbb{W}_{t+1}$ (see §3.2.1.2).

Let $\mathbb{W} = \prod_{t=1}^{T} \mathbb{W}_t$ and $(w^1, \cdots, w^N) \in \mathbb{W}^N$ be N optimization scenarios (see §3.5.2.2) defined as

$$w^i = (w_1^i, \cdots, w_{T_d}^i) \in \mathbb{W} \ , \ \ \forall i \in \{1, .., N\} \ . \tag{5.27}$$

We fix a quantization size S and compute for all time $t \in \{1, .., T_d\}$ the marginal distributions μ_t^{of} by quantization using the procedure described in §4.2.2 upon the samples $(w_t^1, \cdots, w_t^N) \in \mathbb{W}_t^N$ defined by the N optimization scenarios (5.27).

Once the marginal distributions $\{\mu_t^{of}\}_{t\in\{1,..,T_d\}}$ are computed, we are able to compute offline a sequence of value functions corresponding to day d, using the procedure described in Section 3.4. Let $\{V_t\}_{t\in\{0,..,T_d\}}$ be a sequence of value functions, defined by the recursive equations

$$V_{T_d}(x) = 0 \,, \tag{5.28a}$$

$$\begin{aligned} V_t(x_t) = \min_{u,y} & \int_{\mathbb{W}_{t+1}} \big[L_t(x_t, u, y, w_{t+1}) + V_{t+1}\big(f_t(x_t, u, y, w_{t+1})\big)\big]\mu_{t+1}^{of}(dw_{t+1}) \\ & \text{s.t. } y \in \{0, 1\} \,, \;\; u \in \mathcal{U}_t^{ad}(x_t) \,. \end{aligned} \tag{5.28b}$$

The value functions (5.28) are the optimal Bellman value functions of Problem (5.20) when the uncertainties $(\boldsymbol{W}_0, \cdots, \boldsymbol{W}_{T_d})$ are stagewise independent and the distribution of $\boldsymbol{W}_t$ is the probability measure μ_t^{of}.

We aim to solve the recursive equations (5.28) by SDDP. However, the one-step problem (5.28b) is non-convex, because of the switch $y \in \{0, 1\}$. Let $\{\underline{V}_t\}_{t\in\{0,..,T_d\}}$ be another sequence of value functions that satisfies instead the following recursive equations:

$$\underline{V}_{T_d}(x) = 0 \,, \tag{5.29a}$$

$$\begin{aligned} \underline{V}_t(x_t) = \min_{u,y} & \int_{\mathbb{W}_{t+1}} \big[L_t(x_t, u(w_{t+1}), y, w_{t+1}) + \underline{V}_{t+1}\big(f_t(x_t, u(w_{t+1}), y, w_{t+1})\big)\big]\mu_{t+1}^{of}(dw_{t+1}) \\ & \text{s.t. } y \in [0, 1] \,, \;\; u(w_{t+1}) \in \mathcal{U}_t^{ad}(x_t) \,, \;\; u \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{U}_t) \,, \end{aligned} \tag{5.29b}$$

where $\mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{U}_t)$ is the space of $\mathcal{F}$ measurable functions $\phi : \Omega \to \mathbb{U}_t$. Problem (5.29) is of *wait-and-see* type [1] (Wets, 2002), as we allow decisions to depend on the realizations of the uncertainty $\boldsymbol{W}_{t+1}$ between time t and $t + 1$. Thus, the decision u becomes a measurable function in (5.29).

In Problem (5.29), the costs L_t are convex w.r.t. (x, u), the dynamics f_t is affine w.r.t. (x, u), the probability distribution μ_{t+1}^{of} is finite and the admissible set is convex (because of the relaxation $y \in [0, 1]$. Thus, we recover the seminal hypotheses of SDDP and we know that the computation of the value functions (5.29) by the SDDP algorithm converges (Girardeau et al., 2014).

As Problem (5.29) is a relaxation of Problem (5.28), we are able to prove by induction that the value functions $\{\underline{V}_t\}_{t\in\{0,..,T_d\}}$ are lower-bounds of the value functions $\{V_t\}_{t\in\{0,..,T_d\}}$:

$$\underline{V}_t \leq V_t \,, \;\; \forall t \in \{0, .., T_d\} \,. \tag{5.30}$$

5.3.3.2. Online stage

During the online stage, we suppose given a set of value functions $\{\underline{V}_t\}_{t\in\{0,..,T_d\}}$ and a sequence of discrete distributions $\{\mu_t^{on}\}_{t\in\{1,..,T\}}$ such that

$$\mu_t^{on}(\cdot) = \sum_{s=1}^{S} p_s \delta_{\overline{w}_t^s}(\cdot) \,, \;\; \forall t \in \{1, .., T\} \,, \tag{5.31}$$

where $(\overline{w}_t^1, \cdots, \overline{w}_t^S) \in \mathbb{W}_t^S$ are the support size of the online distribution μ_{t+1}^{on} and $(p_1, \cdots, p_S)$ are associated probability weights.

[1] Or Hazard-Decision.

Then, we are able to compute decisions at each time $t \in \{0, .., T-1\}$ with the strategy π^{sddp}

$$\begin{aligned} \pi_t^{\text{sddp}}(x_t) \in \arg\min_{u,y} & \sum_{s=1}^{S} p_s \big[L_t(x_t, u, y, w_{t+1}^s) + \underline{V}_{t+1}\big(f_t(x_t, u, y, w_{t+1}^s)\big)\big] \\ \text{w.r.t.} \quad & y \in \{0,1\}\ , \ \ u \in \mathcal{U}_t^{ad}(x_t)\ . \end{aligned} \tag{5.32}$$

The cost L_t in Equation (5.32) is piecewise-linear, so is the dynamics f_t. We note that, as the constraint (5.20c) is handled explicitly, the problem is a MILP.

5.4. Numerical results

We now apply the three algorithms introduced in §5.3 to a specific case study.

5.4.1. Case study

Prices. We consider on and off-peak hours tariffs for electricity: we set $p_t^{\text{el}} = 0.15$ €/kWh during day (between 7h and 23h) and $p_t^{\text{el}} = 0.09$ €/kWh during night. We consider fixed prices for gas $p^g = 0.06$ €/kWh and $p^{\text{th}} = 0.4$ €/°C for comfort.

Model's parameters. The different parameters of the thermal model of §4.4.3 correspond to the RT 88 specifications (see Annex 4.5.1). The μCHP is a Stirling engine, whose specifications are given in Annex 4.5.2. We use a Lithium-Ion battery, with a size of 3 kWh. The hot water tank has a volume of 500 l.

The solar radiations and the external temperatures are provided by a data-set given by the IFPEB[2], and correspond to the evolution of weather in Orly, France, during the year 2015.

Out-of-sample assessment. We consider 1,000 optimization scenarios and 1,000 assessment scenarios of demands. These scenarios are weekly periodic, to take into account the fact that electricity consumption decreases during week-ends.

5.4.2. Numerical implementation

We implement MPC and SDDP in Julia 0.6, using JuMP as a modeler, `StochDynamicProgramming.jl` as a SDDP solver, and CPLEX 12.5 as a MILP solver. The exact resolution of the MILP problems is generally out-of-reach, and we use a MIPGAP equal to 1% to solve both Problem (5.25) and Problem (5.32). All computations run on a Core i7 2.5 GHz processor, with 16 Go RAM.

5.4.3. Results

We first compare in §5.4.3.1 MPC with the (s, S) policy, and then compare in §5.4.3.2 MPC with SDDP. These two comparisons are performed on the same case study, on the same set of 1,000 assessment scenarios. We use the assessment procedure described in Section 3.5.

5.4.3.1. Comparing MPC with (s, S) policy

We first compare MPC with the (s, S) policy. We present here the results obtained upon assessment scenarios, over one year. Figure 5.2 displays the average daily costs for each day. Table 5.1 details the different costs we pay to operate the system.

[2]Institut Français pour la performance du bâtiment

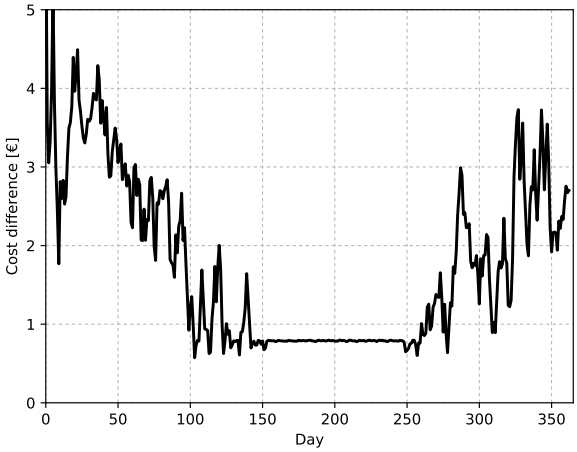


Figure 5.2.: Difference between MPC's daily costs and heuristic's daily costs over one year. As the difference is above 0, MPC beats the heuristics.

	Euros/year	%
$\mathfrak{A}(\pi^{sS})$	2,075	ref
$\mathfrak{A}(\pi^{\text{mpc}})$	1,627	- 28.3%

Table 5.1.: Comparing MPC with a (s, S) policy.

We observe that MPC outperforms the current heuristic in assessment, achieving up to 450 € costs savings. In Figure 5.2, we observe that the cost savings are more important in winter, when we are using the μCHP to heat the building, than in summer, when the μCHP is off. This first study illustrates that it pays to use optimization algorithms such as MPC.

5.4.3.2. Comparing MPC with SDDP

We now compare MPC with SDDP. We consider the same parameters as in the previous comparison.

Comparing the numerical performance. Table 5.2 shows the offline and online execution times of MPC and SDDP. We observe that the time to take a decision online is almost similar between SDDP and MPC. It takes approximately 2' to compute the value functions by SDDP for one day.

	MPC	SDDP
Offline	.	110s
Online	0.010s	0.004s

Table 5.2.: Comparison of SDDP and MPC execution time

Computing cuts offline with SDDP. SDDP is a powerful method to compute value functions as a supremum of affine hyperplanes. The stopping criterion of SDDP relies only on statistical upper-bound, and is renowned for its lack of precision (Shapiro, 2011).

We analyze how many iterations are required for SDDP convergence. We consider a particular day in winter ($d = 10$) and compare the evolution of SDDP's lower bound with the evolution of SDDP's upper bound, estimated every 10 iterations with 1,000 scenarios. Results are given in Figure 5.3. We observe that SDDP converges quickly: the gap between the lower and the upper bounds is less than 0.5 % in about 42 iterations (around 5s execution time), and we observe that the upper bound remains stable after 120 iterations (around 20s execution time).

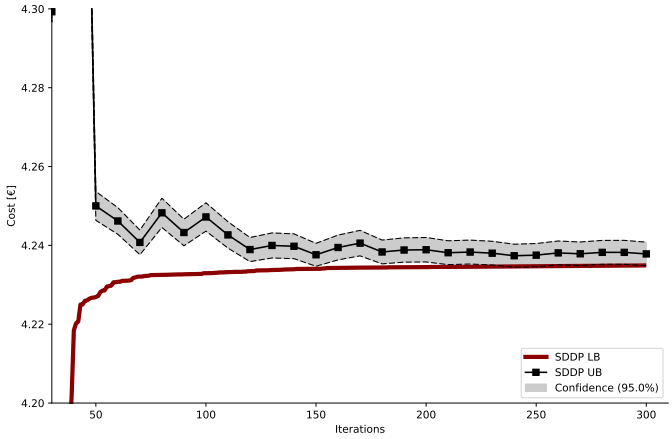


Figure 5.3.: Illustrating the convergence of SDDP, on the linear relaxation of the stochastic MIP, for $d = 10$ (January, 10th).

Comparing the operational results. We detail the operational costs in Table 5.3. In term of energy savings, SDDP shortly beats MPC, by a discrepancy of 0.6%. We observe in Table 5.3 that SDDP consumes less electricity than MPC, and uses the μCHP more (comparing the μCHP uptime), whereas MPC uses more the auxiliary burner to produce heat.

Figure 5.5 displays the average μCHP use over year, for both MPC and SDDP (the darker, the heavier μCHP usage is). We observe that the μCHP runs mostly during winter, when outdoor temperature is low, but is not used during summer. The μCHP trajectories are almost similar for MPC and SDDP. It starts around 5am to fulfill the heating needs and is switched off after 9pm.

Figure 5.4 compares the relative difference between MPC and SDDP daily operational costs. We observe that SDDP beats MPC during winter, whereas the performance of SDDP and MPC are similar during summer, when the μCHP is off. We observe that MPC shortly beats SDDP by a discrepancy being less than 1 % for some days in spring and fall, although the discrepancy is small.

	unit	MPC	SDDP
Cost	€	1627	1617
Electricity bill	€	682	668
μCHP cost	€	833	865
Burner cost	€	101	76
Discomfort	.	11	11
μCHP uptime	h	1984	2059

Table 5.3.: Operational costs for MPC and SDDP. Total electricity consumption is 6,379 kWh.

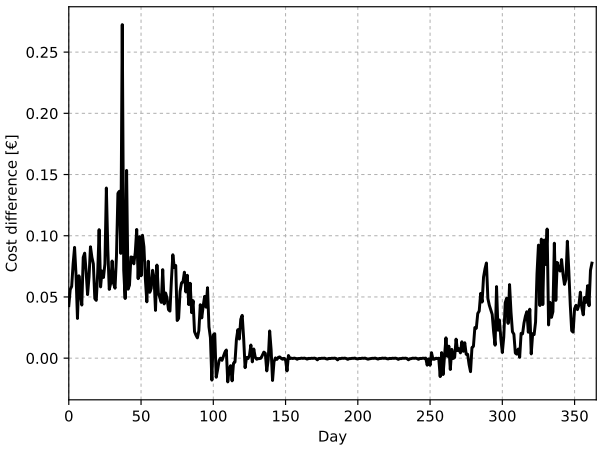


Figure 5.4.: Difference between SDDP's and MPC's daily costs, in percentage. When the difference is greater than 0, SDDP beats MPC, otherwise MPC is better.

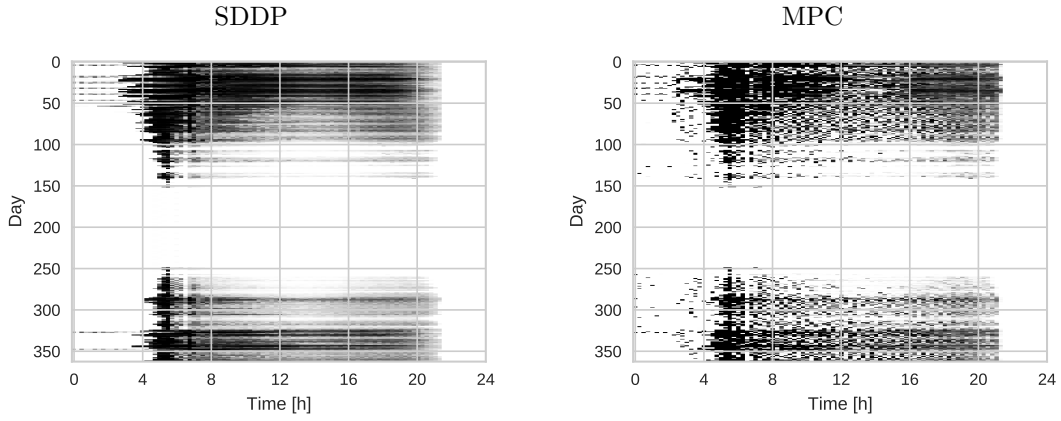


Figure 5.5.: μCHP use over one year, for both SDDP (left) and MPC (right).

Comparing the optimal trajectories. We display on Figure 5.6 the trajectories of battery stocks, and on Figure 5.7 the trajectories of the hot water tank. We observe that SDDP uses less the battery than MPC on the assessment trajectories, and that MPC has a higher use of the hot water tank during the morning.

5.4.3.3. Comparing different microgrid configurations

Finally, we compare different configurations for the microgrid. As we have seen in §5.4.3.2 that the performances of SDDP and MPC are almost equivalent, we compute the operational costs over one year with MPC.

We test different battery's sizes (0, 1.5, 3 and 4.5kWh) and two injection prices: $p^{\text{inj}} = 0$ corresponds to the case when no injection is allowed, $p^{\text{inj}} = 0.088$ €/kWh corresponds to EDF's reinjection tariff (Bonabe de Rougé, 2018). The goal of this comparison is too emphasize the impact of the battery's size upon the controller's performance. The results obtained with MPC are

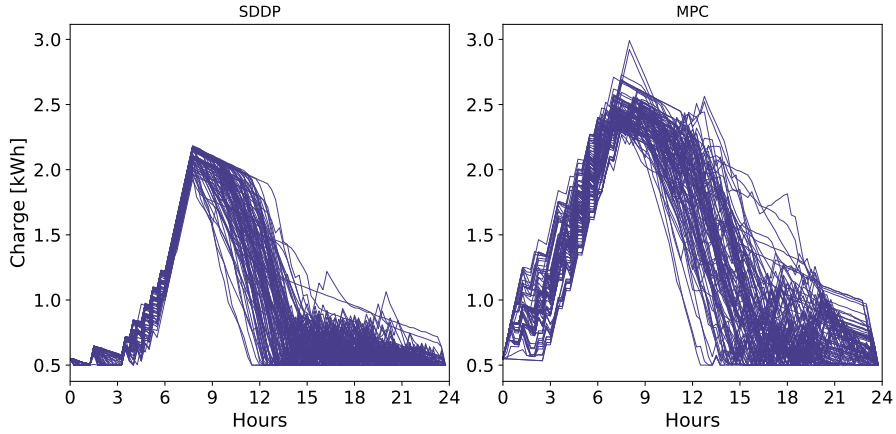


Figure 5.6.: Battery charge trajectories for SDDP and MPC during one day in winter.

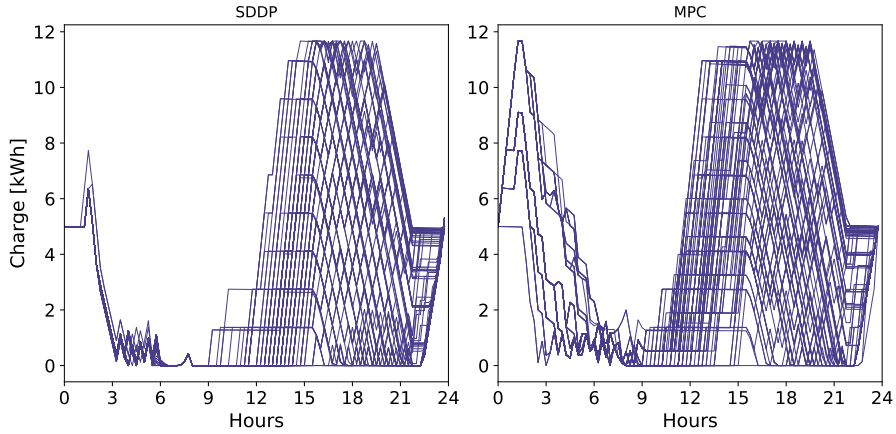


Figure 5.7.: Thermal hot water tank charge trajectories for SDDP and MPC during one day in winter.

shown in Table 5.4. As the performance of SDDP is close to that of MPC, we choose to present the comparison only for MPC.

We comment further the results in Table 5.4.

- If $p^{\text{inj}} = 0$€, using a battery allows to increase the μCHP's uptime by 30%, and decrease operational costs by up to 4% (the bigger the battery, the lower the cost).

- If $p^{\text{inj}} = 0.088$€, the battery's size has less impact in term of cost savings, as the decision maker can sell the surplus of electricity onto the external grid. The self-production is higher than the case without injection, and remains almost constant w.r.t. the battery's size. However, we observe that increasing the battery's size allows to reduce the burner's cost.

- In term of operational cost, it is equivalent to have a system with a 3kWh battery and no injection cost and a system with no battery and injection cost.

	unit	0kWh	1.5kWh	3kWh	4.5kWh
$p^{\text{inj}} = 0$ €/kWh					
Cost	€	1703	1630	1627	1604
Electricity bill	€	775	705	682	678
μCHP cost	€	628	805	833	812
Burner cost	€	286	108	101	102
Discomfort	v.a	14	12	11	12
μCHP uptime	h	1497	1917	2059	1935
Self-production	%	19.9	25.4	27.3	25.7
$p^{\text{inj}} = 0.088$ €/kWh					
Cost	€	1621	1602	1586	1575
Electricity bill	€	690	670	653	643
μCHP cost	€	859	880	883	884
Burner cost	€	61	41	39	37
Discomfort	v.a	11	11	11	11
μCHP uptime	h	2045	2097	2102	2106
Self-production	%	27.1	27.8	27.9	28.0

Table 5.4.: Comparison of yearly operational costs obtained with different configurations. The annual electricity consumption for the considered household is 6,379 kWh.

5.5. Discussion

The contributions of Chapter 5 are twofold. First, we have showed in §5.4.3.1 that using optimization algorithms substantially increases the economic performance. Second, a comparison of MPC with SDDP showed that it does not pay too much to use stochastic optimization algorithms: SDDP policy is only 0.6% better than MPC policy during assessment. We observed in Figure 5.4 that SDDP policy beats MPC policy mostly during winter and fall, when heating is on. In this period, SDDP policy is in average 1% better than MPC policy. Otherwise, in summer, their performances are equivalent, as the μCHP is off and the problem becomes almost deterministic.

We will consider a new case study in Chapter 6, with a residential microgrid equipped with solar panels and a battery. The presence of solar panels increases the stochasticity of the system, as it adds a new perturbation in the electrical load balance equation. We aim at improving MPC and SDDP online policies. SDDP will compute its cuts in Decision-Hazard, and MPC will use a statistical model to update its forecast online. We will observe that during sunny day, when the production of the solar panels is nominal, SDDP policy beats MPC policy by up to 5% in term of cost savings.

Chapter 6.

Optimal management of a home microgrid with solar panels

Contents

6.1. Introduction

In Chapter 5, we have used two algorithms designed to tackle uncertainties: Model Predictive Control (MPC) and Stochastic Dual Dynamic Programming (SDDP). The previous study showed that the performance of these two algorithms were close on the particular example studied in Section 5.4. We want to emphasize in this chapter the sensitivity w.r.t. uncertainties of these two classes of algorithms. This chapter proposes a new case study where we replace the μCHP with solar panels. We follow the same procedure as in Chapter 5.

Even if the procedure remains the same, the present case study differs from those of Chapter 5 in several crucial points. The building is still equipped with a battery, but the electrical system becomes more complex. Besides the solar panel, we suppose here that the building is equipped with electrical heating and electrical hot water tank, thus rendering the thermal system inherently different from the one studied in Chapter 5. Thus, we no longer have to deal with Boolean constraints, as those induced by the CHP's switch. The two main sources of uncertainties become the energy demands (electrical and hot water) and the production of the solar panel.

As in Chapter 5, we take decisions every 15 minutes. However, we do not study the system over one year, but we focus on three particular days in summer, in spring and in winter. We make also the following assumptions.

- The external temperature θ^e is supposed deterministic.

- Electrical and domestic hot water demands are supposed stochastic. Here, these demands correspond to a single family with two kids.

We first give in Section 6.2 a description of the microgrid, present in Section 6.3 the different resolution methods and give, in Section 6.4, numerical results on three different cases.

6.2. Problem statement

As in Chapter 5, we consider a domestic microgrid, where the electrical and thermal demands must be satisfied. Figure 6.2 displays the configuration of the microgrid we study in this chapter.

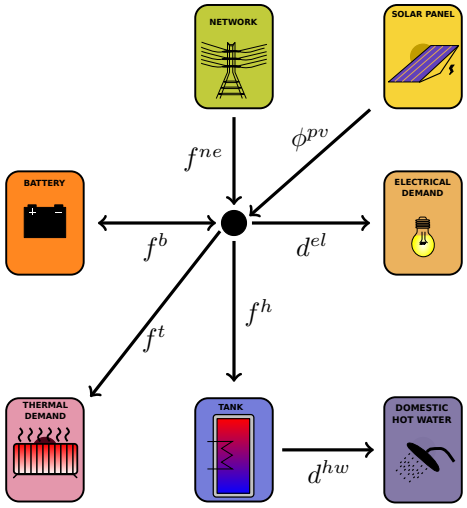


Figure 6.1.: Microgrid's configuration: the electrical system has a battery, and must satisfy the electrical demand. The electrical hot water tank uses electricity to heat water, in order to satisfy the domestic hot water demand. Electrical heaters are used to heat the house.

6.2.1. Load balance

Electrical load balance. Based on Figure 6.2, the *electrical load balance* equation of the microgrid writes, at each time t:

$$\phi^{\mathrm{pv}}(t) + f^{\mathrm{ne}}(t) = f^b(t) + f^t(t) + f^h(t) + d^{\mathrm{el}}(t)\ . \tag{6.1}$$

In the left hand side of Equation (6.1), the load produced consists of

- the production of the solar panel $\phi^{\mathrm{pv}}(t)$,
- the importation from the network $f^{\mathrm{ne}}(t)$, supposed nonnegative (we do not export electricity to the network).

In the right hand side of Equation (6.1), the electrical demand is the sum of

- the power exchanged with the battery $f^b(t)$,
- the power injected in the electrical heater $f^t(t)$,

- the power injected in the electrical hot water tank $f^h(t)$,
- the inflexible demands (lightning, cooking...), aggregated in a single demand $d^{\text{el}}(t)$.

Thermal load balance. We now focus on the thermal load balance, corresponding to the energy balance of the electrical hot water tank. At time t, the input of the tank is equal to

$$q^{\text{in}}(t) = \beta^h f^h(t) \,, \tag{6.2a}$$

where f^h is the electrical power used in the hot water tank's resistors (we denote by β^h is a transmission yields) . Tank's output is

$$q^{\text{out}}(t) = d^{\text{hw}}(t) \,, \tag{6.2b}$$

with d^{hw} the domestic hot water demand.

6.2.2. Uncertainties, controls and states

The EMS takes decisions every 15 minutes to control the system. Thus, we take decisions in discrete time.

We set $\Delta = 15\text{mn}$, and we consider an horizon $T_0 = 24\text{h}$. We adopt the following convention for discrete processes: for $t \in \{0, 1, \cdots, T = \frac{T_0}{\Delta}\}$, we set $x_t = x(t\Delta)$. That is, x_t denotes the value of the variable x at the beginning of the interval $[t\Delta, (t+1)\Delta[$. Otherwise stated, we will from now on denote by $[t, t+1[$ the continuous time interval $[t\Delta, (t+1)\Delta[$.

6.2.2.1. Modeling uncertainties as random variables

Let $(\Omega, \mathcal{F})$ be a measurable space.

Because of their unpredictable nature, we cannot anticipate the realizations of the electrical and the thermal demands. A similar reasoning applies to the production of the solar panel. We choose to model these quantities as random variables over the space $(\Omega, \mathcal{F})$. We adopt the following convention: a random variable will be denoted by an uppercase bold letter $\boldsymbol{Z}$ and its realization will be denoted in lowercase $z = \boldsymbol{Z}(\omega)$ for $\omega \in \Omega$. For each $t = 1, \ldots, T$, we introduce the uncertainty vector

$$\boldsymbol{W}_t = (\mathbf{D}_t^{\text{el}}, \boldsymbol{D}_t^{\text{hw}}, \boldsymbol{\Phi}_t^{\text{pv}}) \,, \tag{6.3}$$

modeled as a random variable. The uncertainty $\boldsymbol{W}_t$ takes value in the set $\mathbb{W}_t = \mathbb{R}^3$.

6.2.2.2. Modeling controls as random variables

As decisions depend on the previous uncertainties, the control is a random variable. We recall that, at each discrete time t, the EMS takes three decisions:

- how much energy to charge/discharge the battery $\boldsymbol{F}_t^b$,
- how much energy to store in the electrical hot water tank $\boldsymbol{F}_t^h$,
- how much energy to inject in the electrical heater $\boldsymbol{F}_t^t$.

We write the decision vector (random variable) at time t (at the beginning of $[t\Delta, (t+1)\Delta[$)

$$\boldsymbol{U}_t = (\boldsymbol{F}_t^b, \boldsymbol{F}_t^h, \boldsymbol{F}_t^t) \,, \tag{6.4}$$

taking values in $\mathbb{U}_t = \mathbb{R}^3$.

Electrical load balance. Then, between two discrete time indexes t and $t+1$, the EMS imports an energy $\boldsymbol{F}^{\mathrm{ne}}_{t+1}$ from the external network to satisfy the load balance equation (6.1) whatever the demand $\mathbf{D}^{\mathrm{el}}_{t+1}$ and the production of the solar panel $\boldsymbol{\Phi}^{\mathrm{pv}}_{t+1}$, unknown at time t. Hence $\boldsymbol{F}^{\mathrm{ne}}_{t+1}$ is a recourse decision taken at time $t+1$. The load balance equation (6.1) now writes as

$$\boldsymbol{F}^{\mathrm{ne}}_{t+1} = \boldsymbol{F}^{b}_{t} + \boldsymbol{F}^{t}_{t} + \boldsymbol{F}^{h}_{t} + \mathbf{D}^{\mathrm{el}}_{t+1} - \boldsymbol{\Phi}^{\mathrm{pv}}_{t+1} \quad \mathbb{P}-\text{a.s.}\,, \tag{6.5}$$

where $\mathbb{P}-$ a.s. indicates that the constraint is fulfilled in the almost sure sense. As the decision $\boldsymbol{F}^{\mathrm{ne}}_{t+1}$ is completely determined by other decisions $\boldsymbol{F}^{b}_{t}, \boldsymbol{F}^{t}_{t}, \boldsymbol{F}^{h}_{t}$ and uncertainties $\mathbf{D}^{\mathrm{el}}_{t+1}, \boldsymbol{\Phi}^{\mathrm{pv}}_{t+1}$, we do not include it in the decision vector $\boldsymbol{U}_t$. Later, we will aggregate the solar panel production $\boldsymbol{\Phi}^{\mathrm{pv}}_{t+1}$ with the demands $\mathbf{D}^{\mathrm{el}}_{t+1}$ in Equation (6.5), as these two quantities appear only by their sum.

The need of a recourse variable is a consequence of stochasticity in the supply-demand equation. The choice of the recourse variable depends on the modeling. Here, we choose the recourse $\boldsymbol{F}^{\mathrm{ne}}_{t+1}$ to be provided by the external network, that is, the external network mitigates the uncertainties in the system.

Thermal balance. In a discrete time setting, the thermal balance (6.2) rewrites

$$\boldsymbol{Q}^{\mathrm{in}}_{t} = \beta^{h}\boldsymbol{F}^{h}_{t}\,, \tag{6.6a}$$

for the input, and

$$\boldsymbol{Q}^{\mathrm{out}}_{t+1} = \boldsymbol{D}^{\mathrm{hw}}_{t+1}\,, \tag{6.6b}$$

for the output, with $\boldsymbol{D}^{\mathrm{hw}}_{t+1}$ the hot water demand between time t and $t+1$.

6.2.2.3. States and dynamics

The state gathers the stock in the battery $\boldsymbol{B}_t$, the energy stored in the electrical hot water tank $\boldsymbol{H}_t$, and the two temperatures of the thermal envelope $(\boldsymbol{\theta}^{\mathrm{w}}_t, \boldsymbol{\theta}^{\mathrm{i}}_t)$. The state random variable $\boldsymbol{X}_t$ writes

$$\boldsymbol{X}_t = (\boldsymbol{B}_t, \boldsymbol{H}_t, \boldsymbol{\theta}^{\mathrm{w}}_t, \boldsymbol{\theta}^{\mathrm{i}}_t)\,. \tag{6.7}$$

Thus, the state vector $\boldsymbol{X}_t$ takes values in $\mathbb{X}_t = \mathbb{R}^4$.

In §4.4, we have detailed the continuous dynamics of the battery, the hot water tank and the thermal inertia of the buildings. We follow the same procedure as in §5.2.2 to get a discrete dynamics writing

$$x_{t+1} = f_t(x_t, u_t, w_{t+1})\,, \tag{6.8}$$

with f_t corresponding to the discretization of the continuous dynamics (4.9)-(4.13)-(4.14). By doing so, we suppose that the control u_t and the uncertainty w_{t+1} are constant over the interval $[t, t+\Delta[$.

The dynamics (6.8) rewrites as an almost-sure constraint:

$$\boldsymbol{X}_{t+1} = f_t\big(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}\big) \qquad \mathbb{P}-\text{a.s.}\,. \tag{6.9}$$

We suppose that we start from a given position x_0, thus adding a new initial constraint: $\boldsymbol{X}_0 = x_0$.

In (6.9), the future state $\boldsymbol{X}_{t+1}$ depends on the current state $\boldsymbol{X}_t$, the decision $\boldsymbol{U}_t$ taken at time t and the realization of the uncertainties $\boldsymbol{W}_{t+1}$ between time t and $t+1$ (that is, during the time interval $[t\Delta, (t+1)\Delta[$).

6.2.2.4. Bounds constraints

By Equations (4.10a) and (4.12), the stocks in the battery $\boldsymbol{B}_t$ and in the tank $\boldsymbol{H}_t$ are bounded. At time t, the control $\boldsymbol{F}^{b}_{t}$ must ensure that the next state $\boldsymbol{B}_{t+1}$ is admissible, that is, $\underline{b} \leq \boldsymbol{B}_{t+1} \leq \overline{b}$

by Equation (4.10a), which rewrites,

$$\underline{b} \leq \boldsymbol{B}_t + \Delta\big[\rho_c(\boldsymbol{F}_t^b)^+ + \frac{1}{\rho_d}(\boldsymbol{F}_t^b)^-\big] \leq \overline{b}\,. \tag{6.10}$$

Thus, the constraints on $\boldsymbol{F}_t^b$ depends on the stock $\boldsymbol{B}_t$. The same reasoning applies for the tank power $\boldsymbol{F}_t^h$. Furthermore, we set bound constraints on controls, that is,

$$-\overline{f}^b \leq \boldsymbol{F}_t^b \leq \overline{f}^b\ ,\ \ 0 \leq \boldsymbol{F}_t^h \leq \overline{f}_t^h\ ,\ \ 0 \leq \boldsymbol{F}_t^t \leq \overline{f}_t^t\,. \tag{6.11}$$

Finally the load-balance equation (6.5) also acts as a constraint on the controls. We gather all these constraints in an admissible set on control $\boldsymbol{U}_t$ depending on the current state $\boldsymbol{X}_t$:

$$\boldsymbol{U}_t \in \mathcal{U}_t^{ad}(\boldsymbol{X}_t) \qquad \mathbb{P}-\text{a.s.}\,. \tag{6.12}$$

6.2.2.5. Objective

At time t, the operational cost $L_t : \mathbb{X}_t \times \mathbb{U}_t \times \mathbb{W}_{t+1} \to \mathbb{R}$ aggregates two different costs:

$$L_t(x_t, u_t, w_{t+1}) = \pi_t^e \times f_{t+1}^{\text{ne}} + \pi_t^d \times \max(0, \overline{\theta_t^i} - \theta_t^{\text{i}})\,. \tag{6.13}$$

First, we pay a price π_t^e to import electricity from the network between time t and $t+1$. Hence, electricity cost is equal to $\pi_t^e \times \boldsymbol{F}_{t+1}^{\text{ne}}$. Second, if the indoor temperature is below a given threshold, we penalize the induced discomfort with a cost $\pi_t^d \times \max(0, \overline{\theta_t^i} - \boldsymbol{\theta}_t^{\text{i}})$, where π_t^d is a virtual price of discomfort. The cost L_t is a convex piecewise linear function, which will prove important for the SDDP algorithm in §6.3.2.

We add a final cost $K : \mathbb{X}_T \to \mathbb{R}$ to ensure that stocks are non empty at final time T

$$K(x_T) = \kappa \times \max(0, x_0 - x_T)\,, \tag{6.14}$$

where κ is a positive penalization coefficient, calibrated by trials and errors.

As decisions $\boldsymbol{U}_t$ and states $\boldsymbol{X}_t$ are random, the costs $L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1})$ become also random variables. We choose to minimize the expected value of the daily operational cost, yielding the criterion

$$\mathbb{E}\Big[\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) + K(\boldsymbol{X}_T)\Big]\,, \tag{6.15}$$

yielding an expected value of a convex piecewise linear cost. We suppose that measurability and integrability assumptions hold, so that the expected value in Equation (6.15) makes sense.

6.2.3. Stochastic optimal control formulation

Finally, the EMS problem translates to a generic stochastic optimal control (SOC) problem

$$\min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}) + K(\boldsymbol{X}_T)\Big]\,, \tag{6.16a}$$

$$\boldsymbol{X}_0 = x_0\,, \tag{6.16b}$$

$$\boldsymbol{X}_{t+1} = f_t\big(\boldsymbol{X}_t, \boldsymbol{U}_t, \boldsymbol{W}_{t+1}\big) \quad \mathbb{P}-\text{a.s.}\,, \tag{6.16c}$$

$$\boldsymbol{U}_t \in \mathcal{U}_t^{ad}(\boldsymbol{X}_t) \quad \mathbb{P}-\text{a.s.}\,, \tag{6.16d}$$

$$\sigma(\boldsymbol{U}_t) \subset \mathcal{F}_t\,. \tag{6.16e}$$

Problem (6.16) states that we want to minimize the expected value of the costs while satisfying the dynamics, the control bounds and the non-anticipativity constraints.

6.3. Resolution methods

The exact resolution of Problem (6.16) is out of reach in general. We propose two different algorithms that provide online policies $\pi_t : \mathbb{X}_0 \times \mathbb{W}_1 \times \cdots \times \mathbb{W}_t \to \mathbb{U}_t$ that map available information $x_0, w_1, \ldots, w_t$ at time t to a decision u_t. This section revisits Section 3.2 by adapting the policies to the current application, embodied by Problem (6.16). In particular, Model Predictive Control §6.3.1 is a lookahead policy (see Section 3.3), whereas Stochastic Dual Dynamic Programming §6.3.2 is a cost-to-go policy (see Section 3.4).

6.3.1. Model Predictive Control (MPC)

MPC is a classical algorithm commonly used to handle uncertainties in energy systems. Its procedure was described in Section 3.3. At time t, we regard the current state x_t and a deterministic forecast $(\overline{w}_{t+1}, \ldots, \overline{w}_T)$ of the future uncertainties $(\boldsymbol{W}_{t+1}, \ldots, \boldsymbol{W}_T)$ and we solve the deterministic problem

$$\min_{(u_t, \cdots, u_{T-1})} \sum_{j=t}^{T-1} \Big[L_j(x_j, u_j, \overline{w}_{j+1}) \Big] + K(x_T) \,, \tag{6.17a}$$

$$x_{j+1} = f_j\big(x_j, u_j, \overline{w}_{j+1}\big) \,, \tag{6.17b}$$

$$u_j \in \mathcal{U}_j^{ad}(x_j) \,. \tag{6.17c}$$

At time t, we solve Problem (6.17), retrieve the optimal decisions $(u_t^\sharp, \ldots, u_{T-1}^\sharp)$ and only keep the first decision $u_t^\sharp$ to control the system between time t and $t+1$. Then, we restart the procedure at time $t+1$.

As Problem (6.17) is linear and the number of time steps remains not too large, we are able to solve it exactly for every t.

6.3.2. Stochastic Dual Dynamic Programming (SDDP)

SDDP computes a set of value functions by solving a set of recursive Bellman equations. Then, we use these value functions to build a policy to take decisions online.

6.3.2.1. Dynamic Programming and Bellman principle

The Dynamic Programming method (Bellman, 1957) looks for solutions of Problem (6.16) as state-feedbacks $\pi_t : \mathbb{X}_t \to \mathbb{U}_t$. Dynamic Programming computes a serie of value functions backward in time by setting $V_T(x_T) = K(x_T)$ and solving the recursive equations

$$V_t(x_t) = \min_{u \in \mathcal{U}_t^{ad}(x_t)} \int_{\mathbb{W}_{t+1}} \big[L_t(x_t, u, w_{t+1}) + V_{t+1}\big(f_t(x_t, u, w_{t+1})\big) \big] \mu_{t+1}^{of}(dw_{t+1}) \,, \tag{6.18}$$

where $\mu_{t+1}^{of} \in \Delta(\mathbb{W}_{t+1})$ is an *offline* probability distribution on $\mathbb{W}_{t+1}$ (see §3.2.1.2).

Once these functions are computed, we compute a decision at time t as a state-feedback:

$$\pi_t(x_t) \in \operatorname*{arg\,min}_{u \in \mathcal{U}_t^{ad}(x_t)} \int_{\mathbb{W}_{t+1}} \big[L_t(x_t, u, w_{t+1}) + V_{t+1}\big(f_t(x_t, u, w_{t+1})\big) \big] \mu_{t+1}^{on}(dw_{t+1}) \,, \tag{6.19}$$

where $\mu_{t+1}^{on} \in \Delta(\mathbb{W}_{t+1})$ is an *online* probability distribution on $\mathbb{W}_{t+1}$. This method proves to be optimal for Problem (6.16) when the uncertainties $\boldsymbol{W}_1, \dots, \boldsymbol{W}_T$ are stagewise independent and when $\mu_t^{on} = \mu_t^{of}$ is the probability distribution of $\boldsymbol{W}_t$ in (6.18).

6.3.2.2. Description of Stochastic Dual Dynamic Programming

Dynamic Programming suffers from the well-known *curse of dimensionality* (Bertsekas, 2005a): its numerical resolution fails for state dimension typically greater than 4 when value functions are computed on a discretized grid. As the state $\boldsymbol{X}_t$ in §6.2.2.3 has 4 dimensions, SDP would be too slow to solve numerically Problem (6.16). Under proper assumptions, the Stochastic Dual Dynamic Programming (SDDP) can bypass the curse of dimensionality by approximating value functions by polyhedral functions. It is optimal for solving Problem (6.16) when uncertainties are stagewise independent, costs L_t and K are convex and dynamics f_t are linear (Girardeau et al., 2014).

SDDP provides an outer approximation $\underline{V}_t^k$ of the value function V_t in (6.18) with a set of supporting hyperplanes $\{(\lambda_t^j, \beta_t^j)\}_{j=1,\cdots,k}$ by

$$\underline{V}_t(x_t) = \min_{\theta_t \in \mathbb{R}} \theta_t \ , \tag{6.20a}$$

$$\left\langle \lambda_t^j \, , x_t \right\rangle + \beta_t^j \leq \theta_t \ , \ \ \forall j = 1, \cdots, k \ . \tag{6.20b}$$

We now describe the SDDP algorithm in our formalism. Each iteration k of SDDP encompasses two passes.

- During the *forward pass*, we draw a scenario $w_1^k, \dots, w_T^k$ of uncertainties, and compute a state trajectory $\left\{x_t^k\right\}_{t=0\cdots T}$ along this scenario. Starting from position x_0, we compute x_{t+1}^k in an iterative fashion: i) we compute the optimal control at time t using the available $\underline{V}_{t+1}^k$ function

$$u_t^k \in \underset{u \in \mathcal{U}_t^{ad}(x_t)}{\arg\min} \int_{\mathbb{W}_{t+1}} \left[L_t(x_t^k, u, w_{t+1}) + \underline{V}_{t+1}^k\big(f_t(x_t^k, u, w_{t+1})\big) \right] \mu_{t+1}^{of}(dw_{t+1}) \ , \tag{6.21}$$

  and ii), we set $x_{t+1}^k = f_t(x_t^k, u_t^k, w_{t+1}^k)$ where f_t is given by (6.8).

- During the *backward pass*, we update the approximated value functions $\left\{\underline{V}_t^k\right\}_{t=0,\cdots,T}$ backward in time along the trajectory $\left\{x_t^k\right\}_{t=0,\cdots,T}$. At time t, we solve the problem

$$\theta_t^{k+1} = \min_{u \in \mathcal{U}_t^{ad}(x_t)} \int_{\mathbb{W}_{t+1}} \left[L_t(x_t^k, u, w_{t+1}) + \underline{V}_{t+1}^{k+1}\big(f_t(x_t^k, u, w_{t+1})\big) \right] \mu_{t+1}^{of}(dw_{t+1}) \ , \tag{6.22}$$

  and we obtain a new cut $(\lambda_t^{k+1}, \beta_t^{k+1})$ where λ_t^{k+1} is a subgradient of optimal cost (6.22) evaluated at point $x_t = x_t^k$ and $\beta_t^{k+1} = \theta_t^{k+1} - \left\langle \lambda_t^{k+1} \, , x_t^k \right\rangle$. This new cut allows to update the function $\underline{V}_t^{k+1}$: $V_t^{k+1} = \max\{\underline{V}_t^k, \left\langle \lambda_t^{k+1} \, , . \right\rangle + \beta_t^{k+1}\}$.

Otherwise stated, SDDP only explores the state space around "interesting" trajectories (those computed during the forward passes) and refines the value functions only in the corresponding space regions (backward passes).

6.3.2.3. Obtaining online controls with SDDP

In order to compute implementable decisions, we use the following procedure.

- The approximated value functions $\left\{\underline{V}_t\right\}_{t \in \{0,..,T\}}$ are computed with the SDDP algorithm described in §6.3.2.2. These computations are done *offline*.

- The approximated value functions $\{\underline{V}_t\}_{t\in\{0,..,T\}}$ are then used to compute *online* a decision at any time t for any state x_t by using Equation (6.19).

We obtain a cost-to-go based policy, as introduced in Section 3.4. More precisely, we compute the SDDP policy π_t^{sddp} by

$$\pi_t^{\text{sddp}}(x_t) \in \underset{u\in\mathcal{U}_t^{ad}(x_t)}{\arg\min} \int_{\mathbb{W}_{t+1}} \big[L_t(x_t,u,w_{t+1}) + \underline{V}_{t+1}\big(f_t(x_t,u,w_{t+1})\big)\big]\mu_{t+1}^{on}(dw_{t+1}) \,, \tag{6.23}$$

which corresponds to replacing the value function V_{t+1} in Equation (6.19) with its approximation $\underline{V}_{t+1}$. The decision $\pi_t^{\text{sddp}}(x_t)$ is used to control the system between time t and $t+1$. Then, we resolve Problem (6.23) at time $t+1$.

To solve numerically Problem (6.23) at time t, we will consider online distributions with finite support $w_t^1,\dots,w_t^S$. The online distribution μ_{t+1}^{on} in Equation (6.23) now writes: $\mu_{t+1}^{on} = \sum_{s=1}^S p_s\delta_{w_{t+1}^s}$ where $\delta_{w_{t+1}^s}$ is the Dirac measure at w_{t+1}^s and $(p_1,\dots,p_S)$ are probability weights. The same reasoning applies to the online distribution μ_{t+1}^{on}. For instance, Problem (6.23) reformulates as

$$\pi_t^{\text{sddp}}(x_t) \in \underset{u\in\mathcal{U}_t^{ad}(x_t)}{\arg\min} \sum_{s=1}^S p_s\big[L_t(x_t,u,w_{t+1}^s) + \underline{V}_{t+1}\big(f_t(x_t,u,w_{t+1}^s)\big)\big] \,. \tag{6.24}$$

6.4. Numerical resolution

We apply the two algorithms described in §6.3 to a specific case study.

6.4.1. Case study

We consider a building equipped with solar panels, a electrical hot water tank and a battery.

6.4.1.1. Settings

We aim to solve the stochastic optimization problem (6.16) over one day, with 96 time steps. The battery's size is 3 kWh, and the hot water tank has a capacity of 120 l. We suppose that the house has a surface $A_p = 20$ m^2 of solar panel at disposal, oriented south, and with a yield of 15%. We penalize the recourse variable $\mathbf{F}_{t+1}^{\text{ne}}$ in (6.13) with on-peak and off-peak tariff, corresponding to Électricité de France's (EDF) individual tariffs. The building's thermal envelope corresponds to the French RT2012 specifications (Journal Officiel, 2013). Meteorological data comes from Meteo France measurements corresponding to the year 2015.

6.4.1.2. Demands scenarios

We have scenarios of electrical and domestic hot water demands at 15 minutes time steps, obtained with StRoBe (Baetens and Saelens, 2016). Figure 6.2 displays 100 scenarios of electrical and hot water demands. We observe that the shape of these scenarios is consistent: demands are almost null during night, and there are peaks around midday and 8 pm. Peaks in hot water demands corresponds to showers. We aggregate the production of the solar panel $\mathbf{\Phi}^{\text{pv}}$ and the electrical demands $\mathbf{D}^{\text{el}}$ in a single variable $\mathbf{D}^{\text{el}}$ to consider only two uncertainties $(\mathbf{D}_t^{\text{el}}, \boldsymbol{D}_t^{\text{hw}})$.

6.4.1.3. Out of sample assessment of strategies

To obtain a fair comparison between SDDP and MPC, we use an out-of-sample validation following the procedure introduced in Section 3.5. We generate 2,000 scenarios of electrical and hot water

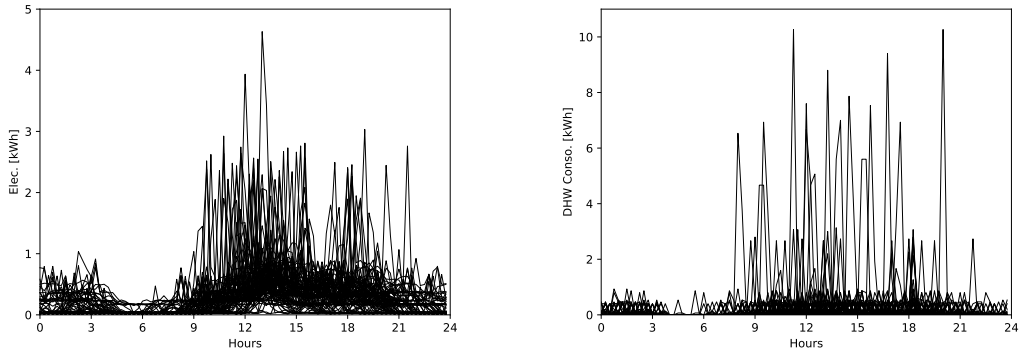


Figure 6.2.: Electrical (left) and domestic hot water (right) demand scenarios.

demands, and we split these scenarios in two distinct parts: the first N_{opt} scenarios are called *optimization scenarios*, and the remaining N_{sim} scenarios are called *assessment scenarios*. We made the choise $N_{opt} = N_{sim} = 1,000$.

First, during the offline phase, we use the optimization scenarios to build models for the uncertainties, under the mathematical form required by each algorithm. Second, during the online phase, we use the assessment scenarios to compare the strategies produced by these algorithms. At time t during the assessment, the algorithms obviously cannot use the future values of the assessment scenarios, but can take advantage of the observed values up to t to update their statistical models of future uncertainties.

6.4.2. Numerical implementation

We detail hereafter the numerical implementation of MPC, SDDP and the heuristics.

6.4.2.1. Implementing the algorithms

We implement MPC and SDDP in Julia 0.6, using JuMP (Dunning et al., 2017) as a modeler, `StochDynamicProgramming.jl` as a SDDP solver, and Gurobi 7.02 (Gurobi Optimization Inc, 2014) as a LP solver. All computations run on a Core i7 2.5 GHz processor, with 16 GB RAM.

6.4.2.2. MPC procedure

Electrical and thermal demands are naturally correlated in time (Widén and Wäckelgård, 2010). To take into account such a dependence across the different time-steps, we chose to model the process $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_T$ with an auto-regressive (AR) process.

Building offline an AR model for MPC. The procedure to fit the parameters of the AR is as follows.

We fit an AR(1) model upon the optimization scenarios (we do not consider higher order lag for the sake of simplicity). For $i \in \{el, hw\}$, the AR model writes

$$d^i_{t+1} = \alpha^i_t d^i_t + \beta^i_t + \varepsilon^i_t \,, \tag{6.25a}$$

where the non-stationary coefficients (α_t^i, β_t^i) are, for all time t, solutions of the least-square problem

$$(\alpha_t^i, \beta_t^i) = \arg\min_{a,b} \sum_{s=1}^{N_{opt}} \left\| d_{t+1}^{i,s} - a d_t^{i,s} - b \right\|_2^2 . \tag{6.25b}$$

The points $(d_t^{i,1}, \dots d_t^{i,N_{opt}})$ correspond to the optimization scenarios. The AR residuals $(\varepsilon_t^{\text{el}}, \varepsilon_t^{\text{hw}})$ are assumed to be a white noise process.

Updating the forecast online. Once the AR model is calibrated, we use it to update the forecast during assessment (see §6.3.1). The update procedure is threefold:

i) we observe the demands $w_t = (d_t^{\text{el}}, d_t^{\text{hw}})$ between time $t-1$ and t,

ii) we update the forecast $\overline{w}_{t+1}$ at time $t+1$ with the AR model

$$\overline{w}_{t+1} = \left(\overline{d}_{t+1}^{\text{el}}, \overline{d}_{t+1}^{\text{hw}}\right) = \left(\alpha_t^{\text{el}} d_t^{\text{el}} + \beta_t^{\text{el}},\ \alpha_t^{\text{hw}} d_t^{\text{hw}} + \beta_t^{\text{hw}}\right) ,$$

iii) we set the forecast between time $t+2$ and T by using the average values of the optimization scenarios:

$$\overline{w}_\tau = \frac{1}{N^{opt}} \sum_{i=1}^{N^{opt}} w_\tau^i \quad \forall \tau = t+2, \cdots, T .$$

Once the forecast $(\overline{w}_{t+1}, \dots, \overline{w}_T)$ is available, it is fed into the MPC algorithm to solve Problem (6.17).

6.4.2.3. SDDP procedure

Even if electrical and thermal demands are naturally correlated in time (Widén and Wäckelgård, 2010), the SDDP algorithm only relies upon marginal distributions.

Building offline probability distributions for SDDP. Rather than fitting an AR model like done with MPC, we use the optimization scenarios to build marginal probability distributions μ_t^{of} that will feed the SDDP procedure in (6.18). We follow the procedure introduced in §4.4 to quantize the optimization scenarios in S representative points. We choose here $S = 20$ to have enough precision.

Computing value functions offline. Then, we use these probability distributions as an input to compute a set of value functions with the procedure described in §6.3.2.2.

Using the value functions online. Once the value functions have been computed by SDDP, we are able to compute online decisions with Equation (6.24) In practice, the quantization size of μ_t^{on} is larger than those of μ_t^{of}, to have a greater accuracy online (we discuss further the impact of the quantization size of μ_t^{on} on the performance of SDDP policy in Remark 6.4.1). SDDP, on the contrary of MPC, does not update the online probability distribution μ_t^{on} during assessment to consider the information brought by the previous observations.

6.4.2.4. Heuristic procedure

We choose to compare the MPC and SDDP algorithms with a basic decision rule. This heuristic is as follows: the battery is charged whenever the solar production $\mathbf{\Phi}^{\text{pv}}$ is available, and discharged to fulfill the demand if there remains enough energy in the battery; the tank is charged ($\boldsymbol{F}_t^h > 0$)

if the tank energy $\boldsymbol{H}_t$ is lower than $\boldsymbol{H}_0$, the heater $\boldsymbol{F}_t^t$ is switched on when the temperature is below the setpoint $\overline{\theta_t^i}$ and switched off whenever the temperature is above the setpoint plus a given margin.

6.4.3. Results

We compare the policies on three different days, corresponding to different meteorological conditions.

6.4.3.1. Assessing on different meteorological conditions

We assess the algorithms on three different days, with different meteorological conditions (see Table 6.1). Therefore, we use three distinct sets of N_{sim} assessment scenarios of demands, one for each typical day.

	Date	Temp. ($°C$)	PV Production (kWh)
Winter Day	February, 19th	3.3	8.4
Spring Day	April, 1st	10.1	14.8
Summer Day	May, 31st	14.1	23.3

Table 6.1.: Different meteorological conditions

These three different days corresponds to different heating needs. During *Winter day*, the heating is maximal, whereas it is medium during *Spring day* and null during *Summer day*. The production of the solar panel varies accordingly.

6.4.3.2. Comparing the algorithms performances

During assessment, we use MPC (see (6.17)) and SDDP (see (6.23)) strategies to compute online decisions along N_{sim} assessment scenarios. Then, we compare the average electricity bill obtained with these two strategies and with the heuristic. The assessment results are given in Table 6.2: means and standard deviation σ are computed by Monte Carlo with the N_{sim} assessment scenarios; the notation $\pm$ corresponds to the interval $\pm 1.96\frac{\sigma}{\sqrt{N_{sim}}}$, which is a 95% confidence interval.

	SDDP	MPC	Heuristic
Offline time	50 s	0 s	0 s
Online time	1.5 ms	0.5 ms	0.005 ms
Electricity bill (€)			
Winter day	4.38 ± 0.02	4.59 ± 0.02	5.55 ± 0.02
Spring day	1.46 ± 0.01	1.45 ± 0.01	2.83 ± 0.01
Summer day	0.10 ± 0.01	0.18 ± 0.01	0.33 ± 0.02

Table 6.2.: Comparison of MPC, SDDP and heuristic strategies

We observe in Table 6.2 that MPC and SDDP exhibit close performance, and do better than the heuristic. If we consider mean electricity bills, SDDP achieves better savings than MPC during *Summer day* and *Winter day*, but SDDP and MPC display similar performances during *Spring day*.

In addition, SDDP achieves better savings than MPC for the vast majority of scenarios. Indeed, if we compare the difference between the electricity bills scenario by scenario, we observe that

SDDP is better than MPC for about 93% of the scenarios. This can be seen on Figure 6.3 that displays the histogram of the absolute gap savings between SDDP and MPC during *Summer day*. The distribution of the gap exhibits a heavy tail that favors SDDP on extreme scenarios. Similar analyses hold for *Winter* and *Spring day*. Thus, we claim that SDDP outperforms MPC for the electricity savings. Concerning the performance on thermal comfort, temperature trajectories are above the temperature setpoints specified in §6.2.2.5 for both MPC and SDDP.

In term of numerical performance, it takes less than a minute to compute a set of cuts as in §6.3.2.2 with SDDP on a particular day. Then, the online computation of a single decision takes 1.5 ms on average, compared to 0.5 ms for MPC. Indeed, MPC is favored by the linearity of the optimization Problem (6.17), whereas, for SDDP, the higher the quantization size S, the slowest is the resolution of Problem (6.23), but the more information the online probability distribution μ_t^{on} carries. We discuss further the impact of the quantization size in Remark 6.4.1.

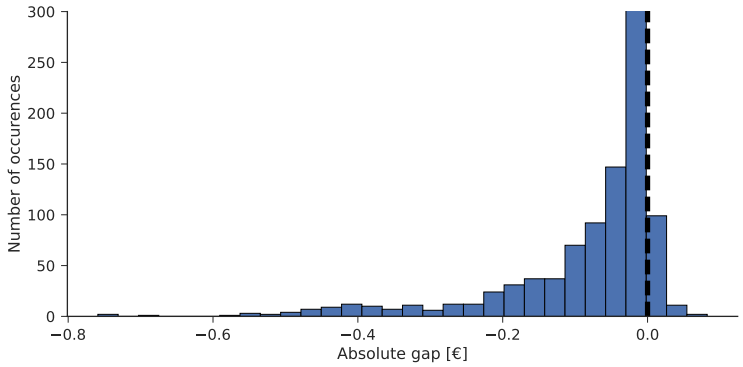
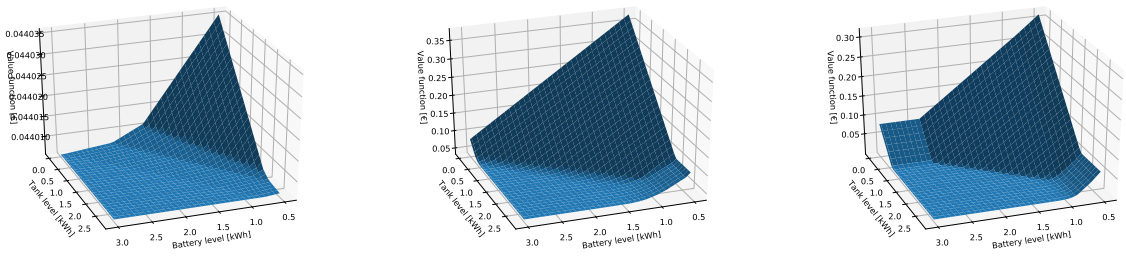


Figure 6.3.: Absolute gap savings between MPC and SDDP during *Summer day*

6.4.3.3. Analyzing the value functions

We now analyse the value functions after convergence. Figure 6.4 displays the value functions at different moment of the day for *Summer day*. We observe that if both the level of the battery or the hot water tank are sufficiently above their lower bound, the value function is equal to 0.



8am 12pm 9pm

Figure 6.4.: Approximated value functions at 8am, 12pm and 21pm during *Summer day*.

6.4.3.4. Analyzing the trajectories

We analyze now the trajectories of stocks in assessment, during *Summer day*. The heating is off, and the production of the solar panel is nominal at midday.

Figure 6.5 displays the state of charge of the battery along a subset of assessment scenarios, for SDDP and MPC. We observe that SDDP charges earlier the battery at its maximum. On the contrary MPC charges the battery later, and does not use the full potential of the battery. The two algorithms discharge the battery to fulfill the evening demands. We notice that each trajectory exhibits a single cycle of charge/discharge, thus decreasing battery's aging.

Figure 6.6 displays the charge of the domestic hot water tank along the same subset of assessment scenarios. We observe a similar behavior as for the battery trajectories: SDDP uses more the electrical hot water tank to store the excess of PV energy, and the level of the tank is greater at the end of the day than in MPC.

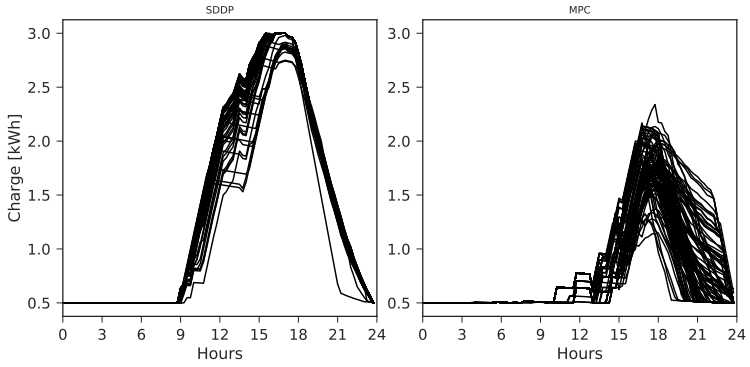


Figure 6.5.: Battery charge trajectories for SDDP and MPC during *Summer day*

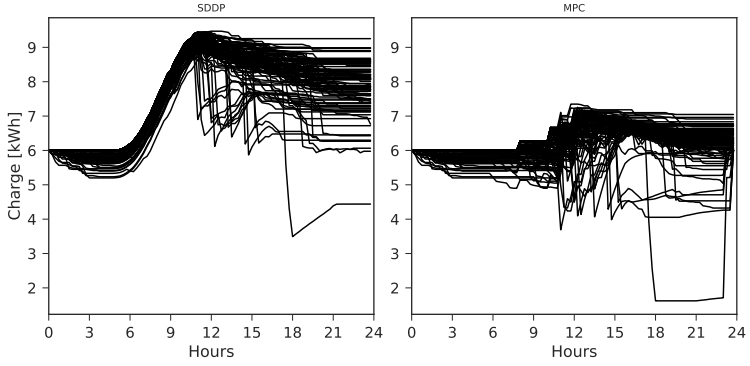


Figure 6.6.: Hot water tank trajectories for SDDP and MPC during *Summer day*

This analysis suggests that SDDP makes a better use of storage capacities than MPC.

Remark 6.4.1. *SDDP policy* (6.24) *requires a discrete probability distribution* μ_{t+1}^{on} *defined on* $\mathbb{W}_{t+1}$ *to compute a decision* u_t *at time* t*. However, the higher the quantization size, the larger the number of constraints are in Problem* (6.24)*, but the richer is the representation of the uncertainty. The decision-maker has to find a trade-off between the computation time and the performance of the SDDP policy. We consider the problem corresponding to the 1st April (see Table 6.1), and*

assess SDDP policy with different quantization size S. Results are given in Figure 6.7. We observe that for a quantization size bigger than 7, the results are almost similar, leaving apart the statistical noise.

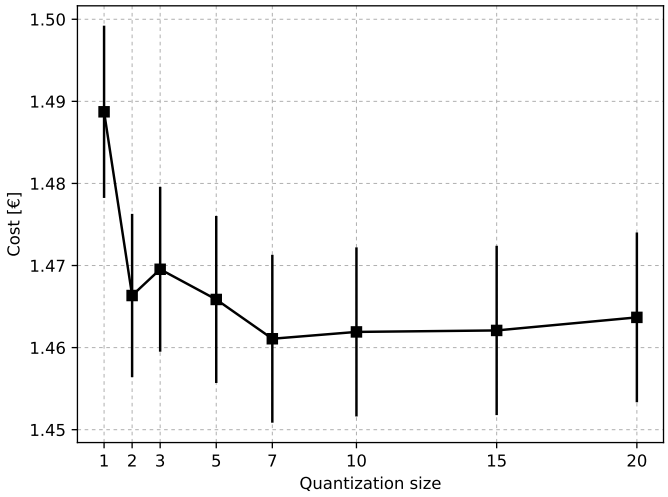


Figure 6.7.: Evolution of assessment costs against quantization size S. The bar indicates the confidence interval of the average cost along the 1,000 assessment scenarios.

◊

6.5. Discussion

We have presented a domestic microgrid energy system, and compared different optimization algorithms to control the stocks with an Energy Management System.

The results show that optimization based strategies outperform the proposed heuristic procedure in term of money savings. Furthermore, SDDP outperforms MPC during *Winter* and *Summer day* — achieving up to 35% costs savings — and displays similar performance as MPC during *Spring day*. Even if SDDP and MPC exhibit close performance, a comparison scenario by scenario shows that SDDP beats MPC most of the time (more than 90% of scenarios during *Summer day*). Thus, we claim that SDDP is better than MPC to manage uncertainties in such a microgrid, although MPC gives also good performance. SDDP also makes a better use of storage capacities.

Our study can be extended in different directions. First, we could mix SDDP and MPC to recover the benefits of these two algorithms. Indeed, SDDP is designed to handles the uncertainties' variability but fails to capture the time correlation, whereas MPC ignores the uncertainties' variability, but considers time correlation by means of a multistage optimization problem. Second, we will study in Chapter 8 the optimization of larger scale microgrids — with different interconnected buildings — by decomposition methods.

Part II.

Mixing time and spatial decomposition in large-scale optimization problems

Abstract. In Part I, we have applied optimization methods and have designed online policies to manage small scale systems, that is, with a limited number of stocks. We have proved the relevance of optimization methods to frame online policies. We have sorted existing algorithms in two classes: the first class represents future uncertainties as scenarios, the other by a set of value functions representing the average cost-to-go to start from a given position. The computation of such cost-to-go relies on Dynamic Programming. However, the resolution of Dynamic Programming equations is confronted by the well-known curse of dimensionality that limits the number of stocks we are able to deal with.
We introduce hereafter spatial decomposition methods in the perspective to overcome the curse of dimensionality. We handle large scale systems gathering small scale units coupled together via a set of coupling constraints. The decomposition breaks apart the coupling constraints to obtain decoupled small scale units, depending on global coordination variables. Once subproblems decoupled, we are able to solve them locally and in parallel by Dynamic Programming by handling only the local stocks and the coordination variables.
We study in Chapter 7 a generic stochastic multistage problem with different subsystems coupled together via a set of coupling constraints, and introduce price and resource decomposition schemes. We prove that we are able to bound the global Bellman functions above by the sum of local resource-decomposed value functions, and below by the sum of local price-decomposed value functions. In Chapter 8, we apply the price and resource decomposition schemes to a problem where the coupling constraints are set on a graph. We analyze a case study consisting of a large scale district microgrid coupling up to 48 small scale units together. In Chapter 9 we show that this study corroborates with another study focusing on the management of hydro-dams valleys instead of microgrids. This chapter is a joint work with Pierre Carpentier, Jean-Philippe Chancelier and Vincent Leclère, and has led to a publication (Carpentier et al., 2018b).

Chapter 7.

Upper and lower bounds for Bellman functions by spatial decomposition

Contents

7.1. Introduction

We consider a multistage stochastic optimization problem where different units are connected together via a coupling constraint. Each unit is a (small) control system. Static constraints couple all units at each time. We propose two decomposition methods, whether decoupling the coupling constraints by prices or by resource. We show that the global Bellman function can be bounded above by a sum of local resource-decomposed value functions, and below by a sum of local price-decomposed value functions. We provide conditions under which these local value functions can be computed by Dynamic Programming.

This chapter is part of a larger work aiming at mixing spatial decomposition methods with stochastic optimization. We refer to (Carpentier and Cohen, 2017) for a generic description of decomposition methods. Recent developments allow to mix spatial decomposition with Dynamic Programming to solve effectively large scale multistage stochastic optimization problems. This work lead to the introduction of the Dual Approximate Dynamic Programming algorithm (Barty et al., 2010a). This algorithm was applied to problems with a single central coupling constraint (Girardeau, 2010) or on coupling cascade constraints, as in hydro-dams valley where the flows of an

upstream dam enters the next downstream dam downstream (Alais, 2013). Theoretical insights concerning stochastic decomposition methods were given in Leclère (2014). We extend in this chapter the previous contributions by tackling more generic coupling constraints where a mix of local functions must belong to a given global admissible set. This framework takes inspiration from the extended monotropic approaches introduced in Bertsekas (2008).

We focus in Section 7.2 on a generic decomposable optimization problem and present the price and resource decomposition schemes. We then apply the two decomposition methods to a multistage stochastic optimization problem by decomposing a global coupling constraint time step by time step with a price and with a resource processes. Then, in Section 7.3 we provide conditions under which the decomposed local price and resource value functions are computable by Dynamic Programming. We thus obtain a mix of spatial and time decompositions. Under proper assumptions, the sum of the local value functions computed by Dynamic Programming bounds the global value functions of the original problem. Eventually, we detail in Section 7.4 methods to obtain tighter bounds by choosing appropriately the price and resource processes among a given design.

7.2. Bounds for an optimization problem under coupling constraints via decomposition

We first introduce a generic unit optimization problem under coupling constraints in §7.2.1 and show in §7.2.2 that, by decomposition, we are able to bound the optimal solution of the generic problem. Then, we focus in §7.2.3 on the special case of multistage stochastic optimization problems.

7.2.1. Global optimization problem formulation from local data

We describe a generic optimization problem coupling different local subproblems. We borrow the abstract duality formalism of Rockafellar (1974), which allows to write duality conditions in a generic setting.

Let $\mathcal{Z}^1, \ldots, \mathcal{Z}^N$ be N sets and

$$J^i : \mathcal{Z}^i \to (-\infty, +\infty] \ , \ \ i \in \{1, .., N\} \ , \tag{7.1}$$

be a local criteria. Let $\Xi^1, \ldots, \Xi^N$ be N vector spaces paired with $(\Xi^1)^\star, \ldots, (\Xi^N)^\star$ by duality pairings

$$\begin{aligned} \left\langle \cdot \, , \cdot \right\rangle : \quad \Xi^i \times (\Xi^i)^\star &\to \mathbb{R} \\ (r^i, \lambda^i) &\mapsto \left\langle \lambda^i \, , r^i \right\rangle . \end{aligned} \tag{7.2}$$

We consider N mappings

$$\vartheta^i : \mathcal{Z}^i \to \Xi^i \ , \ \ i \in \{1, .., N\} \ . \tag{7.3}$$

From these *local* data, we formulate a *global* minimization problem under constraints as follows. We introduce the product set

$$\mathcal{Z} = \mathcal{Z}^1 \times \cdots \times \mathcal{Z}^N \ , \tag{7.4}$$

and the product spaces

$$\Xi = \Xi^1 \times \cdots \times \Xi^N \ , \ \ \Xi^\star = (\Xi^1)^\star \times \cdots \times (\Xi^N)^\star \ , \tag{7.5}$$

Let

$$C \subset \Xi \ , \tag{7.6}$$

be a set included in Ξ that will capture the coupling constraints between the N units.

We define the *global optimization* problem as

$$V^\sharp = \inf_{(z^1,\cdots,z^N)\in\mathcal{Z}} \sum_{i=1}^N J^i(z^i) \;, \tag{7.7a}$$

under the *global coupling constraint*

$$\big(\vartheta^1(z^1),\cdots,\vartheta^N(z^N)\big) \in -C \;. \tag{7.7b}$$

We note that without Constraint (7.7b), Problem (7.7) would decompose into N independent subproblems in a straightforward manner.

7.2.2. Local price and resource value functions. Upper and lower bounds

Let us introduce *local price value functions* $\underline{V}^i : (\Xi^i)^\star \to \overline{\mathbb{R}}$ by

$$\underline{V}^i[\lambda^i] = \inf_{z^i\in\mathcal{Z}^i} \; J^i(z^i) + \big\langle \lambda^i \,, \vartheta^i(z^i)\big\rangle \;, \qquad \forall \lambda^i \in (\Xi^i)^\star \;, \tag{7.8}$$

and *local resource value functions* $\overline{V}^i : \Xi^i \to \overline{\mathbb{R}}$ by

$$\begin{aligned} \overline{V}^i[r^i] = \inf_{z^i} \; & J^i(z^i) \;, \qquad \forall r^i \in \Xi^i \\ & \text{s.t. } \vartheta^i(z^i) = r^i \;. \end{aligned} \tag{7.9}$$

We denote by

$$C^\star = \big\{ \lambda \in \Xi^\star \mid \big\langle \lambda \,, r\big\rangle \geq 0 \;,\;\; \forall r \in C \big\} \;, \tag{7.10}$$

the dual cone of C.

Proposition 7.2.1. *For all $\lambda = (\lambda^1,\cdots,\lambda^N) \in C^\star$ and $r = (r^1,\cdots,r^N) \in -C$, we have the following upper and lower estimates of the global minimum of Problem* (7.7)*:*

$$\sum_{i=1}^N \underline{V}^i[\lambda^i] \leq V^\sharp \leq \sum_{i=1}^N \overline{V}^i[r^i] \;. \tag{7.11}$$

We call $\lambda \in C^\star$ an admissible price vector *and $r \in -C$ an* admissible resource vector*.*

Proof. The dual function associated to Problem (7.7) writes

$$\underline{V}[\lambda] = \inf_{z\in\mathcal{Z}} \sum_{i=1}^N J^i(z^i) + \big\langle \lambda^i \,, \vartheta^i(z^i)\big\rangle \;,\;\; \forall \lambda = (\lambda^1,\cdots,\lambda^N) \in C^\star \;. \tag{7.12}$$

For all $\lambda \in C^\star$, $r \in -C$, we have $\big\langle \lambda \,, r\big\rangle \leq 0$ by definition of $C^\star$ in (7.10), thus we deduce that $\sum_{i=1}^N \big\langle \lambda^i \,, \vartheta^i(z^i)\big\rangle \leq 0$ for all $z = (z^1,\cdots,z^N) \in \mathcal{Z}$ such that $\big(\vartheta^1(z^1),\cdots,\vartheta^N(z^N)\big) \in -C$. Hence, we have $V[\lambda] \leq \sum_{i=1}^N J^i(z^i)$ for all $z \in \mathcal{Z}$ satisfying (7.7b). By taking the inf in the right hand side, and as (7.12) and (7.8) are related by $\underline{V}[\lambda] = \sum_{i=1}^N \underline{V}^i[\lambda^i]$, we obtain the lower bound in (7.11). The upper bound arises directly, as Problem (7.7) is equivalent to $V^\sharp = \inf_{r\in-C} \sum_{i=1}^N \overline{V}^i[r^i]$ we obtain the upper bound in (7.11). □

Restricting the search spaces for bounds. It may prove useful to restrict the admissible set C in Problem (7.7) to ease the resolutions of the price (7.8) and resource value functions (7.9). Thus, we are prone to replace the admissible set C by a smaller set $\underline{C} \subset C$ to ease the resolution of the

primal value function (7.9) in the primal and by a smaller cone $\overline{C}^\star \subset C^\star$ to ease the resolution of the dual value function (7.8). By doing so, we prove that the bounds in (7.11) are preserved.

For all $C \subset \Xi$, we adopt the notation

$$V_C^\sharp = \inf_{(z^1,\cdots,z^N)\in\mathcal{Z}} \sum_{i=1}^N J^i(z^i) \tag{7.13a}$$

$$\left(\vartheta^1(z^1),\cdots,\vartheta^N(z^N)\right) \in -C\ . \tag{7.13b}$$

Let $\underline{C}$ and $\overline{C}$ be two sets such that

$$\underline{C} \subset C \subset \overline{C}\ . \tag{7.14}$$

Then, the following inequalities holds

$$\max_{\lambda\in\overline{C}^\star} \sum_{i=1}^N \underline{V}^i[\lambda^i] \leq V_{\overline{C}}^\sharp \leq V_C^\sharp \leq V_{\underline{C}}^\sharp \leq \inf_{r\in-\underline{C}} \sum_{i=1}^N \overline{V}^i[r^i]\ , \tag{7.15}$$

where the first inequality holds as $\max_{\lambda\in\overline{C}^\star} \sum_{i=1}^N \underline{V}^i[\lambda^i]$ is the dual problem of $V_{\overline{C}}^\sharp$. By restricting Problem $V_C^\sharp$, we obtain a looser bound in the primal, and by relaxing Problem $V_C^\sharp$, we obtain a looser bound in the dual, as illustrated in Figure 7.1.

$$\begin{array}{ccccc} \underline{C} & \subset & C & \subset & \overline{C} \\ & & \vdots & & \\ \overline{C}^\star & \subset & C^\star & \subset & \underline{C}^\star \end{array}$$

Figure 7.1.: Restricting the search spaces in the primal and in the dual

7.2.3. The special case of multistage stochastic optimization problem

Now, we turn to the case where the global optimization problem (7.7) is a multistage stochastic optimization problem elaborated from local data (local state and control spaces) with global coupling constraints at each time step.

We consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, a time span $\{0,\ldots,T\}$ — where $T \in \mathbb{N}^\star$ is a finite horizon — and a number $N \in \mathbb{N}^\star$ of local stochastic control problems.

7.2.3.1. Local data for local stochastic control problems

We detail hereafter the *local* data describing local stochastic control problem, for $i \in \{1,..,N\}$.

Local state, control and uncertainty spaces. For any time $t \in \{0,..,T-1\}$, let $(\mathbb{W}_t^i, \mathcal{W}_t^i)$, $(\mathbb{X}_t^i, \mathcal{X}_t^i)$ and $(\mathbb{U}_t^i, \mathcal{U}_t^i)$ be measurable spaces, and $(\Xi_t^i, \mathcal{T}^i)$ be an Euclidian space, equipped with the usual Euclidian scalar product $\lambda^i \cdot r^i$ for all $\lambda^i \in (\Xi_t^i)^\star$ and $r^i \in \Xi_t^i$.

Local dynamics. Let

$$g_t^i : \mathbb{X}_t^i \times \mathbb{U}_t^i \times \mathbb{W}_{t+1}^i \to \mathbb{X}_{t+1}^i\ ,\ \ \forall t \in \{0,..,T-1\}\ ,\ \ \forall i \in \{1,..,N\}\ , \tag{7.16}$$

be measurable *local dynamics*.

Local coupling. Let

$$\Theta_t^i : \mathbb{X}_t^i \times \mathbb{U}_t^i \to \Xi_t^i \ , \ \ \forall t \in \{0, .., T-1\} \ , \ \ \forall i \in \{1, .., N\} \ , \tag{7.17}$$

be measurable *local coupling* functions.

Local criteria. Let

$$L_t^i : \mathbb{X}_t^i \times \mathbb{U}_t^i \times \mathbb{W}_{t+1}^i \to \overline{\mathbb{R}} \ , \ \ \forall t \in \{0, .., T-1\} \ , \ \ \forall i \in \{1, .., N\} \tag{7.18}$$

be the *local instantaneous cost* and

$$K^i : \mathbb{X}_T^i \to \overline{\mathbb{R}} \ , \ \ \forall i \in \{1, .., N\} \tag{7.19}$$

be the *local final cost*.

We incorporate possible constraints coupling the control with the state directly in the instantaneous costs L_t^i, since they are extended-real valued functions which can possibly take the value $+\infty$.

Local uncertainties. Let

$$\boldsymbol{W}^i = \{\boldsymbol{W}_t^i\}_{t \in \{0,..,T\}} \ , \ \ \boldsymbol{W}_t^i : \Omega \to \mathbb{W}_t^i \ , \tag{7.20}$$

be the *local noise process*, defined on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

7.2.3.2. Global data for the global stochastic control problem

From local data given above, we are going to define the global data.

Global state and control spaces. We define the global measurable spaces as products (over units) of the local measurable spaces:

$$\mathbb{X}_t = \prod_{i=1}^N \mathbb{X}_t^i \ , \qquad \mathbb{U}_t = \prod_{i=1}^N \mathbb{U}_t^i \ , \ \ \forall t \in \{0, .., T-1\} \ . \tag{7.21}$$

Global coupling constraints. Let

$$\Xi_t = \prod_{i=1}^N \Xi_t^i \ , \ \ S_t \subset \Xi_t \ , \ \ \forall t \in \{0, .., T-1\} \ , \tag{7.22}$$

be the *global coupling spaces* Ξ_t and the global coupling sets S_t. We define also

$$\Xi = \prod_{t=0}^{T-1} \Xi_t \ , \ \ S = \prod_{t=0}^{T-1} S_t \ . \tag{7.23}$$

The global coupling constraints writes as a combination of the local couplings Θ_t^i introduced in (7.17), such that for all $t \in \{0, .., T-1\}$,

$$\big(\Theta_t^1(x_t^1, u_t^1), \cdots, \Theta_t^N(x_t^N, u_t^N)\big) \in -S_t \ , \ \ \forall (x_t^1, \cdots, x_t^N) \in \mathbb{X}_t \ , \ \ \forall (u_t^1, \cdots, u_t^N) \in \mathbb{U}_t \ . \tag{7.24}$$

Global uncertainties. We note $(\boldsymbol{W}_1, \cdots \boldsymbol{W}_T)$ the *global noise process*, where

$$\boldsymbol{W}_t = (\boldsymbol{W}_t^1, \cdots, \boldsymbol{W}_t^N) \ , \ \ \forall t \in \{1, .., T\} \ . \tag{7.25}$$

For all $t \in \{1, .., T-1\}$, $\boldsymbol{W}_t$ takes values in the measurable space

$$\mathbb{W}_t = \prod_{i=1}^{N} \mathbb{W}_t^i \ . \tag{7.26}$$

Global information structure. We will only consider two possible information structures.

- The *global information* structure captures the information provided by all uncertainties:

$$\mathcal{F}_t = \sigma(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_t) \ , \ \ \forall t \in \{0, .., T-1\} \ . \tag{7.27a}$$

- The *local information* structure captures only the information provided by the local uncertainties:

$$\mathcal{F}_t^i = \sigma(\boldsymbol{W}_1^i, \cdots, \boldsymbol{W}_t^i) \ , \ \ \forall t \in \{0, .., T-1\} \ , \ \ \forall i \in \{1, .., N\} \ . \tag{7.27b}$$

  Of course, we have that $\mathcal{F}_t^i \subset \mathcal{F}_t$, for all $i \in \{1, .., N\}$.

We recall the notion of $\mathcal{A}$-adapted stochastic processes hereafter.

Definition 7.2.2. *Let $\mathbb{Z} = \prod_{t=0}^{T-1} \mathbb{Z}_t$ be a product of measurable spaces, and $\mathcal{A} = \{\mathcal{A}_t\}_{t \in \{0,..,T\}}$ be a filtration on $(\Omega, \mathcal{A})$. We say that the stochastic process $\boldsymbol{Z} = (\boldsymbol{Z}_0, \cdots, \boldsymbol{Z}_{T-1})$ — where $\boldsymbol{Z}_t : \Omega \to \mathbb{Z}_t$ for all $t \in \{0, .., T-1\}$ — is $\mathcal{A}$-adapted if for all $t \in \{0, .., T-1\}$ $\boldsymbol{Z}_t$ is measurable w.r.t. $\mathcal{A}_t$. We note $\mathbb{L}^0(\Omega, \mathcal{A}, \mathbb{P}, \mathbb{Z})$ the space of $\mathcal{A}$-adapted process with image in the space $\mathbb{Z}$.*

7.2.3.3. Global stochastic control problem

With the local data detailed in §7.2.3.1 and the global data detailed in §7.2.3.2, we formulate a *global optimization problem*

$$V_0^\sharp(x_0) = \min_{\boldsymbol{X}, \boldsymbol{U}} \ \mathbb{E}\big[\sum_{i=1}^{N} \sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + K^i(\boldsymbol{X}_T^i) \big] \ , \tag{7.28a}$$

$$\text{w.r.t. } \boldsymbol{X}_{t+1}^i = g_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) \ , \ \ \boldsymbol{X}_0^i = x_0^i \ , \qquad \forall i \ , \ \forall t \ , \tag{7.28b}$$

$$\sigma(\boldsymbol{U}_t^i) \subset \mathcal{G}_t^i \ , \qquad \forall i \ , \ \forall t \ , \tag{7.28c}$$

$$\big(\Theta_t^1(\boldsymbol{X}_t^1, \boldsymbol{U}_t^1), \cdots, \Theta_t^N(\boldsymbol{X}_t^N, \boldsymbol{U}_t^N)\big) \in -S_t \ , \qquad \forall t \ , \tag{7.28d}$$

where

- $x_0 = (x_0^1, \cdots, x_0^N) \in \mathbb{X}_0$ is the initial position.
- The stochastic processes $\{\boldsymbol{X}_t^i\}_{t \in \{0,..,T\}}$ and $\{\boldsymbol{U}_t^i\}_{t \in \{0,..,T-1\}}$ are called *local state* and *local control* processes.
- The stochastic processes $\boldsymbol{X} = \{\boldsymbol{X}_t\}_{t \in \{0,..,T\}}$ and $\boldsymbol{U} = \{\boldsymbol{U}_t\}_{t \in \{0,..,T-1\}}$ — taking values respectively in $\mathbb{X}_t$ and in $\mathbb{U}_t$ — are called *global state* and *control* processes,
- Decisions $\boldsymbol{U}_t^i$ are measurable w.r.t. given σ-fields $\mathcal{G}_t^i$, that is,

$$\sigma(\boldsymbol{U}_t^i) \subset \mathcal{G}_t^i \ , \ \ \forall t \in \{0, .., T-1\} \ , \tag{7.29}$$

  with $\mathcal{G}_t^i = \mathcal{F}_t^i$ or $\mathcal{G}_t^i = \mathcal{F}_t$ (see Equations (7.27b) and (7.27a)),

- The global coupling constraints

$$\big(\Theta_t^1(\boldsymbol{X}_t^1,\boldsymbol{U}_t^1),\cdots,\Theta_t^N(\boldsymbol{X}_t^N,\boldsymbol{U}_t^N)\big)\in -S_t\ ,\ \ \forall t\in\{0,..,T-1\}\ , \tag{7.30}$$

  have to be taken in the $\mathbb{P}$-almost sure sense.

Problem (7.28) is parameterized by the family of admissible sets $\{S_t\}_{t\in\{0,..,T-1\}}$, the initial position $x_0\in\mathbb{X}_0$ and the measurability constraints (7.29) defined by the σ-field $\mathcal{G}_t^i$.

7.2.3.4. Local price and resource value functions

Following the decomposition schemes introduced in §7.2.2, we will now obtain lower and upper bounds of the global multistage stochastic problem (7.28).

Let $i\in\{1,..,N\}$ be a local unit, and $\boldsymbol{\lambda}^i=(\boldsymbol{\lambda}_0^i,\cdots,\boldsymbol{\lambda}_{T-1}^i)\in\mathbb{L}^0(\Omega,\mathcal{F},\mathbb{P},(\Xi^i)^\star)$ be a *local price process*. The *local price value function*, defined in Equation (7.8), writes here

$$\underline{V}_0^i[\boldsymbol{\lambda}^i](x_0^i)=\min_{\boldsymbol{X}^i,\boldsymbol{U}^i}\ \mathbb{E}\big[\sum_{t=0}^{T-1}L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)+\boldsymbol{\lambda}_t^i\cdot\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)+K^i(\boldsymbol{X}_T^i)\big]\ , \tag{7.31a}$$

$$\text{s.t. } \boldsymbol{X}_{t+1}^i=g_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)\ ,\ \ \boldsymbol{X}_0^i=x_0^i\ , \qquad \forall t\ , \tag{7.31b}$$

$$\sigma(\boldsymbol{U}_t^i)\subset\mathcal{G}_t^i\ , \qquad \forall t\ . \tag{7.31c}$$

We suppose that measurability and integrability assumptions hold, so that the expression in (7.31) (and by extension the expression $\mathbb{E}[\sum_{t=0}^{T-1}\boldsymbol{\lambda}_t^i\cdot\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)]$) makes sense [1].

Let $\boldsymbol{R}^i=(\boldsymbol{R}_0^i,\cdots,\boldsymbol{R}_{T-1}^i)\in\mathbb{L}^0(\Omega,\mathcal{F},\mathbb{P},\Xi^i)$ be a *local resource process*. The *local resource value function*, defined in Equation (7.9), writes here

$$\overline{V}_0^i[\boldsymbol{R}^i](x_0^i)=\min_{\boldsymbol{X}^i,\boldsymbol{U}^i}\ \mathbb{E}\big[\sum_{t=0}^{T-1}L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)+K^i(\boldsymbol{X}_T^i)\big]\ , \tag{7.32a}$$

$$\text{s.t. } \boldsymbol{X}_{t+1}^i=g_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)\ ,\ \ \boldsymbol{X}_0^i=x_0^i\ , \qquad \forall t, \tag{7.32b}$$

$$\sigma(\boldsymbol{U}_t^i)\subset\mathcal{G}_t^i\ , \qquad \forall t, \tag{7.32c}$$

$$\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)=\boldsymbol{R}_t^i\ , \qquad \forall t. \tag{7.32d}$$

In Equations (7.31) and (7.32), the (parametric) dependency w.r.t. the local price process $\boldsymbol{\lambda}^i$ and the local resource process $\boldsymbol{R}^i$ are denoted inside brackets to set it apart from the dependency w.r.t. the local position x_0^i.

We call the global processes $\boldsymbol{\lambda}\in\mathbb{L}^0(\Omega,\mathcal{F},\mathbb{P},\Xi^\star)$ and $\boldsymbol{R}\in\mathbb{L}^0(\Omega,\mathcal{F},\mathbb{P},\Xi)$ *coordination processes*.

7.2.3.5. Global upper and lower bounds

Applying Proposition 7.2.1 to the local price value function (7.31) and resource value function (7.32) makes it possible to bound the global problem (7.28). We first define the notion of *admissible* price and resource processes.

Definition 7.2.3. *We define the admissible set associated to the almost-sure constraints* (7.28d) *by*

$$\boldsymbol{\mathcal{S}}=\{\boldsymbol{Y}=(\boldsymbol{Y}_0,\cdots,\boldsymbol{Y}_{T-1})\in\mathbb{L}^0(\Omega,\mathcal{F},\mathbb{P},\Xi)\mid\boldsymbol{Y}_t\in S_t\quad\mathbb{P}-a.s.\}\ . \tag{7.33}$$

[1] We could consider for instance that for all $t\in\{0,..,T-1\}$, $\boldsymbol{\lambda}_t^i\in\mathbb{L}^2(\Omega,\mathcal{F}_t,\mathbb{P},(\Xi_t^i)^\star)$ and that $\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)\in\mathbb{L}^2(\Omega,\mathcal{F}_t,\mathbb{P},\Xi_t^i)$ are square integrable random variables. Another possibility is to consider that $\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)\in\mathbb{L}^\infty(\Omega,\mathcal{F}_t,\mathbb{P},\Xi_t^i)$ is a bounded random variable, and that $\boldsymbol{\lambda}_t^i\in\mathbb{L}^1(\Omega,\mathcal{F}_t,\mathbb{P},\Xi_t^i)$. We refer to Leclère (2014) for a discussion of the duality in the (L^1,L^∞) pairing.

Its dual cone writes

$$\boldsymbol{\mathcal{S}}^\star = \{\boldsymbol{Z} = (\boldsymbol{Z}_0, \cdots, \boldsymbol{Z}_{T-1}) \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \Xi^\star) \mid \left\langle \boldsymbol{Y} \, , \boldsymbol{Z} \right\rangle \geq 0 \, , \ \forall \boldsymbol{Y} \in \boldsymbol{\mathcal{S}}\} \, . \tag{7.34}$$

We say that $\boldsymbol{\lambda} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \Xi^\star)$ *is an* admissible coordination price process *if*

$$\boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}^\star \, . \tag{7.35}$$

In a similar manner, $\boldsymbol{R} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \Xi)$ *is an* admissible coordination resource process *if*

$$\boldsymbol{R} \in -\boldsymbol{\mathcal{S}} \, . \tag{7.36}$$

By considering admissible price and resource coordination processes, we are able to bound up and down Problem (7.28).

Proposition 7.2.4. *For any admissible price process* $\boldsymbol{\lambda} = (\boldsymbol{\lambda}^1, \cdots, \boldsymbol{\lambda}^N) \in \boldsymbol{\mathcal{S}}^\star$ *and for any admissible resource process* $\boldsymbol{R} = (\boldsymbol{R}^1, \cdots, \boldsymbol{R}^N) \in -\boldsymbol{\mathcal{S}}$*, we have that*

$$\sum_{i=1}^N \underline{V}_0^i[\boldsymbol{\lambda}^i](x_0^i) \leq V_0^\sharp(x_0^1, \cdots, x_0^N) \leq \sum_{i=1}^N \overline{V}_0^i[\boldsymbol{R}^i](x_0^i) \, , \ \forall (x_0^1, \cdots, x_0^N) \in \mathbb{X}_0 \, . \tag{7.37}$$

Proof. Application of Proposition 7.2.1 to Problem (7.28) with the value functions (7.31) and (7.32). □

We draw a parallel between the data of Problem (7.7) and of Problem (7.28) in Table 7.1.

Problem	Generic	Stochastic
Local decision space	$\mathcal{Z}^i$	$\mathbb{U}^i = \prod_{t=0}^{T-1} \mathbb{U}_t^i \, , \ \mathbb{X}^i = \prod_{t=0}^{T-1} \mathbb{X}_t^i$
Local decision variables	z^i	$\boldsymbol{X}^i, \boldsymbol{U}^i$
Local couplings	$\vartheta : \mathcal{Z}^i \to \Xi^i$	$\Theta_t^i : \mathbb{X}_t^i \times \mathbb{U}_t^i \to \Xi_t^i$
Admissible set	C	$\boldsymbol{\mathcal{S}}$
Dual cone	$C^\star$	$\boldsymbol{\mathcal{S}}^\star$
Resource	r	$\boldsymbol{R} = (\boldsymbol{R}_0, \cdots, \boldsymbol{R}_{T-1})$
Price	λ	$\boldsymbol{\lambda} = (\boldsymbol{\lambda}_0, \cdots, \boldsymbol{\lambda}_{T-1})$

Table 7.1.: Comparing the generic problem in §7.2.1 with the multistage stochastic problem in §7.2.3

7.3. Decomposition of local value functions by Dynamic Programming

We have seen in Section 7.2 that we are able to obtain upper and lower bounds of optimization problems by *spatial* decomposition. We now show that spatial decomposition schemes can be made compatible with time decomposition (Dynamic Programming) if price and resource processes are deterministic or Markovian. Furthermore, we even obtain bounds for the Bellman value functions of the original problem.

We focus in §7.3.1 on deterministic price and resource processes, and then study the Markovian case in §7.3.2. Eventually, we discuss in §7.3.4 the impact of the non-anticipativity constraint (7.29) on the resolution of the Dynamic Programming equations and extend the results to the hazard-decision information structure in §7.3.5.

In the sequel, we make the following key assumption for Dynamic Programming.

Assumption 7.3.1. *The global uncertainties $\boldsymbol{W}_1, \cdots, \boldsymbol{W}_T$ in (7.25) are a sequence of independent random variables.*

We note that, at a fixed time t, the local uncertainties $(\boldsymbol{W}_t^1, \cdots, \boldsymbol{W}_t^N)$ are not necessarily independent between units.

Global value functions. We define the *global value functions* $V_t : \mathbb{X}_t \to \mathbb{R}$ for all $t \in \{0, .., T-1\}$ as

$$V_t(x_t) = \min_{\boldsymbol{X}, \boldsymbol{U}} \mathbb{E}_{\boldsymbol{W}}\big[\sum_{i=1}^{N}\sum_{s=t}^{T-1} L_s^i(\boldsymbol{X}_s^i, \boldsymbol{U}_s^i, \boldsymbol{W}_{s+1}^i) + K^i(\boldsymbol{X}_T^i)\big] \,, \tag{7.38a}$$

$$\text{w.r.t.} \quad \boldsymbol{X}_{s+1}^i = g_s^i(\boldsymbol{X}_s^i, \boldsymbol{U}_s^i, \boldsymbol{W}_{s+1}^i) \,, \;\; \boldsymbol{X}_t^i = x_t^i \,, \qquad \forall i \,, \;\forall t, \tag{7.38b}$$

$$\sigma(\boldsymbol{U}_s^i) \subset \mathcal{G}_s^i \,, \qquad \forall i \,, \;\forall t, \tag{7.38c}$$

$$\big(\Theta_s^1(\boldsymbol{X}_s^1, \boldsymbol{U}_s^1), \cdots, \Theta_s^N(\boldsymbol{X}_s^N, \boldsymbol{U}_s^N)\big) \in -S_s \,, \qquad \forall t. \tag{7.38d}$$

In the global value function (7.38), the expected value is taken w.r.t. the global uncertainty process $(\boldsymbol{W}_{t+1}, \cdots, \boldsymbol{W}_T)$. We suppose that measurability and integrability assumptions hold, so that the expected value in (7.38) is well defined.

7.3.1. Decomposed value functions by means of deterministic coordination process

We prove in the sequel that if the admissible coordination price $\boldsymbol{\lambda}$ and resource $\boldsymbol{R}$ processes are deterministic, the local Problems (7.31) and the local Problems (7.32) satisfy local Dynamic Programming equations, provided that Assumption 7.3.1 holds. Furthermore, the Dynamic Programming equations are valid whether the information $\mathcal{G}_t^i$ is global (7.27a) or local (7.27b).

7.3.1.1. Price and resource value functions satisfy a Dynamic Programming equation

We first study the local price value function (7.31).

Proposition 7.3.2. *Let $\lambda^i = (\lambda_0^i, \cdots, \lambda_{T-1}^i) \in (\Xi^i)^\star$ be a deterministic time process. We have that*

$$\underline{V}_0^i[\lambda^i](x_0^i) = \underline{V}_0^i(x_0^i) \,, \tag{7.39}$$

where the function $\underline{V}_0^i$ (which depends on λ^i, despite the notation) is given by the recursive Dynamic Programming equations

$$\underline{V}_T^i(x_T^i) = K^i(x_T^i) \,, \tag{7.40a}$$

$$\underline{V}_t^i(x_t^i) = \min_{u_t^i \in \mathbb{U}_t^i} \mathbb{E}_{\boldsymbol{W}_{t+1}^i}\Big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \lambda_t^i \cdot \Theta_t^i(x_t^i, u_t^i) + \underline{V}_{t+1}^i\big(g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i)\big)\Big] \,. \tag{7.40b}$$

Proof. Let $\lambda^i = (\lambda_0^i, \cdots, \lambda_{T-1}^i) \in (\Xi^i)^\star$ be a deterministic price vector. Then, the price value function (7.31) rewrites

$$\underline{V}_0^i[\lambda^i](x_0^i) = \min_{\boldsymbol{X}^i, \boldsymbol{U}^i} \mathbb{E}\big[\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \lambda_t^i \cdot \Theta_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i) + K^i(\boldsymbol{X}_T^i)\big] \,, \tag{7.41a}$$

$$\text{s.t.} \;\; \boldsymbol{X}_{t+1}^i = g_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) \,, \;\; \boldsymbol{X}_0^i = x_0^i \,, \qquad \forall t \,, \tag{7.41b}$$

$$\sigma(\boldsymbol{U}_t^i) \subset \mathcal{G}_t^i \,, \qquad \forall t \,. \tag{7.41c}$$

Then, provided that Assumption 7.3.1 holds true, Problem (7.41) satisfies the local Dynamic Programming equations (7.40) (Bertsekas, 2005a). □

A similar result holds for the local resource value function (7.32).

Proposition 7.3.3. *Let $r^i = (r_0^i, \cdots, r_{T-1}^i) \in \Xi^i$ be a deterministic time process. We have that*

$$\overline{V}_0^i[r^i](x_0^i) = \overline{V}_0^i(x_0^i) \ , \tag{7.42}$$

where the function $\overline{V}_0^i$ (which depends on r^i, despite the notation) is given by the following Dynamic Programming equations

$$\overline{V}_T^i(x_T^i) = K^i(x_T^i) \ , \tag{7.43a}$$

$$\begin{aligned} \overline{V}_t^i(x_t^i) = \min_{u_t^i \in \mathbb{U}_t^i} \mathbb{E}_{\boldsymbol{W}_{t+1}^i} \Big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \overline{V}_{t+1}^i\big(g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i)\big) \Big] \ , \\ s.t.\ \Theta_t^i(x_t^i, u_t^i) = r_t^i \ . \end{aligned} \tag{7.43b}$$

Proof. Adaptation of the proof of Proposition 7.3.2. □

7.3.1.2. Relations between global and local value functions

As just proved, local price (7.31) and resource value functions (7.32) satisfy local Dynamic Programming equations (7.40) and (7.43), with associated local Bellman value functions $\{\underline{V}_t^i\}_{t\in\{0,..,T\}}$ and $\{\overline{V}_t^i\}_{t\in\{0,..,T\}}$ for all units $i \in \{1, .., N\}$. If the local price and resource processes are admissible, the local Bellman value functions bound the *global* value functions $\{V_t\}_{t\in\{0,..,T\}}$, thus extending the result obtained in Proposition 7.2.4 to all time $t \in \{0, .., T-1\}$.

Bounding the global value functions with local ones.

Proposition 7.3.4. *Let $\lambda = (\lambda_0, \cdots, \lambda_{T-1}) \in S^\star$ and $r = (r_0, \cdots, r_{T-1}) \in -S$ be two admissible deterministic coordination processes. Then the local price and resource value functions $\{\underline{V}_t^i\}_{t\in\{0,..,T\}}$ and $\{\overline{V}_t^i\}_{t\in\{0,..,T\}}$ defined by Equations (7.40) and (7.43) satisfy, for all $t \in \{0, .., T-1\}$*

$$\sum_{i=1}^N \underline{V}_t^i(x_t^i) \leq V_t(x_t^1, \cdots, x_t^N) \leq \sum_{i=1}^N \overline{V}_t^i(x_t^i) \ , \ \ \forall (x_t^1, \cdots, x_t^N) \in \mathbb{X}_t \ , \tag{7.44}$$

where $V_t(x_t)$ is the global value function defined by Equation (7.38).

Proof. Adaptation of the proof of Proposition 7.3.10. □

7.3.2. Decomposed value functions by means of Markovian coordination process

We extend the results obtained in §7.3.1 by moving from deterministic to Markovian price and resource processes. To the contrary of §7.3.1, we will be able to bound the global value functions (7.38) in a straightforward manner if the information structure is global, that is, if $\mathcal{G}_t^i$ satisfies Equation (7.27a): $\mathcal{G}_t^i = \sigma(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_t)$ for all $t \in \{0, .., T-1\}$. We will handle the local information case in §7.3.4.

7.3.2.1. Introducing Markovian design

In what follows, we suppose that the global price and resource processes in (7.31) and in (7.32) depend on extended states $\overline{\boldsymbol{Y}}$ and $\underline{\boldsymbol{Y}}$ whose dynamics are Markovian. For this purpose, we define the notion of price and resource Markovian designs.

Definition 7.3.5. *Let $\underline{\mathbb{Y}}_0, \cdots, \underline{\mathbb{Y}}_{T-1}$ be measurable spaces. Let $\boldsymbol{\lambda} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \Xi^\star)$ be a stochastic process. We say that $\boldsymbol{\lambda}$ is $\underline{\mathcal{D}}_{\underline{y}_0,h,\phi}$-Markovian if there exists $\underline{y}_0 \in \underline{\mathbb{Y}}_0$ and, for all $t \in \{0,..,T-1\}$, Markovian dynamics $h_t : \underline{\mathbb{Y}}_t \times \mathbb{W}_{t+1} \to \underline{\mathbb{Y}}_{t+1}$, and mappings $\phi_t : \underline{\mathbb{Y}}_t \to \Xi_t^\star$ such that*

$$\boldsymbol{\lambda}_t = \phi_t(\underline{\boldsymbol{Y}}_t) \ , \ \ \forall t \in \{0,..,T-1\} \ , \tag{7.45}$$

where $\underline{\boldsymbol{Y}} = (\underline{\boldsymbol{Y}}_0, \cdots, \underline{\boldsymbol{Y}}_{T-1}) \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \overline{\mathbb{Y}})$ is a Markovian process satisfying

$$\begin{cases} \underline{\boldsymbol{Y}}_0 = \underline{y}_0 \ , \\ \underline{\boldsymbol{Y}}_{t+1} = h_t(\underline{\boldsymbol{Y}}_t, \boldsymbol{W}_{t+1}) \ , \ \ \forall t \in \{0,..,T-1\} \ . \end{cases} \tag{7.46}$$

Definition 7.3.6. *Let $\overline{\mathbb{Y}}_0, \cdots, \overline{\mathbb{Y}}_{T-1}$ be measurable spaces. Let $\boldsymbol{R} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \Xi)$ be a stochastic process. We say that $\boldsymbol{R}$ is $\overline{\mathcal{D}}_{\overline{y}_0,h,\psi}$-Markovian if there exists $\overline{y}_0 \in \overline{\mathbb{Y}}_0$ and, for all $t \in \{0,..,T-1\}$, Markovian dynamics $h_t : \overline{\mathbb{Y}}_t \times \mathbb{W}_{t+1} \to \overline{\mathbb{Y}}_{t+1}$ and mappings $\psi_t : \overline{\mathbb{Y}}_t \to \Xi_t$ such that*

$$\boldsymbol{R}_t = \psi_t(\overline{\boldsymbol{Y}}_t) \ , \ \ \forall t \in \{0,..,T-1\} \ , \tag{7.47}$$

where $\overline{\boldsymbol{Y}} = (\overline{\boldsymbol{Y}}_0, \cdots, \overline{\boldsymbol{Y}}_{T-1}) \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \overline{\mathbb{Y}})$ is a Markovian process satisfying

$$\begin{cases} \overline{\boldsymbol{Y}}_0 = \overline{y}_0 \ , \\ \overline{\boldsymbol{Y}}_{t+1} = h_t(\overline{\boldsymbol{Y}}_t, \boldsymbol{W}_{t+1}) \ , \ \ \forall t \in \{0,..,T-1\} \ . \end{cases} \tag{7.48}$$

7.3.2.2. Price and resource local value functions satisfy a Dynamic Programming equation

Let $i \in \{1,..,N\}$ be a local unit. We prove that if the local price process $\boldsymbol{\lambda}^i$ is $\underline{\mathcal{D}}_{y_0,h,\phi^i}$-Markovian, the local value function (7.31) satisfies local Dynamic Programming equations.

Proposition 7.3.7. *For $t \in \{0,..,T-1\}$, let $h_t : \underline{\mathbb{Y}}_t \times \mathbb{W}_{t+1} \to \underline{\mathbb{Y}}_{t+1}$ and $\phi_t^i : \underline{\mathbb{Y}}_t \to (\Xi_t^i)^\star$ be two functions, with $\underline{\mathbb{Y}}_t$ given measurable space. If the local price process $\boldsymbol{\lambda}^i = (\boldsymbol{\lambda}_0^i, \cdots, \boldsymbol{\lambda}_{T-1}^i)$ is $\underline{\mathcal{D}}_{y_0,h,\phi}$-Markovian, we have that*

$$\underline{V}^i[\boldsymbol{\lambda}^i](x_0^i) = \underline{V}_0^i(x_0^i, \underline{y}_0) \ , \ \ \forall \underline{y}_0 \in \underline{\mathbb{Y}}_0 \ , \tag{7.49}$$

where the function $\underline{V}_0^i$ (which depends on $\boldsymbol{\lambda}^i$, despite the notation) is given by the following Dynamic Programming equations

$$\begin{cases} \underline{V}_T^i(x_T^i, \underline{y}_T) = K^i(x_T^i) \ , \\ \underline{V}_t^i(x_t^i, \underline{y}_t) = \min\limits_{u_t^i \in \mathbb{U}_t^i} \mathbb{E}_{\boldsymbol{W}_{t+1}} \Big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \phi_t^i(\underline{y}_t) \cdot \Theta_t^i(x_t^i, u_t^i) + \\ \qquad\qquad \underline{V}_{t+1}^i\big(g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i), h_t(\underline{y}_t, \boldsymbol{W}_{t+1})\big) \Big] \ . \end{cases} \tag{7.50}$$

Proof. If the process $\boldsymbol{\lambda}^i$ is such that $\boldsymbol{\lambda}^i_t = \phi^i_t(\underline{\boldsymbol{Y}}_t)$, the local price function (7.31) rewrites:

$$\begin{aligned}
\underline{V}^i[\boldsymbol{\lambda}^i](x^i_0) = \min_{\boldsymbol{X}^i,\boldsymbol{U}^i} \ & \mathbb{E}_{\boldsymbol{W}}\big[\sum_{t=0}^{T-1} L^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t,\boldsymbol{W}^i_{t+1}) + \big\langle \phi^i_t(\underline{\boldsymbol{Y}}_t)\,,\Theta^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t)\big\rangle + K^i(\boldsymbol{X}^i_T)\big]\,, \\
& \text{s.t. } \boldsymbol{X}^i_{t+1} = g^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t,\boldsymbol{W}^i_{t+1})\,,\ \ \boldsymbol{X}^i_0 = x^i_0\,, && \forall i\ ,\ \forall t, \\
& \quad \underline{\boldsymbol{Y}}_{t+1} = h_t(\underline{\boldsymbol{Y}}_t,\boldsymbol{W}_{t+1})\,,\ \ \underline{\boldsymbol{Y}}_0 = \underline{y}_0\,, && \forall i\ ,\ \forall t, \\
& \quad \sigma(\boldsymbol{U}^i_t) \subset \sigma(\boldsymbol{W}_1,\cdots,\boldsymbol{W}_t)\,, && \forall i\ ,\ \forall t.
\end{aligned} \tag{7.51}$$

Then, as the random variables $\boldsymbol{W}_1,\cdots,\boldsymbol{W}_T$ are supposed time independent, Problem (7.51) satisfies the local Dynamic Programming equations (7.50) with the extended Markovian state $(\boldsymbol{X}^i_t,\underline{\boldsymbol{Y}}_t)$. □

Whereas the Dynamic Programming equations (7.40) depend only on the local noise $\boldsymbol{W}^i$, the Dynamic Programming equations (7.50) depend on the global noise $\boldsymbol{W}$ because the dynamics of the process $\underline{\boldsymbol{Y}}$ in (7.46) depend on the global noise $\boldsymbol{W}$.

The extension of Proposition 7.3.7 to the case of the local resource value function (7.32) by Dynamic Programming is straightforward.

Proposition 7.3.8. *For $t \in \{0,..,T-1\}$, let $h_t : \overline{\mathbb{Y}}_t \times \mathbb{W}_{t+1} \to \overline{\mathbb{Y}}_{t+1}$ and $\psi^i_t : \overline{\mathbb{Y}}_t \to \Xi^i_t$ be two functions, with $\overline{\mathbb{Y}}_t$ given measurable space. If the local resource $\boldsymbol{R}^i = (\boldsymbol{R}^i_0,\cdots,\boldsymbol{R}^i_{T-1})$ is $\overline{\mathcal{D}}_{y_0,h,\psi}$-Markovian, we have*

$$\overline{V}^i_0[\boldsymbol{R}^i](x^i_0) = \overline{V}_0(x^i_0,\overline{y}_0)\ ,\ \ \forall \overline{y}_0 \in \overline{\mathbb{Y}}_0\ , \tag{7.52}$$

where the function $\overline{V}^i_0$ is given by the following Dynamic Programming equations

$$\overline{V}^i_T(x^i_T,\overline{y}_T) = K^i(x^i_T)\ , \tag{7.53a}$$

$$\begin{aligned}
\overline{V}^i_t(x^i_t,\overline{y}_t) = \min_{u^i_t \in \mathbb{U}^i_t} \ & \mathbb{E}_{\boldsymbol{W}_{t+1}}\Big[L^i_t(x^i_t,u^i_t,\boldsymbol{W}^i_{t+1}) + \overline{V}^i_{t+1}\big(g^i_t(x^i_t,u^i_t,\boldsymbol{W}^i_{t+1}),h_t(\overline{y}_t,\boldsymbol{W}_{t+1})\big)\Big] \\
& \textit{s.t. } \Theta^i_t(x^i_t,u^i_t) = \psi^i_t(\overline{y}_t)\ .
\end{aligned} \tag{7.53b}$$

Proof. If the local resource process $\boldsymbol{R}^i$ writes $\boldsymbol{R}^i_t = \psi^i_t(\overline{\boldsymbol{Y}}_t)$ for all $t \in \{0,..,T-1\}$, the local resource function (7.32) rewrites:

$$\overline{V}^i_0[\boldsymbol{R}^i](x^i_0) = \min_{\boldsymbol{X}^i,\boldsymbol{U}^i} \ \mathbb{E}_{\boldsymbol{W}}\big[\sum_{t=0}^{T-1} L^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t,\boldsymbol{W}_{t+1}) + K^i(\boldsymbol{X}^i_T)\big]\ , \tag{7.54a}$$

$$\text{s.t. } \boldsymbol{X}^i_{t+1} = g^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t,\boldsymbol{W}^i_{t+1})\,,\ \ \boldsymbol{X}^i_0 = x^i_0\,, \qquad \forall i\ ,\ \forall t, \tag{7.54b}$$

$$\overline{\boldsymbol{Y}}_{t+1} = h_t(\overline{\boldsymbol{Y}}_t,\boldsymbol{W}_{t+1})\,,\ \ \overline{\boldsymbol{Y}}_0 = \overline{y}_0\,, \qquad \forall t, \tag{7.54c}$$

$$\Theta^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t) = \psi^i_t(\overline{\boldsymbol{Y}}_t)\,, \qquad \forall i\ ,\ \forall t, \tag{7.54d}$$

$$\sigma(\boldsymbol{U}^i_t) \subset \sigma(\boldsymbol{W}_1,\cdots,\boldsymbol{W}_t)\,, \qquad \forall i\ ,\ \forall t. \tag{7.54e}$$

Then, as uncertainties $\boldsymbol{W}_1,\cdots,\boldsymbol{W}_T$ are supposed time independent, Problem (7.54) satisfies the local Dynamic Programming equations (7.53) by considering the extended Markovian state $(\boldsymbol{X}_t,\overline{\boldsymbol{Y}}_t)$. □

We note that, in the expressions of the Dynamic Programming equations (7.50) and (7.53) appears the global noise $\boldsymbol{W}$ (and not the local noise $\boldsymbol{W}^i$).

7.3.2.3. Relations between global and local value functions

We prove in the sequel that if the resource and the price designs are chosen appropriately, we are able to bound the global value functions (7.38) up and down for all time $t \in \{0, .., T-1\}$.

First, we must ensure that the price and resource Markovian designs are admissible.

Definition 7.3.9.

- *We say that the price design $\underline{\mathcal{D}}_{\underline{y}_0,h,\phi}$ (defined in Definition 7.3.5) is* dual admissible *if*

$$\mathrm{im}(\phi_t) \subset S_t^\star \ , \ \ \forall t \in \{0, .., T-1\} \ . \tag{7.55}$$

- *We say that the resource design $\overline{\mathcal{D}}_{\overline{y}_0,h,\psi}$ (defined in Definition 7.3.6) is* primal admissible *if*

$$\mathrm{im}(\psi_t) \subset -S_t \ , \ \ \forall t \in \{0, .., T-1\} \ . \tag{7.56}$$

Proposition 7.3.10. *Let $\underline{\mathcal{D}}_{\underline{y}_0,h,\phi}$ be a* dual admissible *design, and $\overline{\mathcal{D}}_{\overline{y}_0,h,\psi}$ be a* primal admissible *design. We assume that the information $\mathcal{G}_t^i$ is* global*:*

$$\mathcal{G}_t^i = \mathcal{F}_t = \sigma(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_t) \ , \ \ \forall t \in \{0, .., T-1\} \ , \ \ \forall i \in \{1, .., N\} \ . \tag{7.57}$$

Then the local price and resource value functions $\{\underline{V}_t^i\}_{t\in\{0,..,T\}}$ and $\{\overline{V}_t^i\}_{t\in\{0,..,T\}}$ defined by Equations (7.50) *and* (7.53) *satisfy, for all $t \in \{0, .., T-1\}$, $(x_t^1, \cdots, x_t^N) \in \mathbb{X}_t$ and $(\underline{y}_t, \overline{y}_t) \in \underline{\mathbb{Y}}_t \times \overline{\mathbb{Y}}_t$,*

$$\sum_{i=1}^N \underline{V}_t^i(x_t^i, \underline{y}_t) \leq V_t(x_t^1, \cdots, x_t^N) \leq \sum_{i=1}^N \overline{V}_t^i(x_t^i, \overline{y}_t) \ , \tag{7.58}$$

where $V_t(x_t)$ is the global value function defined by Equation (7.38)*.*

Proof. We prove the proposition by induction.

The global value functions (7.38) satisfy the global Dynamic Programming equations (Bertsekas, 2005a).

$$\begin{aligned} V_T(x_T) &= K(x_T) \ , \\ V_t(x_t) &= \min_{u_t \in \mathbb{U}_t} \mathbb{E}_{\boldsymbol{W}}\big[\sum_{i=1}^N L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + V_{t+1}\big(\boldsymbol{X}_{t+1}^1, \cdots, \boldsymbol{X}_{t+1}^N\big)\big] \\ &\qquad \text{s.t. } \boldsymbol{X}_{t+1}^i = g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) \ , \qquad \forall i \in \{1, .., N\} \ , \\ &\qquad\quad \big(\Theta_t^1(x_t^1, u_t^1), \cdots, \Theta_t^N(x_t^N, u_t^N)\big) \in -S_t \ . \end{aligned} \tag{7.59}$$

For $t \in \{0, .., T-1\}$, we define $\underline{V}_t[\boldsymbol{\lambda}]$ the dual global value function associated to Problem (7.38):

$$\begin{aligned} \underline{V}_t[\boldsymbol{\lambda}](x_t, \underline{y}_t) := \min_{\boldsymbol{X},\boldsymbol{U}} \ & \mathbb{E}_{\boldsymbol{W}}\big[\sum_{i=1}^N \sum_{s=t}^{T-1} L_s^i(\boldsymbol{X}_s^i, \boldsymbol{U}_s^i, \boldsymbol{W}_{s+1}^i) + \phi_s^i(\underline{\boldsymbol{Y}}_s) \cdot \Theta_s^i(\boldsymbol{X}_s^i, \boldsymbol{U}_s^i) + K^i(\boldsymbol{X}_T^i)\big] \ , \\ & \text{s.t. } \boldsymbol{X}_{s+1}^i = g_s^i(\boldsymbol{X}_s^i, \boldsymbol{U}_s^i, \boldsymbol{W}_{s+1}^i) \ , \ \ \boldsymbol{X}_t^i = x_t^i \ , \qquad \forall i \ , \ \forall s, \\ & \qquad \sigma(\boldsymbol{U}_s^i) \subset \sigma(\boldsymbol{W}_{t+1}, \cdots, \boldsymbol{W}_s) \ , \qquad\qquad \forall i \ , \ \forall s, \end{aligned} \tag{7.60}$$

Using Proposition 7.2.1, we know that for all admissible price process satisfying $\boldsymbol{\lambda}_s \in \boldsymbol{\mathcal{S}}_s^\star$ for all $s \in \{t, .., T-1\}$, we have

$$\underline{V}_t[\boldsymbol{\lambda}](\cdot, \underline{y}_t) \leq V_t(\cdot) \ , \qquad \forall \underline{y}_t \in \underline{\mathbb{Y}}_t \ . \tag{7.61}$$

We now prove by induction that for all $t \in \{0, .., T-1\}$, we have:

$$\underline{V}_t[\boldsymbol{\lambda}](x_t) = \sum_{i=1}^{N} \underline{V}_t^i(x_t^i, \underline{y}_t) \ , \ \ \forall x_t = (x_t^1, \cdots, x_t^N) \in \mathbb{X}_t \ , \ \ \underline{y}_t \in \underline{\mathbb{Y}}_t \ . \tag{7.62}$$

- The property (7.62) holds at time $t = T$.
- Let $t < T$ such that the property (7.62) holds at time $t+1$, by induction hypothesis. For all $x_t \in \mathbb{X}_t$ and $\underline{y}_t \in \underline{\mathbb{Y}}_t$, we have

$$\begin{aligned}
\underline{V}_t[\boldsymbol{\lambda}](x_t, \underline{y}_t) &= \min_{u_t \in \mathbb{U}_t} \mathbb{E}_{\boldsymbol{W}} \Big[\sum_{i=1}^{N} \big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \phi_t^i(\underline{y}_t) \cdot \Theta_t^i(x_t^i, u_t^i) + \\
&\qquad\qquad V_{t+1}[\boldsymbol{\lambda}]\big(g_t(x_t, u_t, \boldsymbol{W}_{t+1}), \underline{\boldsymbol{Y}}_{t+1}\big)\big] \Big] \\
&\text{by Dynamic Programming equation (7.59)} \\
&= \min_{u_t \in \mathbb{U}_t} \mathbb{E}_{\boldsymbol{W}} \Big[\sum_{i=1}^{N} \big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \phi_t^i(\underline{y}_t) \cdot \Theta_t^i(x_t^i, u_t^i) + \\
&\qquad\qquad \sum_{i=1}^{N} \underline{V}_{t+1}^i\big(g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i), \underline{\boldsymbol{Y}}_{t+1}\big)\big] \Big] \\
&\text{by induction hypothesis (7.62)} \\
&= \sum_{i=1}^{N} \min_{u_t^i \in \mathbb{U}_t^i} \mathbb{E}_{\boldsymbol{W}} \big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \phi_t^i(\underline{y}_t) \cdot \Theta_t^i(x_t^i, u_t^i) + \\
&\qquad\qquad \underline{V}_{t+1}^i\big(g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i), \underline{\boldsymbol{Y}}_{t+1}\big)\big] \\
&= \sum_{i=1}^{N} \underline{V}_t^i(x_t^i, \underline{y}_t) \qquad \text{using Equation (7.50).}
\end{aligned}$$

  Thus we have proven the induction hypothesis at time t.

This ends the proof.

We adapt the proof for the upper bound. □

We note that by choosing dynamics $h_t : \underline{\mathbb{Y}}_t \times \mathbb{W}_{t+1} \to \underline{\mathbb{Y}}_{t+1}$ satisfying

$$h_t(\underline{y}_t, w_{t+1}) = (\underline{y}_t, w_{t+1}) \ , \ \ \forall t \in \{0, .., T-1\} \ , \tag{7.63}$$

we have $\underline{y}_t = (\underline{y}_0, w_1, \cdots, w_t)$ for all $t \in \{0, .., t\}$. Thus, the Markovian process $\underline{\boldsymbol{Y}}$ embeds all previous uncertainties. However, the size of $\underline{\boldsymbol{Y}}$ becomes quickly intractable to solve local value functions $\underline{V}_0^i[\boldsymbol{\lambda}^i]$ by Dynamic Programming. Therefore we usually use dynamics h_t that embed less information.

7.3.3. Producing admissible policies

We suppose that we have computed every local value functions $\{\underline{V}_t^i\}_{t \in \{0,..,T\}}$ and $\{\overline{V}_t^i\}_{t \in \{0,..,T\}}$ by using Equations (7.40) or (7.50) for the price value functions and Equations (7.43) or (7.53) for the resource value functions.

We are able to recover global admissible policies by using the sum of the value functions $\{\underline{V}_t^i\}_{t \in \{0,..,T\}}$ and $\{\overline{V}_t^i\}_{t \in \{0,..,T\}}$.

We obtain a *global price policy* $\underline{\pi}_t : \mathbb{X}_t \to \mathbb{U}_t$

$$\begin{aligned}\underline{\pi}_t(x_t^1, \cdots, x_t^N) \in \underset{u_t^1, \cdots, u_t^N}{\arg\min}\ & \mathbb{E}\Big[\sum_{i=1}^N L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}) + \underline{V}_{t+1}^i\big(\boldsymbol{X}_{t+1}^i\big)\Big] \ , \\ & \text{s.t. } \boldsymbol{X}_{t+1}^i = g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}) \ , \ \ \forall i \in \{1, .., N\} \ , \\ & \quad \big(\Theta_t^1(x_t^1, u_t^1), \cdots, \Theta_t^N(x_t^N, u_t^N)\big) \in -S_t \ ,\end{aligned} \tag{7.64}$$

and a *global resource policy* $\overline{\pi}_t : \mathbb{X}_t \to \mathbb{U}_t$

$$\begin{aligned}\overline{\pi}_t(x_t^1, \cdots, x_t^N) \in \underset{u_t^1, \cdots, u_t^N}{\arg\min}\ & \mathbb{E}\Big[\sum_{i=1}^N L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}) + \overline{V}_{t+1}^i\big(\boldsymbol{X}_{t+1}^i\big)\Big] \ , \\ & \text{s.t. } \boldsymbol{X}_{t+1}^i = g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}) \ , \ \ \forall i \in \{1, .., N\} \ , \\ & \quad \big(\Theta_t^1(x_t^1, u_t^1), \cdots, \Theta_t^N(x_t^N, u_t^N)\big) \in -S_t \ .\end{aligned} \tag{7.65}$$

For all time $t \in \{0, .., T-1\}$, the expected cost of a given policy $\pi_t, \cdots, \pi_{T-1}$ starting from position x at time t writes

$$V_t^\pi(x) = \mathbb{E}\Big[\sum_{i=1}^N \sum_{s=t}^{T-1} L_s^i(\boldsymbol{X}_s^i, \pi_s^i(x_s), \boldsymbol{W}_{t+1}^i) + K^i(\boldsymbol{X}_T^i) \mid \boldsymbol{X}_t = x\Big] \ . \tag{7.66}$$

We prove hereafter that we are able to bound the performance of the global resource policy (7.65).

Proposition 7.3.11. *Let $t \in \{0, .., T\}$ and $x_t = (x_t^1, \cdots, x_t^N) \in \mathbb{X}_t$ be a given state. Then, we have the following upper bound on the expected value of the global resource policy* (7.65)

$$V_t^{\overline{\pi}}(x_t) \leq \sum_{i=1}^N \overline{V}_t^i(x_t^i) \ . \tag{7.67}$$

Proof. We prove the result by backward induction. At time $t = T$, the result is straightforward as for all $i \in \{1, .., N\}$, $\overline{V}_t^i = K^i$.

Let $t \in \{0, .., T-1\}$ such that the result holds at time $t+1$: we have $V_{t+1}^{\overline{\pi}} \leq \sum_{i=1}^N \overline{V}_{t+1}^i$. Then, we deduce that, for all $x_t \in \mathbb{X}_t$,

$$V_t^{\overline{\pi}}(x_t) = \mathbb{E}\big[\sum_{i=1}^N \big(L_t^i(x_t^i, \overline{\pi}_t^i(x_t), \boldsymbol{W}_{t+1}^i)\big) + V_{t+1}^{\overline{\pi}}(\boldsymbol{X}_{t+1})\big] \ , \tag{7.68a}$$

by definition of $V_t^{\overline{\pi}}$ in (7.66). As, by induction hypothesis, $V_{t+1}^{\overline{\pi}} \leq \sum_{i=1}^N \overline{V}_{t+1}^i$, we have

$$V_t^{\overline{\pi}}(x_t) \leq \mathbb{E}\big[\sum_{i=1}^N L_t^i(x_t^i, \overline{\pi}_t^i(x_t), \boldsymbol{W}_{t+1}^i) + \overline{V}_{t+1}^i(\boldsymbol{X}_{t+1}^i)\big] \ . \tag{7.68b}$$

By using the definition of the global resource policy in Equation (7.65), we obtain

$$\begin{aligned}V_t^{\overline{\pi}}(x_t) \leq \min_{u_t^1, \cdots, u_t^N}\ & \mathbb{E}\big[\sum_{i=1}^N L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \overline{V}_{t+1}^i(\boldsymbol{X}_{t+1}^i)\big] \\ & \text{s.t.} \quad \big(\Theta_t^1(x_t^1, u_t^1), \cdots, \Theta_t^N(x_t^N, u_t^N)\big) \in -S_t\end{aligned} \tag{7.68c}$$

By restraining the problem with an admissible allocation $(r_t^1, \cdots, r_t^N) \in -S_t$, we have

$$V_t^{\overline{\pi}}(x_t) \leq \min_{u_t^1, \cdots, u_t^N} \mathbb{E}\big[\sum_{i=1}^N L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \overline{V}_{t+1}^i(\boldsymbol{X}_{t+1}^i)\big] \\ \text{s.t.} \quad \Theta_t^i(x_t^i, u_t^i) = r_t^i \ , \tag{7.68d}$$

that is,

$$V_t^{\overline{\pi}}(x_t) \leq \sum_{i=1}^N \min_{u_t^i, \Theta_t^i(u_t^i) = r_t^i} \mathbb{E}\big(L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \overline{V}_{t+1}^i(\boldsymbol{X}_{t+1}^i)\big) \ , \tag{7.68e}$$

as we do not have any coupling left in (7.68d). Thus $V_t^{\overline{\pi}}(x_t) \leq \sum_{i=1}^N \overline{V}_t^i(x_t^i)$ by using Equation (7.43) and by choosing the admissible allocation $(r_t^1, \cdots, r_t^N)$ used to define the local value functions $\overline{V}_t^i$ in (7.43). Hence the result at time t. □

Furthermore, we know that for all policy π, we have $V_t \leq V_t^\pi$ as the global Bellman function gives the optimal cost starting at any point $x_t \in \mathbb{X}_t$. Combining this lower bound with the upper bound given by Equation (7.67), we are able to bound the expected value of the global resource policy at any time t, as

$$V_t(x_t) \leq V_t^{\overline{\pi}}(x_t) \leq \sum_{i=1}^N \overline{V}_t^i(x_t^i) \ , \quad \forall x_t = (x_t^1, \cdots, x_t^N) \in \mathbb{X}_t \ . \tag{7.69a}$$

Concerning the global price policy (7.64) obtained with the price value functions $\{\underline{V}_t^i\}_{t \in \{0,..,T\}}$, we get only a lower bound, as

$$\sum_{i=1}^N \underline{V}_t^i(x_t^i) \leq V_t(x_t) \leq V_t^{\underline{\pi}}(x_t) \ . \tag{7.69b}$$

7.3.4. Discussing the impact of the information constraints

In Problem (7.28), the local decision processes $\{\boldsymbol{U}_t^i\}_{t \in \{0,..,T-1\}}$ satisfy the non-anticipativity constraints (7.29) constraining their measurability with the σ-fields $\mathcal{G}_t^i$. We discuss here the impact of the choice of σ-field $\mathcal{G}_t^i$ on price and resource decompositions. We sum up the different possible choices in Table 7.2 and state when we are able to bound the global Bellman functions (7.38) for all time $t \in \{0, .., T-1\}$.

	σ-field	Local	Global
Information	$\mathcal{G}_t^i$	$\sigma(\boldsymbol{W}_1^i, \cdots, \boldsymbol{W}_t^i)$	$\sigma(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_t)$
Deterministic	DP	$\min_{u^i} \mathbb{E}_{\boldsymbol{W}_{t+1}^i}$	$\min_{u^i} \mathbb{E}_{\boldsymbol{W}_{t+1}^i}$
	Bounds on V_t	Yes	Yes
	Bounds on V_0	Yes	Yes
Markovian	DP	$\min_{u^i} \mathbb{E}_{\boldsymbol{W}_{t+1}^i}$	$\min_{u^i} \mathbb{E}_{\boldsymbol{W}_{t+1}}$
	Bounds on V_t	No	Yes
	Bounds on V_0	Yes	Yes

Table 7.2.: Summing up the differences between local and global information structure w.r.t. the bounds obtained in decomposition.

Local information structure. If the information structure is local, the local decisions $\boldsymbol{U}_t^i$ are measurable with the previous local uncertainties: $\mathcal{G}_t^i = \sigma(\boldsymbol{W}_1^i, \cdots, \boldsymbol{W}_t^i)$. In that case,

- The expected value in local Dynamic Programming equations (7.40)-(7.43)-(7.50)-(7.53) is taken w.r.t. the local noise $\boldsymbol{W}_t^i$.
- We are able to write locally the recursive Dynamic Programming equations (7.40)-(7.43) with *deterministic* price λ and resource r processes. The global Bellman functions (7.38) are bounded up and down by the sum of the local Bellman functions, as stated in Proposition 7.3.4.
- As the Markovian designs defined in Definition 7.3.5 and Definition 7.3.6 depend on the global uncertainty $\boldsymbol{W}$, in Equation (7.50) (resp. (7.53)) the local decisions depend on the global extended states $\underline{\boldsymbol{Y}}$ (resp. $\overline{\boldsymbol{Y}}$).
- If instead of a global extended state $\underline{\boldsymbol{Y}}$ we consider a local extended state $\underline{\boldsymbol{Y}}^i$ with associated initial position $\underline{y}_0^i$ and dynamics $h_t^i : \underline{\mathbb{Y}}_t^i \times \mathbb{W}_{t+1}^i \to \underline{\mathbb{Y}}_{t+1}^i$, the local decisions would be measurable w.r.t. the local previous uncertainties, but we are generally unable to ensure that

$$(\phi_t^1(\underline{y}_t^1), \cdots, \phi_t^N(\underline{y}_t^N)) \in S_t^\star \ , \tag{7.70}$$

  holds globally for all $\underline{y}_t^1, \cdots, y_t^N$ (apart if y_t^i are constant). However, if S_t has simple structure, we would be able to ensure that (7.70) holds true for all time t (consider for instance $S_t = \{0\}$ with associated dual cone $S_t^\star = \Xi_t$).

Global information structure. By choosing the global information structure, the decisions are measurable w.r.t. the past local global uncertainties: $\mathcal{G}_t^i = \sigma(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_t)$. In that case,

- The expected value is taken w.r.t. the local noise $\boldsymbol{W}_t^i$ in the deterministic case (7.40)-(7.43), and w.r.t. the global noise $\boldsymbol{W}_t^i$ in the Markovian case (7.50)-(7.53).
- If the price and resource processes are deterministic, the local value functions (7.31) and (7.32) satisfy local Dynamic Programming equations depending on the local uncertainties $\boldsymbol{W}_t^i$. We are able to bound the global Bellman value functions (7.38) for all time t.
- If the price and resource processes are Markovian, the local value functions (7.31) and (7.32) satisfy local Dynamic Programming equations depending this time on the global uncertainties $\boldsymbol{W}_t$ as the dynamics of the extended states $\overline{\boldsymbol{Y}}$ and $\underline{\boldsymbol{Y}}$ in (7.46) and in (7.48) depend on the global uncertainties. We are also able to bound the global Bellman value functions (7.38) for all time t.

7.3.5. Hazard decision information structure

Until now, we have handled only the decision-hazard[2] as information structure. However, we are able to extend the results to other information structures, namely hazard-decision[3].

In hazard-decision, decisions are taken knowing the realization of the global uncertainties between time t and $t+1$. In this case, the σ-field in (7.29) writes either

$$\mathcal{G}_t^i = \sigma(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_t, \boldsymbol{W}_{t+1}) \ , \ \ \forall i \in \{1, .., N\} \ , \ \ \forall t \in \{0, .., T-1\} \ . \tag{7.71a}$$

for the global information case, or

$$\mathcal{G}_t^i = \sigma(\boldsymbol{W}_1^i, \cdots, \boldsymbol{W}_t^i, \boldsymbol{W}_{t+1}^i) \ , \ \ \forall i \in \{1, .., N\} \ , \ \ \forall t \in \{0, .., T-1\} \ . \tag{7.71b}$$

[2]Or here-and-now in the stochastic programming terminology
[3]Or wait-and-see in the stochastic programming terminology

for the local information case. As decisions are measurable w.r.t. $\boldsymbol{W}_{t+1}$, we are able to handle more generic coupling constraints than in Problem (7.28), namely ones depending on the local uncertainties:

$$\Theta_t^i : \mathbb{X}_t^i \times \mathbb{U}_t^i \times \mathbb{W}_{t+1}^i \to \Xi_t^i \ , \ \ \forall i \in \{1,..,N\} \ , \ \ \forall t \in \{0,..,T-1\} \ . \tag{7.72}$$

We obtain a new global problem:

$$V_0^\sharp(x_0) = \min_{\boldsymbol{X},\boldsymbol{U}} \ \mathbb{E}\big[\sum_{i=1}^N \sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + K^i(\boldsymbol{X}_T^i)\big] \ , \tag{7.73a}$$

$$\text{w.r.t.} \quad \boldsymbol{X}_{t+1}^i = g_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) \ , \ \ \boldsymbol{X}_0^i = x_0^i \ , \qquad \forall i \ , \ \forall t, \tag{7.73b}$$

$$\sigma(\boldsymbol{U}_t^i) \subset \mathcal{G}_t^i \ , \qquad \forall i \ , \ \forall t, \tag{7.73c}$$

$$\big(\Theta_t^1(\boldsymbol{X}_t^1, \boldsymbol{U}_t^1, \boldsymbol{W}_{t+1}^1), \cdots, \Theta_t^N(\boldsymbol{X}_t^N, \boldsymbol{U}_t^N, \boldsymbol{W}_{t+1}^N)\big) \in -S_t \ , \qquad \forall t. \tag{7.73d}$$

We adapt the results of Section 7.3.1 and Section 7.3.2 to the hazard-decision case.

Dynamic Programming equations with price processes. The Dynamic Programming equations in Proposition 7.3.2 and in Proposition 7.3.7 rewrite in the following manners, for all units $i \in \{1,..,N\}$.

- If the local price process $\lambda^i = (\lambda_0^i, \cdots, \lambda_{T-1}^i)$ is deterministic, we look at the following Dynamic Programming equations

$$\underline{V}_t^i(x_t^i) = \mathbb{E}_{\boldsymbol{W}_{t+1}^i}\big[\min_{\boldsymbol{U}_t^i \in \mathbb{U}_t^i} L_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \lambda_t^i \cdot \Theta_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \underline{V}_{t+1}^i\big(g_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i)\big)\big] \ . \tag{7.74}$$

- If the local price process $\boldsymbol{\lambda}^i = (\boldsymbol{\lambda}_0^i, \cdots, \boldsymbol{\lambda}_{T-1}^i)$ is generated by a price Markovian design $\underline{\mathcal{D}}_{y_0,h,\psi}$ (see Definition 7.3.5), we deal with the following Dynamic Programming equations

$$\underline{V}_t^i(x_t^i, y_t) = \mathbb{E}_{\boldsymbol{W}_{t+1}}\Big[\min_{\boldsymbol{U}_t^i \in \mathbb{U}_t^i} L_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \phi_t^i(y_t, \boldsymbol{W}_{t+1}) \cdot \Theta_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \underline{V}_{t+1}^i\big(g_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i), h_t(y_t, \boldsymbol{W}_{t+1})\big)\Big] \ . \tag{7.75}$$

Dynamic Programming equations with resource processes. We adapt similarly the Dynamic Programming equations detailed in Proposition 7.3.3 and in Proposition 7.3.8.

- If the local resource process $r^i = (r_0^i, \cdots, r_{T-1}^i)$ is deterministic, we look at the following Dynamic Programming equations

$$\begin{aligned} \overline{V}_t^i(x_t^i) = \min_{\boldsymbol{U}_t^i \in \mathbb{U}_t^i} \ & \mathbb{E}_{\boldsymbol{W}_{t+1}^i}\Big[L_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \overline{V}_{t+1}^i\big(g_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i)\big)\Big] \ , \\ & \text{s.t. } \Theta_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) = r_t^i \ . \end{aligned} \tag{7.76}$$

- If the local resource process is generated by a resource Markovian design $\underline{\mathcal{D}}_{y_0,h,\psi}$ (see Defi-

nition 7.3.6), we look at the following Dynamic Programming equations

$$\overline{V}_t^i(x_t^i, \overline{y}_t) = \min_{\boldsymbol{U}_t^i \in \mathbb{U}_t^i} \mathbb{E}_{\boldsymbol{W}_{t+1}} \Big[L_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \overline{V}_{t+1}^i\big(g_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i), h_t(\overline{y}_t, \boldsymbol{W}_{t+1})\big) \Big]$$
$$\text{s.t. } \Theta_t^i(x_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) = \psi_t^i(\overline{y}_t, \boldsymbol{W}_{t+1}) \ . \tag{7.77}$$

With the above notations, the results stated in Proposition 7.3.4 and Proposition 7.3.10 hold true in the hazard-decision information structure.

7.4. Improving bounds

We have seen in Proposition 7.3.10 that, for a given design, we are able to obtain upper and lower bounds for the global Bellman functions (7.38). By choosing properly the price and resource Markovian designs, we will obtain tighter bounds in Equation (7.58). We will exhibit classes of designs that we can interpret in term of a relaxation (for price design) or a restriction (for resource design) of the original primal problem (7.28), so as to preserve the global bounds (7.15).

7.4.1. Selecting a class of designs for global price processes

Viewing admissible feedbacks as a restriction in the dual. Let $\underline{\mathcal{D}}_{y_0,h,\phi}$ be an admissible price design (see Definition 7.3.9), with fixed dynamics $h_t : \underline{\mathbb{Y}}_t \times \mathbb{W}_{t+1} \to \underline{\mathbb{Y}}_{t+1}$ and initial position $\underline{y}_0 \in \underline{\mathbb{Y}}_0$. We aim at finding the mappings $\phi_s : \underline{\mathbb{Y}}_s \to S_s^\star$ as solutions of the following problem:

$$\begin{aligned} \max_{\phi_0,\cdots,\phi_{T-1}} \min_{\boldsymbol{X},\boldsymbol{U}} \ & \mathbb{E}\Big[\sum_{i=1}^{N} \sum_{t=0}^{T-1} \Big(L_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \phi_t^i(\underline{\boldsymbol{Y}}_t) \cdot \Theta_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i) \Big) + K^i(\boldsymbol{X}_T^i) \Big] \ , \\ & \text{w.r.t. } \boldsymbol{X}_{t+1}^i = g_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) \ , \ \ \boldsymbol{X}_0^i = x_0^i \ , \qquad \forall i \ , \ \forall t, \\ & \qquad \underline{\boldsymbol{Y}}_{t+1} = h_t(\underline{\boldsymbol{Y}}_t, \boldsymbol{W}_{t+1}) \ , \ \ \boldsymbol{Y}_0 = y_0 \ , \qquad \forall t, \\ & \qquad \sigma(\boldsymbol{U}_t^i) \subset \mathcal{G}_t^i \ , \ \ \phi_t(\underline{\boldsymbol{Y}}_t) \in S_t^\star \ , \qquad \forall i \ , \ \forall t. \end{aligned} \tag{7.78}$$

By solving Problem (7.78), we will obtain the tightest lower bounds in Equation (7.58), for given dynamics $h_t : \underline{\mathbb{Y}}_t \times \mathbb{W}_{t+1} \to \underline{\mathbb{Y}}_{t+1}$ and initial position $\underline{y}_0$. Furthermore, we are able to interpret Problem (7.78) as the dual of a primal problem corresponding to a relaxation of the original problem (7.28).

We suppose that for all t the set S_t in (7.78) is a closed convex cone — as we will deal with conditional expectations. We introduce the following set:

$$\boldsymbol{\mathcal{P}} = \{ \boldsymbol{\lambda} \in \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P}, \Xi^\star) \mid \boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}^\star \ , \ \ \mathbb{E}[\boldsymbol{\lambda}_t \mid \underline{\boldsymbol{Y}}_t] = \boldsymbol{\lambda}_t \ , \ \ \forall t \in \{0, .., T-1\} \} \ . \tag{7.79}$$

In (7.79), we restrict to square-integrable stochastic processes to ensure that conditional expectations work fine.

As the proof of the next proposition relies on the Doob lemma, the result holds true only for *measurable* feedback mappings $\phi_s : \underline{\mathbb{Y}}_s \to S_s^\star$.

Proposition 7.4.1. *Let $\boldsymbol{\lambda} \in \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P}, \Xi^\star)$ be a $\underline{\mathcal{D}}_{y_0,h,\phi}$-Markovian design. We suppose that the mappings $\phi_0, \cdots, \phi_{T-1}$ of $\underline{\mathcal{D}}_{y_0,h,\phi}$ are measurable, the spaces $\Xi_0, \cdots, \Xi_{T-1}$ and $S_0, \cdots, S_{T-1}$ defined in* (7.22) *are respectively separable complete metric spaces and closed convex cones. We suppose also that for all $t \in \{0, .., T-1\}$, the random variable $\Theta_t(\boldsymbol{X}_t, \boldsymbol{U}_t) \in \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P}, \Xi_t)$ is*

square integrable. Then, Problem (7.78) *is equivalent to*

$$\begin{aligned}\max_{\boldsymbol{\lambda}\in\boldsymbol{\mathcal{S}}^\star}\ \min_{\boldsymbol{X},\boldsymbol{U}}\ & \mathbb{E}\Big[\sum_{i=1}^{N}\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)+\boldsymbol{\lambda}_t^i\cdot\mathbb{E}[\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)\mid\underline{\boldsymbol{Y}}_t]+K^i(\boldsymbol{X}_T^i)\Big]\ ,\\ & s.t.\ \boldsymbol{X}_{t+1}^i=g_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)\ ,\ \ \boldsymbol{X}_0^i=x_0^i,\quad \forall i\ ,\ \forall t,\\ & \quad\ \underline{\boldsymbol{Y}}_{t+1}=h_t(\underline{\boldsymbol{Y}}_t,\boldsymbol{W}_{t+1})\ ,\ \ \underline{\boldsymbol{Y}}_0=y_0,\qquad\qquad\ \forall t,\\ & \quad\ \sigma(\boldsymbol{U}_t^i)\subset\mathcal{G}_t^i,\qquad\qquad\qquad\qquad\ \forall i\ ,\ \forall t.\end{aligned}\tag{7.80}$$

Proof. For all $t\in\{0,..,T-1\}$ we have $\boldsymbol{\lambda}_t=\phi_t(\underline{\boldsymbol{Y}}_t)$. Thus, the process $\boldsymbol{\lambda}$ is measurable by $\underline{\boldsymbol{Y}}$, and we have

$$\boldsymbol{\lambda}_t=\mathbb{E}[\boldsymbol{\lambda}_t\mid\underline{\boldsymbol{Y}}_t]\ ,\ \ \forall t\in\{0,..,T-1\}\ .\tag{7.81}$$

We consider the set $\boldsymbol{\mathcal{P}}$, as defined in Equation (7.79). Then, by replacing the measurable mappings $\phi_0,\cdots,\phi_{T-1}$ by random variables $\boldsymbol{\lambda}_0,\cdots,\boldsymbol{\lambda}_{T-1}$ measurable w.r.t. $\underline{\boldsymbol{Y}}$, Problem (7.78) rewrites as

$$\max_{\boldsymbol{\lambda}\in\boldsymbol{\mathcal{P}}}\ \min_{\boldsymbol{X},\boldsymbol{U}}\ \mathbb{E}\Big[\sum_{i=1}^{N}\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)+\boldsymbol{\lambda}_t^i\cdot\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)+K^i(\boldsymbol{X}_T^i)\Big]\ ,\tag{7.82a}$$

$$\text{s.t. } \boldsymbol{X}_{t+1}^i=g_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)\ ,\ \ \boldsymbol{X}_0^i=x_0^i\ ,\qquad \forall i\ ,\ \forall t,\tag{7.82b}$$

$$\underline{\boldsymbol{Y}}_{t+1}=h_t(\underline{\boldsymbol{Y}}_t,\boldsymbol{W}_{t+1})\ ,\ \ \underline{\boldsymbol{Y}}_0=y_0\ ,\qquad \forall t,\tag{7.82c}$$

$$\sigma(\boldsymbol{U}_t^i)\subset\mathcal{G}_t^i\ ,\qquad \forall i\ ,\ \forall t.\tag{7.82d}$$

Furthermore, we know that, for all $t\in\{0,..,T-1\}$, as the process $\boldsymbol{\lambda}^i$ is $\underline{\boldsymbol{Y}}$ measurable:

$$\begin{aligned}\mathbb{E}\big[\boldsymbol{\lambda}_t^i\cdot\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)\big]&=\mathbb{E}\big[\mathbb{E}[\boldsymbol{\lambda}_t^i\mid\underline{\boldsymbol{Y}}_t]\cdot\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)\big]\\ &=\mathbb{E}\big[\boldsymbol{\lambda}_t^i\cdot\mathbb{E}[\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)\mid\underline{\boldsymbol{Y}}_t]\big]\ .\end{aligned}\tag{7.83}$$

The second equality holds true because the conditional expectation is self-adjoint on $\mathbb{L}^2(\Omega,\mathcal{F},\mathbb{P},\Xi)$. We replace $\mathbb{E}\big[\boldsymbol{\lambda}_t^i\cdot\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i)\big]$ by its expression (7.83) in Problem (7.82). We deduce that an optimal solution of (7.82) allows to find an admissible solution $\boldsymbol{\lambda}_0,\cdots,\boldsymbol{\lambda}_{T-1}$ of Problem (7.80).

Conversely, let $\boldsymbol{\lambda}\in\mathbb{L}^2(\Omega,\mathcal{F},\mathbb{P},\Xi^\star)$ be a solution of Problem (7.80). By using the relation (7.83), we deduce that the process

$$\mathbb{E}[\boldsymbol{\lambda}_0\mid\underline{\boldsymbol{Y}}_0],\cdots,\mathbb{E}[\boldsymbol{\lambda}_{T-1}\mid\underline{\boldsymbol{Y}}_{T-1}]\tag{7.84}$$

is also an optimal solution of Problem (7.80). Thus, we are able to find an optimal solution that is $\underline{\boldsymbol{Y}}$-measurable. We note by $\boldsymbol{\mu}$ the process defined by

$$\boldsymbol{\mu}_t=\mathbb{E}[\boldsymbol{\lambda}_t\mid\underline{\boldsymbol{Y}}_t]\ ,\ \ \forall t\in\{0,..,T-1\}\ .\tag{7.85}$$

By applying Doob-Dynkin lemma (Dellacherie and Meyer, 1975, Chapter 1, p. 18) to the process $\boldsymbol{\mu}$, there exist measurable mappings $\phi_0,\cdots,\phi_{T-1}$ such that

$$\boldsymbol{\mu}_t=\phi_t(\underline{\boldsymbol{Y}}_t)\ ,\ \ \phi_t(\underline{\boldsymbol{Y}}_t)\in S^\star\ ,\ \ \forall t\in\{0,..,T-1\}\ .\tag{7.86}$$

Then, the mappings $\phi_0,\cdots,\phi_{T-1}$ are an admissible solution of Problem (7.78).

Hence the equivalence, as Problems (7.78) and (7.80) have the same objective. □

Remark 7.4.2. *Under proper assumptions, Problem* (7.80) *can be interpreted as the dual problem*

of:

$$\min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\Big[\sum_{i=1}^{N}\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i) + K^i(\boldsymbol{X}_T^i)\Big]\ , \tag{7.87a}$$

$$s.t.\ \boldsymbol{X}_{t+1}^i = g_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)\ ,\ \ \boldsymbol{X}_0^i = x_0^i\ , \qquad \forall i\ ,\ \forall t, \tag{7.87b}$$

$$\underline{\boldsymbol{Y}}_{t+1} = h_t(\underline{\boldsymbol{Y}}_t,\boldsymbol{W}_{t+1})\ ,\ \ \underline{\boldsymbol{Y}}_0 = y_0\ , \qquad \forall t, \tag{7.87c}$$

$$\sigma(\boldsymbol{U}_t^i) \subset \mathcal{G}_t^i\ , \qquad \forall i\ ,\ \forall t, \tag{7.87d}$$

$$\big(\mathbb{E}[\Theta_t^1(\boldsymbol{X}_t^1,\boldsymbol{U}_t^1)\mid \boldsymbol{Y}_t],\cdots,\mathbb{E}[\Theta_t^N(\boldsymbol{X}_t^N,\boldsymbol{U}_t^N)\mid \boldsymbol{Y}_t]\big) \in -S_t\ , \qquad \forall t. \tag{7.87e}$$

We note that Problem (7.87) *is a relaxation of Problem* (7.28)*. We refer to (Leclère, 2014, Chapter 8) for a proof.* ♢

7.4.2. Selecting a class of designs for global resource processes

We proceed in a similar manner to optimize the admissible Markovian design of resource processes $\boldsymbol{R}$. We look for admissible mappings $\psi_s : \overline{\mathbb{Y}}_s \to -S_s$ minimizing the problem:

$$\min_{\psi_0,\cdots,\psi_{T-1}} \min_{\boldsymbol{X}^i,\boldsymbol{U}^i} \mathbb{E}\Big[\sum_{i=1}^{N}\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i) + K^i(\boldsymbol{X}_T^i)\Big]\ , \tag{7.88a}$$

$$\text{w.r.t.}\quad \boldsymbol{X}_{t+1}^i = g_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)\ ,\ \ \boldsymbol{X}_0^i = x_0^i, \qquad \forall i\ ,\ \forall t, \tag{7.88b}$$

$$\overline{\boldsymbol{Y}}_{t+1} = h_t(\overline{\boldsymbol{Y}}_t,\boldsymbol{W}_{t+1})\ ,\ \ \overline{\boldsymbol{Y}}_0 = y_0, \qquad \forall i\ ,\ \forall t, \tag{7.88c}$$

$$\sigma(\boldsymbol{U}_t^i) \subset \mathcal{G}_t^i\ ,\ \ \psi_t(\overline{\boldsymbol{Y}}_t) \in -S_t, \qquad \forall i\ ,\ \forall t, \tag{7.88d}$$

$$\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i) = \psi_t^i(\overline{\boldsymbol{Y}}_t), \qquad \forall i\ ,\ \forall t. \tag{7.88e}$$

By solving Problem (7.88), we will obtain the tightest upper bound in Equation (7.58), for given dynamics $h_t : \overline{\mathbb{Y}}_t \times \mathbb{W}_{t+1} \to \overline{\mathbb{Y}}_{t+1}$ and initial position $\overline{y}_0$. We are able to interpret Problem (7.88) as a restriction of the original problem (7.28).

Proposition 7.4.3. *Let $\boldsymbol{R}$ be a $\overline{\mathcal{D}}_{y_0,h,\psi}$-Markovian design. We suppose that the mappings $\psi_0,\cdots,\psi_{T-1}$ of $\overline{\mathcal{D}}_{y_0,h,\psi}$ are measurable, the spaces $\Xi_0,\cdots,\Xi_{T-1}$ and $S_0,\cdots,S_{T-1}$ defined in* (7.22) *are respectively separable complete metric spaces and closed convex cones. Then, Problem* (7.88) *is equivalent to*

$$\min_{\boldsymbol{R}\in-\underline{\boldsymbol{\mathcal{S}}}} \min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\Big[\sum_{i=1}^{N}\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i) + K^i(\boldsymbol{X}_T^i)\Big]\ , \tag{7.89a}$$

$$w.r.t.\quad \boldsymbol{X}_{t+1}^i = g_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_{t+1}^i)\ ,\ \ \boldsymbol{X}_0^i = x_0^i, \qquad \forall i\ ,\ \forall t, \tag{7.89b}$$

$$\overline{\boldsymbol{Y}}_{t+1} = h_t(\overline{\boldsymbol{Y}}_t,\boldsymbol{W}_{t+1})\ ,\ \ \overline{\boldsymbol{Y}}_0 = y_0\ , \qquad \forall t, \tag{7.89c}$$

$$\sigma(\boldsymbol{U}_t^i) \subset \mathcal{G}_t^i, \qquad \forall i\ ,\ \forall t, \tag{7.89d}$$

$$\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i) = \boldsymbol{R}_t^i, \qquad \forall i\ ,\ \forall t, \tag{7.89e}$$

with

$$\underline{\boldsymbol{\mathcal{S}}} = \{\boldsymbol{R}\in\boldsymbol{\mathcal{S}} \mid \mathbb{E}[\boldsymbol{R}_t\mid\overline{\boldsymbol{Y}}_t] = \boldsymbol{R}_t\ ,\ \forall t\in\{0,..,T-1\}\}\ . \tag{7.90}$$

Proof. Let $\psi_0,\cdots,\psi_{T-1}$ be the solution of Problem (7.88) and $\boldsymbol{R}$ the corresponding $\overline{\mathcal{D}}_{y_0,h,\psi}$-Markovian design. The stochastic process $\boldsymbol{R}$ is measurable w.r.t. $\overline{\boldsymbol{Y}}$:

$$\boldsymbol{R}_t = \mathbb{E}[\boldsymbol{R}_t\mid\overline{\boldsymbol{Y}}_t]\ . \tag{7.91}$$

Thus, $\boldsymbol{R}$ is an admissible solution for Problem (7.89).

By using the Doob lemma (Dellacherie and Meyer, 1975, Chapter 1, p. 18), we prove that for $\boldsymbol{R}$ solution of Problem (7.89), there exists measurable mappings $\psi_0, \cdots, \psi_{T-1}$ such that $\boldsymbol{R}_t = \psi_t(\overline{\boldsymbol{Y}}_t)$ for all $t \in \{0, .., T-1\}$. As $\boldsymbol{R} \in -\boldsymbol{\mathcal{S}}$, we have that for all t, $\text{im}(\psi_t) \subset -S_t$. Thus, the mappings $\psi_0, \cdots, \psi_{T-1}$ are an admissible solution of Problem (7.88).

Hence the equivalence, as Problems (7.88) and (7.89) have the same objective. □

As $\underline{\boldsymbol{\mathcal{S}}} \subset \boldsymbol{\mathcal{S}}$, we note that Problem (7.89) is a restriction of Problem (7.28).

7.5. Discussion

We have presented in Section 7.2 a formalism to decompose optimization problems by prices and by resources. Depending on the decomposition scheme, we have obtained upper and lower bounds for the original problem. We have applied this formalism to multistage stochastic problems, and proved in Section 7.3 that, under some assumptions, we were able to bound the Bellman value functions of the original problem up and down for all time t. In Section 7.4 we have been able to obtain tighter bounds by choosing appropriately the Markovian designs used to fetch the price and resource processes. As we considered multistage stochastic problems, the information structure played a key role in the design of the decomposition schemes.

The question of the choice of the dynamics $h_t : \mathbb{Y}_t \times \mathbb{W}_{t+1} \to \mathbb{Y}_{t+1}$ in the price Markovian design (see Definition 7.3.5) and in the resource Markovian design (see Definition 7.3.6) remains open. In Strugarek (2006), the author gets a closed-form dynamics for the optimal price processes for a particular problem. However, the result was not extended to more generic problems.

In Chapter 8, we will describe a case study to apply the theoretical results presented in this chapter. We will prove that the formalism introduced in Section 7.2 allows to apply decomposition algorithms onto problems formulated on a graph, where the local problems are set both on nodes and on edges. We will illustrate the effectiveness of price and resource decompositions with numerical applications.

Chapter 8.

Optimal management of district microgrids

Contents

8.1. Introduction

Whereas the presence of stocks implies a temporal coupling in energy systems, the presence of different interconnected units induces a spatial coupling between different units. Spatial decomposition methods aim at breaking the spatial coupling to obtain local decoupled subproblems, easier to solve. Once the subproblems decomposed, we are able to solve them locally by temporal decomposition, that is, by Dynamic Programming. The spatial decomposition of large-scale optimization problems was studied in Cohen (1980), and extended to open-loop stochastic optimization problems in Cohen and Culioli (1990). This study was extended to the closed-loop stochastic case in Barty et al. (2010b), giving a new algorithm called Dual Approximate Dynamic Programming (DADP). DADP was applied to distributed unit-commitment problems in Girardeau (2010), where a unique spatial coupling constraint was considered. Then, DADP was applied with more complex

coupling constraints in Alais (2013) and Leclère (2014), where the authors considered large-scale dams-valley problems, each dam being affected by the dams upstream via their turbinated flows. We have handled in Chapter 7 generic coupling constraints, and we now aim at applying the theoretical results to a particular microgrid problem.

We tackle here a district microgrid with different prosumers exchanging energy via a local network. A broad overview of the emergence of consumer-centric electricity markets is given in Pinson et al. (2018). Some local units are able to produce their own energy with some solar panels, so as to satisfy their needs and export the surplus to other consumers. Other prosumers are equipped with batteries to store energy when necessary. The exchanges through the network are modeled as a network flow problem on a graph. We suppose that the system is impacted by uncertainties, either in production (e.g. renewable units) or in demands (e.g. local electrical demands).

Thus, the global problem formulates naturally as a sum of local multistage stochastic optimization subproblems coupled together via a global network constraint. We refer to Mahey et al. (2017) for a previous application of decomposition methods to such problems.

In this chapter, we write a global optimization problem on a graph in Section 8.2 and detail how to decompose the problem node by node and edge by edge in Section 8.3. Once the problem decomposed, we apply price and resource decomposition algorithms to find the most appropriate price and resource process among a given design in Section 8.4. We apply these decomposition algorithms to the district microgrid problem. We give numerical results comparing the price and resource decomposition algorithms with the well-known Stochastic Dual Dynamic Programming in Section 8.5. Eventually, we sketch some hints to extend the decomposition schemes beyond the classical price and resource decomposition algorithms in Section 8.6.

8.2. Stocks and flows global optimization problem on a graph

Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, with $\mathcal{V}$ the set of nodes and $\mathcal{E}$ the set of edges. We denote by N the number of nodes, and by L the number of edges.

We first detail in §8.2.1 the different flows occurring in the graph and the coupling existing between edges and nodes flows. Each node comprises local stocks, and one can formulate a local multistage stochastic optimization problem, as explained in §8.2.2. In §8.2.3 we handle the global coupling constraints induced by the graph and gather the local nodal subproblems inside a global optimization problem.

8.2.1. Exchanging flows through edges

Flows are transported through the graph via the edges, each edge ℓ transporting a flow q^ℓ and each node $i \in \{1, .., N\}$ importing a flow f^i.

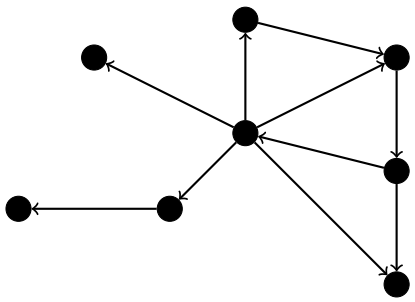


Figure 8.1.: A graph $G = (\mathcal{V}, \mathcal{E})$.

The *node injection* f^i and the *edge flows* q^ℓ are related via a balance equation. The sum of the algebraic edge flows arriving at a particular node i is equal to the injection flow f^i:

$$f^i = \sum_{\ell \in \mathcal{E}^i_+} q^\ell - \sum_{\ell \in \mathcal{E}^i_-} q^\ell \ , \ \ \forall i \in \{1, .., N\} \ , \tag{8.1}$$

where $\mathcal{E}^i_+$ indexes the edges arriving to node i and $\mathcal{E}^i_-$ indexes the edges departing from node i.

Equation (8.1) is Kirchhoff's Current Law, relating the nodal flows $f = (f^1, \cdots, f^N)$ with the algebraic edge flows $q = (q^1, \cdots, q^L)$. It writes in matrix form as

$$Aq + f = 0 \ , \tag{8.2}$$

where $A \in \mathbb{R}^{N \times L}$ is the node-edge incidence matrix of the graph $G = (\mathcal{V}, \mathcal{E})$ (see Annex 8.7.1 for further details).

8.2.2. Local costs on nodes and edges

We first detail the nodal subproblems corresponding to each node $i \in \{1, .., N\}$, and then give the costs on the edges $\ell \in \{1, .., L\}$.

8.2.2.1. Local cost on each node

The graph $G = (\mathcal{V}, \mathcal{E})$ connects N *nodal subproblems*. For node $i \in \{1, .., N\}$, the nodal subproblem is the minimization of a functional $J^i_\mathcal{V}(\boldsymbol{F}^i)$ depending on the injection flow process $\boldsymbol{F}^i = (\boldsymbol{F}^i_0, \cdots, \boldsymbol{F}^i_{T-1})$ arriving at node i between time 0 and $T-1$.

Let $\{\mathbb{X}^i_t\}_{t \in \{0,..,T\}}$, $\{\mathbb{U}^i_t\}_{t \in \{0,..,T-1\}}$ $\{\mathbb{W}^i_t\}_{t \in \{1,..,T\}}$ and $\{\mathbb{G}^i_t\}_{t \in \{0,..,T-1\}}$ be sequences of measurable spaces.

The optimal nodal cost $J^i_\mathcal{V}$ is given by:

$$J^i_\mathcal{V}(\boldsymbol{F}^i) = \min_{\boldsymbol{X}^i, \boldsymbol{U}^i} \ \mathbb{E}\big[\sum_{t=0}^{T-1} L^i_t(\boldsymbol{X}^i_t, \boldsymbol{U}^i_t, \boldsymbol{W}^i_{t+1}) + K^i(\boldsymbol{X}^i_T) \big] \ , \tag{8.3a}$$

$$\text{s.t. } \boldsymbol{X}^i_{t+1} = g^i_t(\boldsymbol{X}^i_t, \boldsymbol{U}^i_t, \boldsymbol{W}^i_{t+1}) \ , \ \ \boldsymbol{X}^i_0 = x^i_0 \ , \tag{8.3b}$$

$$\Delta^i_t(\boldsymbol{X}^i_t, \boldsymbol{U}^i_t, \boldsymbol{W}^i_{t+1}, \boldsymbol{F}^i_t) \in \Gamma_t \ , \tag{8.3c}$$

$$\sigma(\boldsymbol{U}^i_t) \subset \sigma(\boldsymbol{W}_1, \cdots, \boldsymbol{W}_t, \boldsymbol{W}_{t+1}) \ , \tag{8.3d}$$

with $\boldsymbol{X}^i, \boldsymbol{U}^i$ local state and control processes, $\boldsymbol{W}^i$ local uncertainty process, $L^i_t : \mathbb{X}^i_t \times \mathbb{U}^i_t \times \mathbb{W}^i_{t+1} \to \mathbb{R}$ local instantaneous costs and $g^i_t : \mathbb{X}^i_t \times \mathbb{U}^i_t \times \mathbb{W}^i_{t+1} \to \mathbb{X}^i_{t+1}$ local dynamics (see §7.2.3 for further details). We note by $\Delta^i_t : \mathbb{X}^i_t \times \mathbb{U}^i_t \times \mathbb{W}^i_{t+1} \times \mathbb{F}^i_t \to \mathbb{G}^i_t$ the constraint between the state x^i_t, the control u^i_t, the injection flow f^i_t and by $\Gamma_t \subset \mathbb{G}^i_t$ a given admissible set.

Throughout this chapter, we will suppose that all decisions follow the hazard-decision information structure, as specified in §7.3.5:

8.2.2.2. Local transportation cost on each edge

We now consider the edge cost arising at each edge $\ell \in \{1, .., L\}$. The edge cost $J^\ell_\mathcal{E}(\boldsymbol{Q}^\ell)$ depends on the flows process $\boldsymbol{Q}^\ell = (\boldsymbol{Q}^\ell_0, \cdots, \boldsymbol{Q}^\ell_{T-1})$ transported through edge ℓ between time 0 and $T-1$, and writes

$$J^\ell_\mathcal{E}(\boldsymbol{Q}^\ell) = \mathbb{E}\big[\sum_{t=0}^{T-1} l^\ell_t(\boldsymbol{Q}^\ell_t) \big] \ , \tag{8.4}$$

where $l_t^\ell : \mathbb{R} \to \mathbb{R}$ are real valued functions (e.g. quadratic). The cost l_t^ℓ can be engendered by a difference in pricing, a fixed toll between the different nodes, or by the energy losses through the network.

8.2.3. Formulating a global optimization problem on the graph

Each nodal problem (8.3) formulates as a multistage stochastic optimization problem whose solution depends on a nodal flow process

$$\boldsymbol{F}^i = (\boldsymbol{F}_0^i, \cdots, \boldsymbol{F}_{T-1}^i) \,, \tag{8.5}$$

corresponding to the energy exchanges with the graph. We now aim to couple all nodal subproblems together by considering the coupling constraints induced by the graph.

8.2.3.1. Nodal global cost

We aggregate all nodal subproblems (8.3) in a single criterion. Let

$$\boldsymbol{F} = (\boldsymbol{F}^1, \cdots, \boldsymbol{F}^N) \,, \tag{8.6}$$

be the *global* injection flows at nodes. The nodal global cost writes as the sum of all nodal costs

$$J_{\mathcal{V}}(\boldsymbol{F}) = \sum_{i=1}^{N} J_{\mathcal{V}}^i(\boldsymbol{F}^i) + \mathbb{I}_{\underline{F}^i \leq \cdot \leq \overline{F}^i}(\boldsymbol{F}^i) \,, \tag{8.7}$$

where $\mathbb{I}_{\underline{F}^i \leq \cdot \leq \overline{F}^i}$ is the characteristic function of the set $[\underline{F}^i, \overline{F}^i]$, added to ensure that all incidence flows are bounded:

$$\underline{F}^i \leq \boldsymbol{F}_t^i \leq \overline{F}^i \,, \quad \forall t \in \{0, .., T-1\} \,. \tag{8.8}$$

8.2.3.2. Edges global cost

The edges global cost aggregates the exchange costs (8.4) through the different edges in the graph

$$J_{\mathcal{E}}(\boldsymbol{Q}) = \mathbb{E}\Big[\sum_{\ell=1}^{L} J_{\mathcal{E}}^\ell(\boldsymbol{Q}^\ell) + \mathbb{I}_{-\overline{Q}^\ell \leq \cdot \leq \overline{Q}^\ell}(\boldsymbol{Q}^\ell) \Big] \,, \tag{8.9}$$

where the characteristic function $\mathbb{I}$ ensures that grid flows are bounded

$$-\overline{Q}^\ell \leq \boldsymbol{Q}_t^\ell \leq \overline{Q}^\ell \,, \;\; \forall t \in \{0, .., T-1\} \,. \tag{8.10}$$

The *global edge cost* $J_{\mathcal{E}}$ is decomposable w.r.t. time and edges

8.2.3.3. Global optimization problem

We have stated a global nodal criterion (8.7) and a global edge criterion (8.9), both depending on flows processes coupled by the coupling (8.2).

We are now able to formulate a *global* optimization problem as

$$V^\sharp = \min_{\boldsymbol{F}, \boldsymbol{Q}} \;\; J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) \tag{8.11a}$$

$$\text{s.t. } A\boldsymbol{Q}_t + \boldsymbol{F}_t = 0 \,, \;\; \forall t \in \{0, .., T-1\} \,, \tag{8.11b}$$

where

- $J_{\mathcal{V}} : \mathbb{R}^{NT} \to \mathbb{R}$ is the criterion (8.7), aggregating all nodal costs,
- $J_{\mathcal{E}} : \mathbb{R}^{LT} \to \mathbb{R}$ is the criterion (8.9), aggregating all edges costs,
- $A \in \mathbb{R}^{N \times L}$ is the node-edge incidence matrix of the graph $G = (\mathcal{V}, \mathcal{E})$,
- $\boldsymbol{F} = (\boldsymbol{F}^1, \cdots, \boldsymbol{F}^N)$ are the exchange flows at the different nodes of the graph,
- $\boldsymbol{Q} = (\boldsymbol{Q}^1, \cdots, \boldsymbol{Q}^L)$ are the flows through the different edges of the graph.

Problem (8.11) couples two criteria which were initially independent through the coupling constraint (8.11b). As the resulting criterion is additive and the coupling constraint (8.11b) is affine, this problem has a nice form to use decomposition-coordination methods.

8.2.3.4. Introducing the decomposition formalism

We introduced in Chap. 7 a generic framework to bound a global problem by decomposing it in smaller local subproblems, easier to solve. The global problem (7.28) displayed global coupling constraints (7.28d). We prove here that Problem (8.11) follows the same formalism.

Proposition 8.2.1. *Problem* (8.11) *is equivalent to Problem* (7.28)*, with the global constraints*

$$(\boldsymbol{F}_t, \boldsymbol{Q}_t) \in S_t \quad \mathbb{P}\text{-}a.s. \ , \ \ \forall t \in \{0, .., T-1\} \ , \tag{8.12}$$

with the convex cone S_t *of* $\mathbb{R}^N \times \mathbb{R}^L$

$$S_t = \{(f_t, q_t) \in \mathbb{R}^N \times \mathbb{R}^L \mid Aq_t + f_t = 0\} \ . \tag{8.13}$$

Proof. The injection flow $\boldsymbol{F}^i$ are a local decision for all nodes $i \in \{1, .., N\}$, thus the local node problems (8.3) formulate in a similar fashion as in §7.2.3. Similarly, the edge problems (8.4) are local multistage stochastic problems without stocks with local decisions corresponding to the edge flow $\boldsymbol{Q}^\ell$ for all edge $\ell \in \{1, .., L\}$.

Thus, the convex cone S_t defined in (8.13) is the coupling set between the local node problems (8.3) and edge problems (8.4). The coupling equation $A\boldsymbol{Q}_t + \boldsymbol{F}_t = 0$ becomes a special case of the generic coupling constraint (7.28d) introduced in §7.2.3. □

It can easily be seen that the dual cone of the cone S_t defined in (8.13) writes for all time $t \in \{0, .., T-1\}$:

$$S_t^\star = \{(\lambda_t, \mu_t) \in \mathbb{R}^N \times \mathbb{R}^L \mid A^\top \lambda_t - \mu_t = 0\} \ . \tag{8.14}$$

We have the following relation.

Proposition 8.2.2. *For all* $(\lambda, \mu) \in S_t^\star$*, we have*

$$\lambda \cdot f + \mu \cdot q = \lambda \cdot (Aq + f) \ , \ \ \forall (f, q) \in S_t \ , \tag{8.15}$$

where $(u, v) \mapsto u \cdot v$ *is the usual Euclidian scalar product on* $\mathbb{R}^N$*.*

Proof. Let $(\lambda, \mu) \in S_t^\star$. We have

$$\lambda \cdot f + \mu \cdot q = \lambda \cdot f + (A^\top \lambda) \cdot q = \lambda \cdot (Aq + f) \ . \tag{8.16}$$

Hence the result. □

8.3. Mixing nodal and time decomposition

The direct resolution of Problem (8.11) is generally out of reach, as it writes as a sum of coupled multistage stochastic optimization subproblems. However, its structure renders it suitable for the decomposition schemes introduced in Chapter 7.

The decomposition of Problem (8.11) arises in two steps: first, in §8.3.1 we decouple Problem (8.11) *spatially* into nodal and edge subproblems; then we decompose in §8.3.2 the nodal subproblems (8.3) *temporally* by Dynamic Programming, as explained in Section 7.3.

8.3.1. Decomposition of the global problem

We follow the procedure introduced in Section 7.3 and decompose the coupling constraint (8.11b) spatially by fixing a global price process $\boldsymbol{\lambda}$ or a global resource process $\boldsymbol{R}$.

8.3.1.1. Decomposition of the price value function

Let $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_0, \cdots, \boldsymbol{\lambda}_{T-1}) \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$ (see Definition 7.2.2 for the definition of $\mathcal{F}$-adapted stochastic processes space $\mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$) be a *nodal price process* and $\boldsymbol{\mu} = (\boldsymbol{\mu}^1, \cdots, \boldsymbol{\mu}^N) \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^L)$ be a *edge price process*. We reconsider the admissible sets $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{S}}^\star$ defined in Definition 7.2.3.

We define the global price value function:

$$\underline{V}[\boldsymbol{\lambda}, \boldsymbol{\mu}] = \min_{\boldsymbol{F}, \boldsymbol{Q}} \; J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \big\langle \boldsymbol{\lambda} \, , \boldsymbol{F} \big\rangle + \big\langle \boldsymbol{\mu} \, , \boldsymbol{Q} \big\rangle \, , \tag{8.17}$$

with $\big\langle \cdot, \cdot \big\rangle$ being a scalar product between stochastic processes, writing, for all $\boldsymbol{\lambda} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$ and $\boldsymbol{F} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$,

$$\big\langle \boldsymbol{\lambda} \, , \boldsymbol{F} \big\rangle = \mathbb{E}\Big[\sum_{t=0}^{T-1} \boldsymbol{\lambda}_t \cdot \boldsymbol{F}_t \Big] \, , \tag{8.18}$$

where $(u, v) \in \mathbb{R}^N \times \mathbb{R}^N \mapsto u \cdot v$ is the usual Euclidian scalar product on $\mathbb{R}^N$. We suppose that measurability and integrability assumptions hold, so that the expression in (8.18) makes sense [1].

Proposition 8.3.1. *Let $\boldsymbol{\lambda} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$ be a nodal price process and $\boldsymbol{\mu} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^L)$ be a edge price process. The global price value function* (8.17) *is equal to*

$$\underline{V}[\boldsymbol{\lambda}, \boldsymbol{\mu}] = \min_{\boldsymbol{F}, \boldsymbol{Q}} \; J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \big\langle \boldsymbol{\lambda} \, , A\boldsymbol{Q} + \boldsymbol{F} \big\rangle \, , \tag{8.19}$$

which decomposes naturally in a nodal problem

$$\underline{V}_{\mathcal{V}}[\boldsymbol{\lambda}] = \min_{\boldsymbol{F}} \; \big\{ J_{\mathcal{V}}(\boldsymbol{F}) + \big\langle \boldsymbol{\lambda} \, , \boldsymbol{F} \big\rangle \big\} \, , \tag{8.20a}$$

and a edge problem

$$\underline{V}_{\mathcal{E}}[\boldsymbol{\lambda}] = \min_{\boldsymbol{Q}} \; \big\{ J_{\mathcal{E}}(\boldsymbol{Q}) + \big\langle \boldsymbol{\lambda} \, , A\boldsymbol{Q} \big\rangle \big\} \, . \tag{8.20b}$$

Proof. Using Proposition 8.2.2, we have

$$\big\langle \boldsymbol{\lambda} \, , \boldsymbol{F} \big\rangle + \big\langle \boldsymbol{\mu} \, , \boldsymbol{Q} \big\rangle = \big\langle \boldsymbol{\lambda} \, , A\boldsymbol{Q} + \boldsymbol{F} \big\rangle \, , \;\; \forall (\boldsymbol{F}, \boldsymbol{Q}) \in -\boldsymbol{\mathcal{S}} \, , \;\; \forall (\boldsymbol{\lambda}, \boldsymbol{\mu}) \in \boldsymbol{\mathcal{S}}^\star \, . \tag{8.21}$$

[1] Such assumptions hold for instance if $\boldsymbol{\lambda} \in \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$ and $\boldsymbol{F} \in \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$, or if $\boldsymbol{\lambda} \in \mathbb{L}^1(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$ and $\boldsymbol{F} \in \mathbb{L}^\infty(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$.

Thus, we are able to get rid of the process $\boldsymbol{\mu}$ and consider only the price process $\boldsymbol{\lambda}$ in (8.17) (which is unconstrained). The global price value function (8.17) rewrites

$$\underline{V}[\boldsymbol{\lambda}] = \min_{\boldsymbol{F},\boldsymbol{Q}} \; J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \left\langle \boldsymbol{\lambda} \, , A\boldsymbol{Q} + \boldsymbol{F} \right\rangle . \tag{8.22}$$

Then, we decompose the global price value function (8.19) w.r.t. nodes and edges as

$$\underline{V}[\boldsymbol{\lambda}] = \underline{V}_{\mathcal{V}}[\boldsymbol{\lambda}] + \underline{V}_{\mathcal{E}}[\boldsymbol{\lambda}] \, , \tag{8.23}$$

with $\underline{V}_{\mathcal{V}}[\boldsymbol{\lambda}]$ and $\underline{V}_{\mathcal{E}}[\boldsymbol{\lambda}]$ defined by Equation (8.20). □

We are able to define the *nodal price value functions* corresponding to Equation (7.31):

$$\underline{V}^i_{\mathcal{V}}[\boldsymbol{\lambda}^i] = \min_{\boldsymbol{F}^i} J^i_{\mathcal{V}}(\boldsymbol{F}^i) + \left\langle \boldsymbol{\lambda}^i \, , \boldsymbol{F}^i \right\rangle , \;\; \forall i \in \{1, .., N\} \, , \tag{8.24}$$

so that the price nodal problem rewrites

$$\underline{V}_{\mathcal{V}}[\boldsymbol{\lambda}] = \sum_{i=1}^{N} \underline{V}^i_{\mathcal{V}}[\boldsymbol{\lambda}^i] \, . \tag{8.25}$$

8.3.1.2. Decomposition of the resource value function

Let $\boldsymbol{R} = (\boldsymbol{R}_0, \cdots, \boldsymbol{R}_{T-1}) \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$ be a global resource process. We decompose the global constraints (8.11b) w.r.t. the nodes and the edges as, for all time $t \in \{0, .., T-1\}$,

$$\begin{cases} \boldsymbol{F}_t = \boldsymbol{R}_t \, , \\ A\boldsymbol{Q}_t = -\boldsymbol{R}_t \, . \end{cases} \tag{8.26}$$

We define the *global resource value function* as:

$$\overline{V}[\boldsymbol{R}] = \Big[\min_{\boldsymbol{F}} \; J_{\mathcal{V}}(\boldsymbol{F}) + \min_{\boldsymbol{Q}} J_{\mathcal{E}}(\boldsymbol{Q}) \Big] \tag{8.27a}$$

$$\text{s.t. } A\boldsymbol{Q} + \boldsymbol{R} = 0 \, , \tag{8.27b}$$

$$\boldsymbol{F} - \boldsymbol{R} = 0 \, . \tag{8.27c}$$

Proposition 8.3.2. *Let $\boldsymbol{R} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$ be a resource process. The global resource value function* (8.27) *decomposes w.r.t. the nodes and the edges:*

$$\overline{V}[\boldsymbol{R}] = \overline{V}_{\mathcal{V}}[\boldsymbol{R}] + \overline{V}_{\mathcal{E}}[\boldsymbol{R}] \, , \tag{8.28}$$

with the resource nodal problem

$$\overline{V}_{\mathcal{V}}[\boldsymbol{R}] = \min_{\boldsymbol{F}-\boldsymbol{R}=0} J_{\mathcal{V}}(\boldsymbol{F}) \, , \tag{8.29}$$

and the resource edge problem

$$\overline{V}_{\mathcal{E}}[\boldsymbol{R}] = \min_{A\boldsymbol{Q}+\boldsymbol{R}=0} J_{\mathcal{E}}(\boldsymbol{F}) \, . \tag{8.30}$$

Proof. Straightforward, by definition of the global resource value function (8.27). □

We note that, if $\boldsymbol{R} \notin \mathrm{im}(A)$, we have $\overline{V}[\boldsymbol{R}] = +\infty$ as the constraint $A\boldsymbol{Q} + \boldsymbol{R} = 0$ is not admissible.

We are able to define the *nodal resource value functions* corresponding to Equation (7.32):

$$\begin{aligned} \overline{V}^i_{\mathcal{V}}[\boldsymbol{R}^i] = \min_{\boldsymbol{F}^i} \; & J^i_{\mathcal{V}}(\boldsymbol{F}^i) \; , \;\; \forall i \in \{1,..,N\} \\ \text{s.t. } & \boldsymbol{F}^i - \boldsymbol{R}^i = 0 \; , \end{aligned} \tag{8.31}$$

so that the resource nodal problem rewrites

$$\overline{V}[\boldsymbol{R}] = \sum_{i=1}^{N} \overline{V}^i_{\mathcal{V}}[\boldsymbol{R}^i] \; . \tag{8.32}$$

Remark 8.3.3. *Another resource allocation scheme would be to fix the allocations on edges rather than on nodes. Let $\boldsymbol{R} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^L)$ be a resource process. We decompose Problem* (8.11) *as, for all $t \in \{0,..,T-1\}$,*

$$\begin{cases} \boldsymbol{F}_t = -A\boldsymbol{R}_t \; , \\ \boldsymbol{Q}_t = \boldsymbol{R}_t \; . \end{cases} \tag{8.33}$$

By doing so, we obtain the global resource value function

$$\overline{\overline{V}}[\boldsymbol{R}] = \Big[\min_{\boldsymbol{F}} \;\; J_{\mathcal{V}}(\boldsymbol{F}) + \min_{\boldsymbol{Q}} J_{\mathcal{E}}(\boldsymbol{Q}) \Big] \tag{8.34a}$$

$$s.t. \; \boldsymbol{Q} = \boldsymbol{R} \; , \tag{8.34b}$$

$$\boldsymbol{F} + A\boldsymbol{R} = 0 \; . \tag{8.34c}$$

that is, $\overline{V}[\boldsymbol{R}] = J_{\mathcal{V}}(-A\boldsymbol{R}) + J_{\mathcal{E}}(\boldsymbol{R})$. ◊

8.3.1.3. Upper and lower bounds by spatial decomposition

Applying Proposition 7.2.4 to the price value functions (8.19) and resource value functions (8.27), we obtain that

$$\underline{V}_{\mathcal{V}}[\boldsymbol{\lambda}] + \underline{V}_{\mathcal{E}}[\boldsymbol{\lambda}] \leq V^\sharp \leq \overline{V}_{\mathcal{V}}[\boldsymbol{R}] + \overline{V}_{\mathcal{E}}[\boldsymbol{R}] \; . \tag{8.35}$$

By computing the global price and resource value functions, we are able to bound up and down the optimal value $V^\sharp$ of Problem (8.11).

8.3.2. Temporal decomposition of nodal value functions

We decomposed Problem (8.11) spatially by introducing a global price process and a global resource process. We now adapt the results of §7.3 to compute the price value function (8.19) and resource value function (8.27) by Dynamic Programming.

8.3.2.1. Time decomposition of each nodal price value function

Considering the definition of $J^i_{\mathcal{V}}$ in Equation (8.3), the nodal price value function (8.24) is written in a expended manner:

$$\underline{V}^i_{\mathcal{V}}[\boldsymbol{\lambda}^i](x^i_0) = \min_{\boldsymbol{X}^i, \boldsymbol{U}^i, \boldsymbol{F}^i} \; \mathbb{E}\big[\sum_{t=0}^{T-1} L^i_t(\boldsymbol{X}^i_t, \boldsymbol{U}^i_t, \boldsymbol{W}^i_{t+1}) + \big\langle \boldsymbol{\lambda}^i_t \, , \boldsymbol{F}^i_t \big\rangle + K^i(\boldsymbol{X}^i_T) \big] \; , \tag{8.36a}$$

$$\text{s.t. } \boldsymbol{X}^i_{t+1} = g^i_t(\boldsymbol{X}^i_t, \boldsymbol{U}^i_t, \boldsymbol{W}^i_{t+1}) \; , \;\; \boldsymbol{X}^i_0 = x^i_0 \; , \tag{8.36b}$$

$$\Delta^i_t(\boldsymbol{X}^i_t, \boldsymbol{U}^i_t, \boldsymbol{W}^i_{t+1}, \boldsymbol{F}^i_t) \in \Gamma_t \; , \tag{8.36c}$$

$$\sigma(\boldsymbol{U}^i_t) \subset \mathcal{G}^i_t \; . \tag{8.36d}$$

By Proposition 7.3.7, we know that $\underline{V}^i_{\mathcal{V}}[\boldsymbol{\lambda}^i]$ satisfies a Dynamic Programming equation, provided that the nodal price process $\boldsymbol{\lambda}^i$ is $\underline{\mathcal{D}}_{y_0,h,\phi^i}$-Markovian (see Definition 7.3.6).

8.3.2.2. Time decomposition of each nodal quantity value function

The nodal resource value function (8.31) is rewritten in an expended manner:

$$\overline{V}^i_{\mathcal{V}}[\boldsymbol{R}^i](x^i_0) = \min_{\boldsymbol{X}^i,\boldsymbol{U}^i,\boldsymbol{F}^i} \mathbb{E}\big[\sum_{t=0}^{T-1} L^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t,\boldsymbol{W}^i_{t+1}) + K^i(\boldsymbol{X}^i_T)\big] \;, \tag{8.37a}$$

$$\text{s.t. } \boldsymbol{X}^i_{t+1} = g^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t,\boldsymbol{W}^i_{t+1}) \;,\;\; \boldsymbol{X}^i_0 = x^i_0 \;, \tag{8.37b}$$

$$\Delta^i_t(\boldsymbol{X}^i_t,\boldsymbol{U}^i_t,\boldsymbol{W}^i_{t+1},\boldsymbol{F}^i_t) \in \Gamma_t \;, \tag{8.37c}$$

$$\sigma(\boldsymbol{U}^i_t) \subset \mathcal{G}^i_t \;, \tag{8.37d}$$

$$\boldsymbol{F}^i_t - \boldsymbol{R}^i_t = 0 \;. \tag{8.37e}$$

In a similar manner, we know by Proposition 7.3.8 that $\overline{V}^i_{\mathcal{V}}[\boldsymbol{R}^i]$ satisfies a Dynamic Programming equation, provided that the nodal resource process $\boldsymbol{R}^i$ is $\overline{\mathcal{D}}_{y_0,h,\psi^i}$-Markovian.

8.4. Algorithmic implementation

In Section 8.3 we decomposed Problem (8.11) spatially, then temporally. We now detail how to obtain tighter bounds in Equation (8.35), by following the procedure introduced in §7.4. We will observe that improving optimal feedbacks $\phi_0,\cdots,\phi_{T-1}$ and $\psi_0,\cdots,\psi_{T-1}$ in the price and resource Markovian designs turn to implement a gradient-like algorithm.

We first detail in §8.4.1 the price decomposition algorithm, and then present in §8.4.2 the resource decomposition algorithm.

8.4.1. Price decomposition

First, we detail how to improve the lower bound given by the price value functions in Equation (8.35).

8.4.1.1. Lower bound improvement

Let $\boldsymbol{\lambda}$ be a $\underline{\mathcal{D}}_{y_0,h,\phi}$-Markovian price process. We consider the price value function $\underline{V}[\boldsymbol{\lambda}]$, whose expression is given by Equation (8.19).

We aim at solving Problem (7.78), which rewrites as

$$\max_{\phi_0,\cdots,\phi_{T-1}} \min_{\boldsymbol{F},\boldsymbol{Q}} J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \big\langle \phi(\boldsymbol{Y}) \,, A\boldsymbol{Q} + \boldsymbol{F} \big\rangle \tag{8.38}$$

We reformulate Problem (8.38) as explained in §7.4.1. We know by Proposition 7.4.1 that Problem (8.38) is equivalent to the relaxed problem

$$\max_{\boldsymbol{\lambda}} \min_{\boldsymbol{F},\boldsymbol{Q}} J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \big\langle \boldsymbol{\lambda} \,, \mathbb{E}[A\boldsymbol{Q} + \boldsymbol{F} \mid \boldsymbol{Y}] \big\rangle \;. \tag{8.39}$$

The price value function now writes:

$$\underline{V}[\boldsymbol{\lambda}] = \min_{\boldsymbol{F},\boldsymbol{Q}} J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \big\langle \boldsymbol{\lambda} \,, \mathbb{E}[A\boldsymbol{Q} + \boldsymbol{F} \mid \boldsymbol{Y}] \big\rangle \;. \tag{8.40}$$

Proposition 8.4.1. *We suppose that*

- *Each local noise* $\boldsymbol{W}_t^i$ *in* (8.3) *has a finite support;*
- *The costs* $L_t^i : \mathbb{X}_t^i \times \mathbb{U}_t^i \times \mathbb{W}_{t+1}^i \to \mathbb{R}$ *in* (8.3) *are strongly convex w.r.t.* (x^i, u^i)*;*
- *The edge costs* l_t^ℓ *in* (8.4) *are strongly convex w.r.t.* q^ℓ*;*
- *The local dynamics* g_t^i *in* (8.3) *are affine w.r.t.* (x^i, u^i)*;*
- *There exists an admissible solution* $(\boldsymbol{F}, \boldsymbol{Q})$ *for Problem* (8.40)*.*

Then the functions $J_{\mathcal{V}}$ *in* (8.7) *and* $J_{\mathcal{E}}$ *in* (8.9) *are differentiable w.r.t. the processes* $\boldsymbol{F}$ *and* $\boldsymbol{Q}$ *and the price value function* $\underline{V}[\boldsymbol{\lambda}]$ *is differentiable w.r.t. the price process* $\boldsymbol{\lambda}$*, with gradient*

$$\nabla \underline{V}[\boldsymbol{\lambda}] = \mathbb{E}[A\boldsymbol{Q}^\sharp(\boldsymbol{\lambda}) + \boldsymbol{F}^\sharp(\boldsymbol{\lambda}) \mid \boldsymbol{Y}] \,, \tag{8.41}$$

where $(\boldsymbol{F}^\sharp(\boldsymbol{\lambda}), \boldsymbol{Q}^\sharp(\boldsymbol{\lambda}))$ *is solution of Problem* (8.40)*.*

Proof. As the uncertainties have finite supports and the processes $\boldsymbol{F}$ and $\boldsymbol{Q}$ are measurable w.r.t. the uncertainties $\boldsymbol{W}$, we deduce that $\boldsymbol{F}$ and $\boldsymbol{Q}$ have only a finite number of values. Thus, the discrete values of $\boldsymbol{F}$ and $\boldsymbol{Q}$ lie in finite dimensional Euclidian spaces that we will denote respectively by $\mathbb{R}^{|F|}$ and $\mathbb{R}^{|Q|}$.

As local costs L_t and l_t are strongly convex, we deduce that $J_{\mathcal{V}}$ and $J_{\mathcal{E}}$ are differentiable w.r.t. the finite support coordination processes $\boldsymbol{F}$ and $\boldsymbol{Q}$.

Let $g : \mathbb{R}^{|F|} \times \mathbb{R}^{|Q|} \times \mathbb{R}^{|F|} \to \mathbb{R}$

$$g(\boldsymbol{F}, \boldsymbol{Q}, \boldsymbol{\lambda}) = J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \left\langle \boldsymbol{\lambda} \,, \mathbb{E}[A\boldsymbol{Q} + \boldsymbol{F} \mid \boldsymbol{Y}] \right\rangle \,,$$

be a convex-concave mapping. We have that:

- The mappings $J_{\mathcal{V}}$ and $J_{\mathcal{E}}$ are coercive functions.
- For all $\boldsymbol{\lambda}$, $(\boldsymbol{F}, \boldsymbol{Q}) \mapsto g(\boldsymbol{F}, \boldsymbol{Q}, \boldsymbol{\lambda})$ is continuous w.r.t. $\boldsymbol{F}$ and $\boldsymbol{Q}$.
- For all $\boldsymbol{F}, \boldsymbol{Q}$, the mapping $\boldsymbol{\lambda} \to g(\boldsymbol{F}, \boldsymbol{Q}, \boldsymbol{\lambda})$ is convex, differentiable.
- $\underline{V}[\boldsymbol{\lambda}]$ in (8.39) is finite valued.
- As the local costs L_t^i are strongly convex, the costs $J_{\mathcal{V}}$ and $J_{\mathcal{E}}$ are strongly convex. Thus, the mapping $g(\cdot, \cdot, \boldsymbol{\lambda})$ is minimized at a unique point $(\boldsymbol{F}^\sharp(\boldsymbol{\lambda}), \boldsymbol{Q}^\sharp(\boldsymbol{\lambda}))$.

Then, applying (Hiriart-Urruty and Lemaréchal, 2012, Corollary 4.4.5, p.191), we know that $\underline{V}[\boldsymbol{\lambda}]$ is differentiable at $\boldsymbol{\lambda}$, and

$$\nabla \underline{V}[\boldsymbol{\lambda}] = \mathbb{E}[A\boldsymbol{Q}^\sharp(\boldsymbol{\lambda}) + \boldsymbol{F}^\sharp(\boldsymbol{\lambda}) \mid \boldsymbol{Y}] \,. \tag{8.42}$$

Hence the result. □

8.4.1.2. Price decomposition algorithm

Using Proposition 8.4.1, we are able to solve Problem (8.39) with a gradient ascent algorithm. At iteration k, we suppose given a price process $\boldsymbol{\lambda}^{(k)}$ and a sequence $\{\rho_t^{(k)}\}_{t \in \{0,..,T-1\}}$ of gradient steps. The algorithm is given by:

$$\boldsymbol{F}^{(k+1)} = \arg\min_{\boldsymbol{F}} J_{\mathcal{V}}(\boldsymbol{F}) + \left\langle \boldsymbol{\lambda}^{(k)} \,, \boldsymbol{F} \right\rangle \,, \tag{8.43a}$$

$$\boldsymbol{Q}^{(k+1)} = \arg\min_{\boldsymbol{Q}} J_{\mathcal{E}}(\boldsymbol{Q}) + \left\langle \boldsymbol{\lambda}^{(k)} \,, A\boldsymbol{Q} \right\rangle \,, \tag{8.43b}$$

$$\boldsymbol{\lambda}_t^{(k+1)} = \boldsymbol{\lambda}_t^{(k)} + \rho_t^{(k)} \, \mathbb{E}\big[A\boldsymbol{Q}_t^{(k+1)} + \boldsymbol{F}_t^{(k+1)} \mid \boldsymbol{Y}_t\big] \,, \quad \forall t \in \{0, .., T-1\} \,. \tag{8.43c}$$

Let us describe Algorithm (8.43) step by step.

- The resolution of Equation (8.43a) and Equation (8.43b) decomposes spatially w.r.t. nodes and edges, as explained in §8.3.1. We solve the local nodal problems (8.24) by Dynamic Programming, as the local price process $\boldsymbol{\lambda}^i$ are $\underline{\mathcal{D}}_{y_0,h,\phi^i}$-Markovian. The local edge subproblems are solved directly by standard mathematical programming methods.

- For all time t, updating $\boldsymbol{\lambda}_t$ requires the computation of a conditional expectation

$$\mathbb{E}\big[A\boldsymbol{Q}_t^{(k+1)} + \boldsymbol{F}_t^{(k+1)} \mid \boldsymbol{Y}_t\big] \tag{8.44}$$

  that we usually perform by Monte-Carlo. The gradient $\nabla \underline{V}[\boldsymbol{\lambda}]$ can be used in more sophisticated descent methods (BFGS, Nesterov accelerated gradient) than the classical gradient algorithm presented in (8.43c).

Convergence. Under proper assumptions, Algorithm (8.43) converges.

Proposition 8.4.2. *We assume that:*

- *Each local noise $\boldsymbol{W}^i$ in* (8.3) *has a finite support;*
- *The costs $L_t^i : \mathbb{X}_t^i \times \mathbb{U}_t^i \times \mathbb{W}_{t+1}^i \to \mathbb{R}$ in* (8.3) *are strongly convex w.r.t. (x^i, u^i);*
- *The edge costs l_t^ℓ in* (8.4) *are strongly convex w.r.t. q^ℓ;*
- *The local dynamics g_t^i in* (8.3) *are affine w.r.t. (x^i, u^i);*
- *There exists an admissible solution $(\boldsymbol{F}, \boldsymbol{Q})$ for Problem* (8.40).

Then, the sequence of processes $(\boldsymbol{F}^{(k)}, \boldsymbol{Q}^{(k)})_{k\in\mathbb{N}}$ given by Algorithm (8.43) *converges toward the optimal processes $(\boldsymbol{F}^\sharp, \boldsymbol{Q}^\sharp)$ solution of the relaxed problem* (8.39).

Proof. See Leclère (2014). □

8.4.2. Resource allocation

We now focus on the improvement of the upper bound given by the resource value functions in Equation (8.35).

8.4.2.1. Upper bound improvement

Let $\boldsymbol{R}$ be a $\mathcal{D}_{\underline{y_0},h,\psi}$-Markovian process, as defined in Definition 7.3.5. We consider the resource value function $\overline{V}[\boldsymbol{R}]$, defined in Equation (8.27). Problem (7.88) rewrites

$$\min_{\psi_0,\cdots,\psi_{T-1}} \overline{V}[\psi(\boldsymbol{Y})]\,. \tag{8.45}$$

Using the interpretation introduced in §7.4.2, we know that Problem (8.45) interprets as

$$\begin{aligned} \min_{\boldsymbol{R}} \min_{\boldsymbol{F},\boldsymbol{Q}} & J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) \\ \text{s.t. } & \boldsymbol{F} = \mathbb{E}[\boldsymbol{R} \mid \boldsymbol{Y}] \\ & A\boldsymbol{Q} = -\mathbb{E}[\boldsymbol{R} \mid \boldsymbol{Y}]\,. \end{aligned} \tag{8.46}$$

The resource value function associated to Problem (8.46) is:

$$\begin{aligned} \overline{V}[\boldsymbol{R}] = \min_{\boldsymbol{F},\boldsymbol{Q}} \; & J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) \\ \text{s.t. } & \boldsymbol{F} = \mathbb{E}[\boldsymbol{R} \mid \boldsymbol{Y}] \\ & A\boldsymbol{Q} = -\mathbb{E}[\boldsymbol{R} \mid \boldsymbol{Y}] \; . \end{aligned} \tag{8.47}$$

We note that $\overline{V}[\boldsymbol{R}] = +\infty$ if $\mathbb{E}[\boldsymbol{R}|\boldsymbol{Y}] \notin \mathrm{im}(A)$, $\mathbb{P}$-a.s..

Proposition 8.4.3. *We suppose that*

- *Each local noise $\boldsymbol{W}^i$ in* (8.3) *has a finite support;*
- *The costs $L_t^i : \mathbb{X}_t^i \times \mathbb{U}_t^i \times \mathbb{W}_{t+1}^i \to \mathbb{R}$ in* (8.3) *are strongly convex w.r.t. (x^i, u^i);*
- *The edge costs l_t^ℓ in* (8.4) *are strongly convex w.r.t. q^ℓ;*
- *The local dynamics g_t^i in* (8.3) *are affine w.r.t. (x^i, u^i);*
- *There exists an admissible solution $(\boldsymbol{F}, \boldsymbol{Q})$ for Problem* (8.47).

Then the functions $J_{\mathcal{V}}$ (8.7) *and $J_{\mathcal{E}}$* (8.9) *are differentiable w.r.t. the processes $\boldsymbol{F}$ and $\boldsymbol{Q}$ and the resource value function $\overline{V}[\boldsymbol{R}]$* (8.47) *is differentiable w.r.t. the resource process $\boldsymbol{R}$. Its gradient writes*

$$\nabla \overline{V}[\boldsymbol{R}] = \mathbb{E}[\boldsymbol{\lambda} + \boldsymbol{\mu} \mid \boldsymbol{Y}] \; , \tag{8.48}$$

with $\boldsymbol{\lambda} \in \nabla \overline{V}_{\mathcal{V}}[\boldsymbol{R}]$ and $\boldsymbol{\mu} \in \nabla \overline{V}_{\mathcal{E}}[\boldsymbol{R}]$.

Proof. The reasoning is similar to the proof of Proposition 8.4.1. □

We detail the computation of the processes $\boldsymbol{\lambda} \in \nabla \overline{V}_{\mathcal{V}}[\boldsymbol{R}]$ and $\boldsymbol{\mu} \in \nabla \overline{V}_{\mathcal{E}}[\boldsymbol{R}]$ in §8.7.2.

8.4.2.2. Resource allocation algorithm

We solve Problem (8.45) with an iterative method. At iteration k we suppose given the allocation $\boldsymbol{R}^{(k)}$ and a sequence $\{\rho_t^{(k)}\}_{t \in \{0,..,T-1\}}$ of gradient steps.

The gradient descent update writes:

$$\boldsymbol{R}_t^{(k+1)} = \mathrm{proj}_{\mathrm{im}(A)}\Big(\boldsymbol{R}_t^{(k)} - \rho_t^{(k)} \, \mathbb{E}\big[\boldsymbol{\lambda}_t^{(k+1)} + \boldsymbol{\mu}_t^{(k+1)} \mid \boldsymbol{Y}_t\big]\Big) \; , \tag{8.49}$$

where $\mathrm{proj}_{\mathrm{im}(A)}$ is the projection onto the subspace $\mathrm{im}(A)$. We note that the computation of the update (8.49) requires also the computation of a conditional expected value, usually estimated by Monte-Carlo.

Remark 8.4.4. *If the graph $G = (\mathcal{V}, \mathcal{E})$ is connected, the projection operator* $\mathrm{proj}_{\mathrm{im}(A)}$ *is the projection onto the hyperplane with equation $\mathbf{1}^\top x = 0$.*

We are also able to handle the boxed constraint $\underline{F} \leq \boldsymbol{F} \leq \overline{F}$ in the projection operator rather tha in the cost function, by using the projection on the intersection of the hyperplane with equation $\mathbf{1}^\top x = 0$ with the boxed set $\underline{F} \leq \cdot \leq \overline{F}$. ◊

Convergence. We refer to (Mataoui, 1990, Theorem IV.4, p.46) for a proof of convergence of Algorithm (8.49).

8.5. Numerical applications

We apply the price and resource decomposition algorithms described in Section 8.4 to a microgrid management problem, where different buildings are connected together. The energy management system (EMS) controls the different energy flows inside the microgrid, so as to ensure that the production meets the demand at all time at least cost.

8.5.1. Problem

We look at a local distribution network connecting different buildings together. We model the distribution network as a directed graph $G = (\mathcal{V}, \mathcal{E})$, with buildings set on nodes and distribution lines set on edges. The buildings exchange energy between each other via the distribution network, and if the local production is unable to fulfill the local demand, energy can be imported from an external regional grid.

The buildings configurations correspond to heterogeneous domestic buildings. All buildings are equipped with electrical hot water tanks, some have solar panels and some others have batteries. As batteries and solar panels are expensive, they are mutualized across the distribution grid. Furthermore, we suppose that all agents are benevolent and share their devices across the network.

We look at a given day in summer. We consider a one day horizon, discretized using a 15mn time step. We view the batteries and the electrical hot water tank as energy stocks. Each house has its own electrical and domestic hot water demands profile. The mathematical modeling of the buildings is similar to the one of Chapter 6. However, we do not consider the thermal inertia of the buildings in the modeling as thermal heating is off during summer.

8.5.1.1. Nodal dynamics

We model the stock dynamics with the models introduced in Chapter 4. Each building has the same modeling as the building described in Chapter 6, but without capturing the thermal inertia.

Let $i \in \{1, .., N\}$ be a node of the graph. We denote by ΔT the discretization step, and by T the horizon. We set $\Delta T = 15$ mn combined with a horizon of one day, which gives $T = 96$.

Nodal state equation. Batteries are modeled with the discrete dynamics

$$\boldsymbol{B}^i_{t+1} = \alpha_b \boldsymbol{B}^i_t + \Delta T\big(\rho_c (\boldsymbol{U}^{b,i}_t)^+ - \frac{1}{\rho_d}(\boldsymbol{U}^{b,i}_t)^-\big)\ ,\ \ \forall t \in \{0, .., T-1\} \tag{8.50a}$$

where $\boldsymbol{B}^i_t$ is the energy stored inside the battery at time t, α_b is the auto-discharge rate, $\boldsymbol{U}^{b,i}_t$ is the power exchanged with the battery, and (ρ_d, ρ_c) are given yields.

The electrical hot water tanks are modeled with the stock dynamics

$$\boldsymbol{H}^i_{t+1} = \alpha_h \boldsymbol{H}^i_t + \Delta T\big(\beta_h \boldsymbol{U}^{t,i}_t - \boldsymbol{D}^{hw,i}_{t+1}\big)\ ,\ \ \forall t \in \{0, .., T-1\}\ , \tag{8.50b}$$

where $\boldsymbol{H}^i_t$ is the energy stored inside the tank at time t, α_h is a discharge rate corresponding to the losses by conduction, $\boldsymbol{U}^{t,i}_t$ is the power used to heat the tank, and $\boldsymbol{D}^{hw,i}_{t+1}$ is the domestic hot water demand between time t and $t+1$.

Depending on the presence of battery inside the building, the nodal state $\boldsymbol{X}^i_t$ has dimension 2 or 1.

- If node i has a battery, its states is $\boldsymbol{X}^i_t = (\boldsymbol{B}^i_t, \boldsymbol{H}^i_t)$;
- otherwise, its state is $\boldsymbol{X}^i_t = \boldsymbol{H}^i_t$.

Nodal uncertainty. At node i, the uncertainty has the expression

$$\boldsymbol{W}_{t+1}^i = (\boldsymbol{D}_{t+1}^{el,i}, \boldsymbol{D}_{t+1}^{hw,i}) \ , \ \ \forall t \in \{0, .., T-1\} \ , \tag{8.51}$$

with $\boldsymbol{D}_{t+1}^{el,i}$ the local electricity demand between time t and $t+1$, and $\boldsymbol{D}_{t+1}^{hw,i}$ the domestic hot water demand. We choose to aggregate the production of the solar panels with the local electricity demands as described in Chapter 6.

We model the distributions of the local energy demands process $\{\boldsymbol{W}_t^i\}_{t\in\{1,..,T\}}$ with discrete probability distributions $\{\mu_t^i\}_{t\in\{1,..,T\}}$ defined on $\mathbb{W}^i$. We note by S the size of its finite support.

Assumption 8.5.1. *For all time $t \in \{0, .., T-1\}$, the nodal uncertainties $\boldsymbol{W}_t^1, \cdots, \boldsymbol{W}_t^N$ are spatially independent.*

Assumption 8.5.1 assumes that the electrical and hot water demands are independent between the different buildings.

Nodal balance equation. At node $i \in \{1, .., N\}$, the *load balance equation* between production and demand corresponds to the mapping Δ_t in Problem (8.3), and writes for all time $t \in \{0, .., T-1\}$

$$\Delta_t(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i, \boldsymbol{F}_t^i) = \boldsymbol{U}_t^{ne,i} - \boldsymbol{D}_t^{el,i} - \boldsymbol{U}_t^{b,i} - \boldsymbol{U}_t^{t,i} - \boldsymbol{F}_t^i \ , \tag{8.52}$$

and we suppose that production must be greater than the demand at all time

$$\Delta_t(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i, \boldsymbol{F}_t^i) \geq 0 \ . \tag{8.53}$$

We suppose that the buildings cannot export electricity: if the global production exceeds the sum of local demands, then the energy surplus is wasted.

Nodal production costs. For all time $t \in \{0, .., T-1\}$, we pay a price p_t^{el} to import electricity from the external regional network. The price p_t^{el} corresponds to *on-peak* and *off-peak* tariffs.

Let

$$L_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) = p_t^{el} \times \max\{0, \boldsymbol{U}_t^{ne,i}\} + \epsilon_1 (\boldsymbol{F}_t^i)^2 \ , \ \ \forall t \in \{0, .., T-1\} \ , \tag{8.54}$$

be the nodal costs. We add a small quadratic term on $\boldsymbol{F}_t^i$ to ensure that the function $J_{\mathcal{V}}(\cdot)$ in (8.7) is strongly convex.

We add a final penalization $K : \mathbb{X}_T \to \mathbb{R}$ to avoid empty stock at midnight for the electrical hot water tank:

$$K(x_T) = \kappa_{\text{hw}} \times \max\{0, h_0 - h_T\} \ , \ \ \forall x_T = (b_T, h_T) \ , \tag{8.55}$$

with $h_0 \in \mathbb{R}$ the initial tank position.

8.5.1.2. Transportation costs

For all edge $\ell \in \{1, .., L\}$, the transportation cost $l_t^\ell : \mathbb{R} \to \mathbb{R}$ in (8.9) is a quadratic function, modeling the losses through the distribution line ℓ:

$$l_t^\ell(q_t^\ell) = \frac{1}{2} a_1 (q_t^\ell)^2 \ . \tag{8.56}$$

We note that we use a simplified modeling for the distribution network, as we consider only energy exchanges without considering the power flow equations. We refer to Kargarian et al. (2016) for an application of distribution and decentralized optimization methods to DC optimal power flow problems. However, the extension to the stochastic case remains to be done.

8.5.1.3. Test-cases

We consider six different problems with growing dimension. Table 8.1 displays the different dimensions considered. We note that the support size of the global noises $\boldsymbol{W}_t$ grows exponentially with the number of nodes, as we suppose that the uncertainties $\{\boldsymbol{W}_t^i\}_{i\in\{1,..,N\}}$ are independent node by node and time by time (see Assumption 8.5.1).

Problem	N (nodes)	L (edges)	$dim(\mathbb{X}_t)$	$dim(\mathbb{W}_t)$	$supp(\boldsymbol{W}_t)$
2-Nodes	2	1	3	4	10^2
3-Nodes	3	3	4	6	10^3
6-Nodes	6	7	8	12	10^6
12-Nodes	12	15	16	24	10^{12}
24-Nodes	24	15	32	48	10^{24}
48-Nodes	48	30	64	96	10^{48}

Table 8.1.: The problems have growing dimensions

Graphs topology. We depict the topology of the different graphs in Figure 8.2.

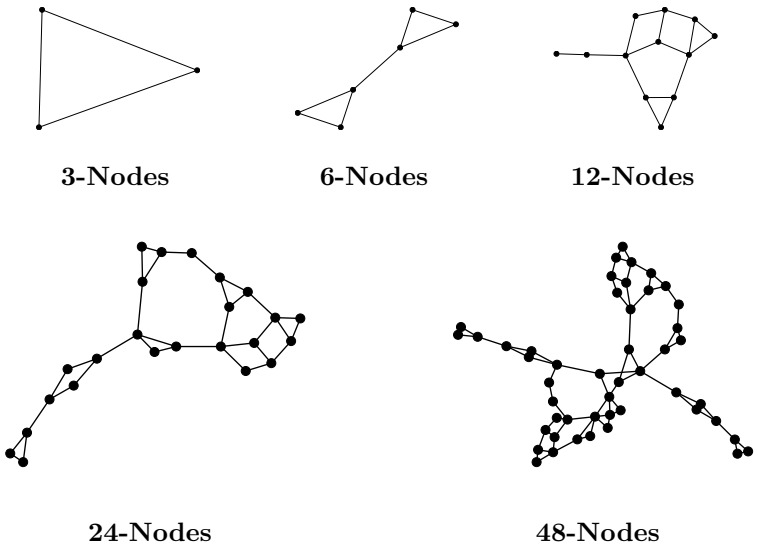


Figure 8.2.: Topology of the different graphs

A description of the 12-Nodes problem. We detail the configuration of the 12-Nodes problem as an example, the configurations of the others problems having a similar structure. 12-Nodes problem gathers twelve buildings, connected via a local distribution network whose topology is given in Figure 8.2. Four buildings are equipped with a 3kWh battery, and four other buildings are equipped with 16m^2 of solar panels. We dispatch the equipments in the different buildings so that each building equipped with solar panels is connected to at least one building with a battery.

8.5.2. Resolution algorithms

We reconsider the two decomposition algorithms introduced in Section 8.4 and apply them to each problem described in Figure 8.2. We will denote by DADP the price decomposition algorithm described in §8.4.1 and by PADP the resource decomposition algorithm described in §8.4.2. We compare DADP and PADP with the well-known Stochastic Dual Dynamic Programming (SDDP) algorithm applied to the global problem.

Each algorithm returns a set of value functions that allow to build a global control policy.

8.5.2.1. Resolution by nodal decomposition

We detailed the nodal decomposition algorithms in Section 8.4. We provide here additional details on the numerical resolution of the nodal subproblems (8.24) and (8.31).

Choosing an appropriate design. We saw in Section 7.3 that if the price process is $\underline{\mathcal{D}}_{y_0,h,\phi}$-Markovian and resource process is $\overline{\mathcal{D}}_{y_0,h,\psi}$-Markovian, we are able to solve the nodal problems (8.24) and (8.31) by Dynamic Programming.

Here, the price process $\lambda = (\lambda_0, \cdots, \lambda_{T-1})$ and the resource process $r = (r_0, \cdots, r_{T-1})$ are chosen deterministic, hence easing the Dynamic Programming resolution of the nodal subproblems (as we do not have to extend the state of each node).

Resolution of nodal subproblems by Dynamic Programming. Once the global problem decomposed into nodal subproblems, we solve each subproblem locally by Dynamic Programming.

Gradient-like algorithms. By Proposition 8.4.2, we know that the gradient-like algorithm converges to the optimal solution of Problem (8.39). However, it is well known that the usual gradient descent algorithm may be slow to converge to the optimum. To overcome this issue, we use a Quasi-Newton algorithm to approximate numerically the Hessian of the global value functions $\underline{V}[\boldsymbol{\lambda}]$ and $\overline{V}[\boldsymbol{R}]$. The gradient (Equation (8.41)) is used to build the Hessian approximation.

Remark 8.5.2. *Once Problem* (8.11) *has been decomposed with a fixed allocation or a fixed price, we solve the* local *Dynamic Programming equations with SDDP. As the nodal state $\boldsymbol{X}_t^i$ exhibits small dimension (1 or 2), SDDP converges in few iterations (less than 30). However, we may have chosen as well SDP to solve the local problems.* ◊

8.5.2.2. Resolution by Stochastic Dual Dynamic Programming

To assess the performance of the nodal decomposition algorithms, we compute aside the *global* Bellman value functions (7.38) by SDDP.

However, the global problem (8.11) exhibits large dimensions — both in the state and in the noise spaces (see Table 8.1)— slowing down the performance of SDDP. SDDP requires the exact computation of an expected value at each time step during the backward pass, which proves to be impossible for the global problem (8.11). To make tractable the global resolution, we resample the finite distribution of the noise process.

Noise resampling. By Assumption 8.5.1, the nodal uncertainties are independent node by node. Thus, the probability distribution μ_t^g of the global uncertainty $\boldsymbol{W}_t$ writes as a product of the local distributions

$$\mu_t^g = \mu_t^1 \otimes \cdots \otimes \mu_t^N \,, \tag{8.57}$$

and its support size is equal to S^N, where S is the support's size of the local uncertainty $\boldsymbol{W}_t^i$. The support size of the global distribution grows exponentially with the number of nodes. Without

resampling, the exact computation of the expected values during SDDP's backward passes become intractable as the size of the problem increases.

To overcome this issue, we resample the probability distribution μ_t^g (supported by S^N points) by quantization to obtain a more tractable size $S' << S^N$. We use the k-means algorithm as a *continuous scenario reduction* method, as described in Rujeerapaiboon et al. (2017). By using the Jensen inequality w.r.t. the noises, we know that the optimal quantization a finite distribution yields a new optimization problem whose optimal value is a lower-bound of the optimal value of the original problem, provided that the local problems are convex w.r.t. the noises $\boldsymbol{W}_1^i, \cdots, \boldsymbol{W}_{T-1}^i$ (Löhndorf and Shapiro, 2017).

In the sequel, we fix the resampling size to $S' = 100$.

Cuts selection. We use a level-one cut selection method (de Matos et al., 2015a) to remove redundant cuts along SDDP's iterations. We know from (Guigues, 2017) that removing cuts with the level-one cut selection algorithm does not impact the convergence of SDDP.

8.5.2.3. Recovering a control policy

Once convergence achieved, all algorithms return a collection of *global* value functions $\{V_t^\infty\}_{t\in\{0,..,T\}}$ approximating the original value functions, where the cost-to-go V_t^∞ is defined by

- $V_t^\infty = \underline{V}_t$ for SDDP,
- $V_t^\infty = \sum_{i=1}^N \underline{V}_t^i$ for price decomposition,
- $V_t^\infty = \sum_{i=1}^N \overline{V}_t^i$ for resource decomposition.

We use these global value functions to build a global control policy for all time $t \in \{0,..,T-1\}$. Such control policy is an *admissible strategy* usable in simulation. The control policy writes, for all global state $x_t \in \mathbb{X}_t$ and global noise $w_{t+1} \in \mathbb{W}_{t+1}$, as solution of a one-step DP problem:

$$\begin{aligned}\pi(x_t, w_{t+1}) \in \arg\min_{u_t} \min_{f_t, q_t} & \sum_{i=1}^N L_t^i(x_t^i, u_t^i, w_{t+1}^i) + \sum_{\ell=1}^L l_t^\ell(q_t^\ell) + V_{t+1}^\infty\big(x_{t+1}^1, \cdots, x_{t+1}^N\big) \\ \text{s.t. } & x_{t+1}^i = g_t^i(x_t^i, u_t^i, w_{t+1}^i)\ , \quad \forall i \in \{1,..,N\}\ , \\ & \Delta_t^i(x_t^i, u_t^i, w_{t+1}^i, f_t^i) \geq 0\ , \quad \forall i \in \{1,..,N\}\ , \\ & Aq_t + f_t = 0\ .\end{aligned} \tag{8.58}$$

As the strategy induced by (8.58) is admissible, the expected value of its cost is an upper bound of the optimal value $V^\sharp$ of the original problem (8.11).

8.5.3. Numerical results

We first compare the time taken by each algorithm to compute the value functions, and evaluate in a second time the quality of the strategy (8.58) obtained.

8.5.3.1. Algorithms parameters

We detail hereafter the different parameters of the algorithms.

Decomposition ingredients. The algorithm uses $N^{mc} = 500$ Monte Carlo samples to estimate the conditional expected gradient $\mathbb{E}[A\boldsymbol{Q}^{(k+1)} + \boldsymbol{F}^{(k+1)} \mid \boldsymbol{Y}]$ by Monte-Carlo in Algorithm (8.43) as

$$\frac{1}{N^{mc}} \sum_{s=1}^{N^{mc}} (A\boldsymbol{Q}^{(k+1)}(w^s) + \boldsymbol{F}^{(k+1)}(w^s)) \,, \tag{8.59}$$

where $(w^1, \cdots, w^{N^{mc}})$ is a fixed set of scenarios used at each iteration of the price and resource decomposition algorithms.

Globally, the gradient-like algorithm is performed with L-BFGS-B 3.0 (Zhu et al., 1997). The algorithm stops when no descent direction is found. The number of iterations depends on the number of nodes considered.

Each nodal subproblem (8.3) is solved by a DP-like method (SDDP algorithm in this application, as it converges in a few iterations).

Global SDDP ingredients. Global SDDP uses a level-one cut selection algorithm (Guigues, 2017) and keeps only the 100 most relevant cuts. By doing so, we divide almost by three the global SDDP's computation time.

We stop SDDP when the gap between its lower bound and its upper bound (estimated statistically every 10 iterations by using a fixed set of 1,000 scenarios) is lower than 1%. That corresponds to the standard SDDP's stopping criterion described in Shapiro (2011), which is reputed to be more consistent than the first stopping criterion introduced in Pereira and Pinto (1991) (we refer to §10.1.2 for a discussion on the relevance of the different stopping criteria of SDDP).

8.5.3.2. Computation of Bellman value functions.

We solve Problem (8.11) by global SDDP, resource allocation (PADP) and price decomposition (DADP). Table 8.2 details the number of iterations and execution time taken before reaching convergence.

Graph	**3-Nodes**	**6-Nodes**	**12-Nodes**	**24-Nodes**	**48-Nodes**
$\lvert\mathbb{X}_t\rvert$	4	8	16	32	64
SDDP time	1'	3'	10'	79'	453'
SDDP Nit	30	100	180	500	1500
DADP time	6'	14'	29'	41'	128'
DADP Nit	27	34	30	19	29
PADP time	3'	7'	22'	49'	91'
PADP Nit	11	12	20	19	20

Table 8.2.: Optimization results for SDDP, DADP and PADP.

We note that for the **24-Nodes** and **48-Nodes** problems, DADP and PADP are almost three times faster than SDDP. However, for small-scale problem like **3-Nodes**, SDDP remains faster than DADP and PADP.

SDDP convergence. Figure 8.3 displays the convergence of SDDP for the 12 nodes problem. The approximate upper-bound is estimated every 10 iterations, with 1,000 scenarios. We observe that the gap between the upper and lower bounds is below 1% after 180 iterations. The lower bound remains stable after 250 iterations.

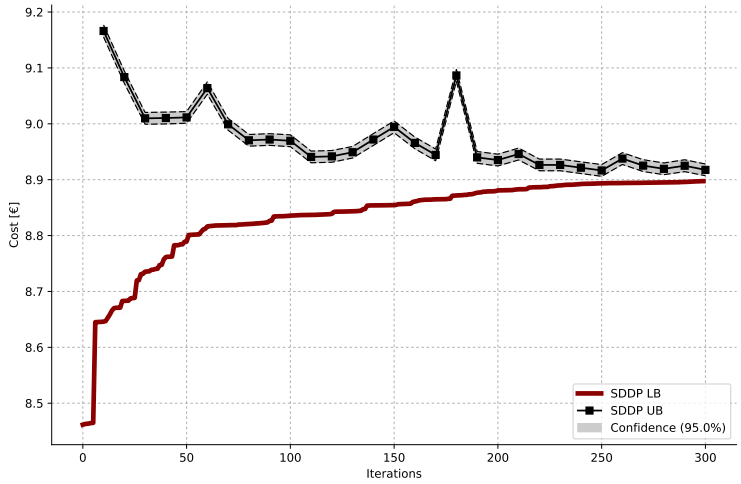
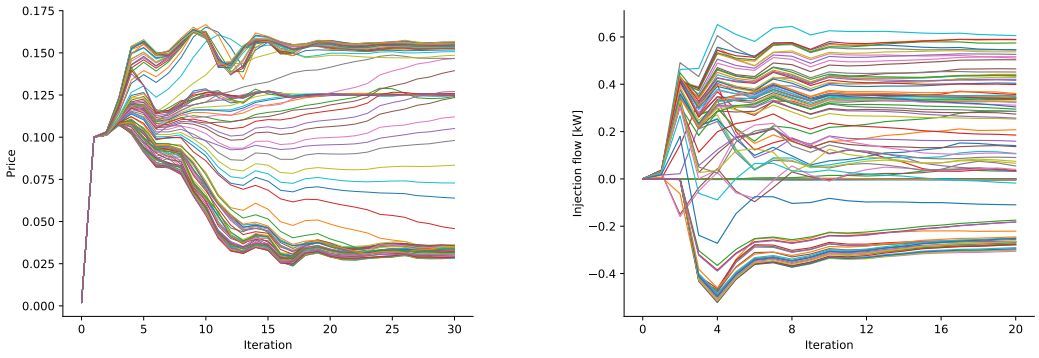


Figure 8.3.: Evolution of SDDP lower and upper bounds for the 12-Nodes problem

DADP and PADP convergence. We exhibit in Figure 8.4 the convergence of DADP's price process and PADP's resource process along iterations, for the **12-Nodes** problem. We depict the convergence only for the first node, the evolution of price process and resource process in other nodes being similar. As we consider a one day horizon combined with a 15mn time step, we get 96 different stages for each nodal subproblem. Thus, the price and resource processes have 96 different values.

On the left-hand side, we plot the evolution of the 96 different values of the price process $\lambda^1 = (\lambda^1_0, \cdots, \lambda^1_{T-1})$ for each iteration of DADP. We observe that most of the prices start to stabilize after 15 iterations, and do not exhibit sensitive variation after 20 iterations.

On the right-hand side, we plot the evolution of the 96 different values of the resource process $r^1 = (r^1_0, \cdots, r^1_{T-1})$, for each iteration of PADP. We observe that the convergence of resources is quicker than for prices, as the evolution of most resources starts to stabilize after only 10 iterations.

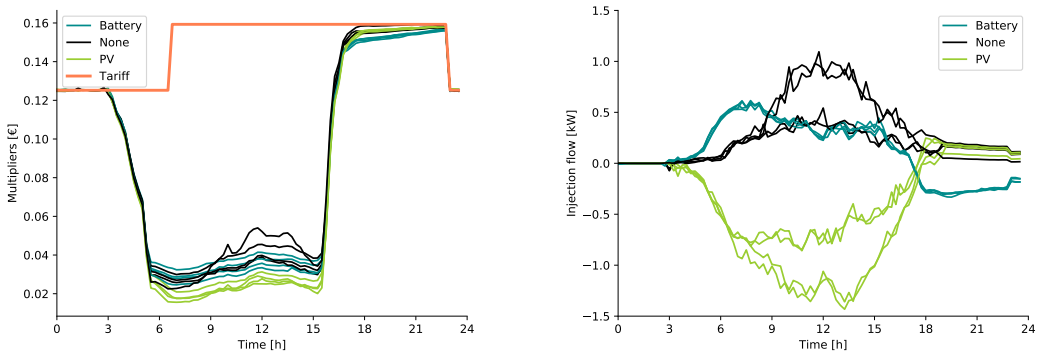


(a) (b)

Figure 8.4.: Convergence of DADP's prices (a) and PADP resources (b) for the **12-Nodes** problem.

Optimal DADP's multipliers and PADP's flows. We display on Figure 8.5 the optimal prices $\boldsymbol{\lambda}$ and resources $\boldsymbol{R}$ obtained by DADP and PADP in the **12-Nodes** problem. As we have twelve nodes, we obtain twelve different curves in each subfigure, corresponding respectively to the price process (on the left) and to the resource process (on the right). The positioning of the different devices is depicted in Figure 8.6. We comment hereafter Figure 8.5 in detail.

- On the left-hand side (a), we observe that the shape of the multipliers reflects on-peak/off-peak electricity tariffs (0.125€ between 23pm and 7am, 0.165€ the remaining time). The values of the multipliers are always lower than the electricity tariffs.
- On the right-hand size (b), we observe that during day, the exporting node are nodes equipped with solar panels (green curves). During night, the nodes equipped with batteries (blue curves) export the energy they stored during day.
- Combining the results from (a) and (b), we observe that the exporting node exhibits a lower $\boldsymbol{\lambda}$ than importing nodes. Here, the price $\boldsymbol{\lambda}$ is an information depicting the marginal price to export a given amount of energy at a given node, and the decision maker has to choose to import electricity from the nodes with the lowest marginal prices.



(a) (b)

Figure 8.5.: DADP optimal prices (b) and PADP optimal resources (b).

8.5.3.3. Simulation results

We now compare the performance of the different algorithms in simulation, using the control strategy induced by (8.58).

Lower bounds. We first give the lower and upper bounds given by DADP, PADP and SDDP in Table 8.3. The lower bound of the SDDP algorithm is the classical lower bound of SDDP, whereas DADP and PADP lower and upper bound are given by Equation (8.35).

We observe that:

- The lower-bound returned by DADP with deterministic coordination price process is better than SDDP's lower bound (which uses a poorer representation of the uncertainties distributions) for problems with more than 12 nodes.
- Whereas SDDP and DADP lower-bounds are close to each other, the upper-bound given by PADP is looser.

Problem	**3-Nodes**	**6-Nodes**	**12-Nodes**	**24-Nodes**	**48-Nodes**
SDDP LB	2.252	4.559	8.897	17.528	33.103
DADP LB	2.137	4.473	8.967	17.870	33.964
PADP UB	2.521	5.285	10.523	21.007	40.166

Table 8.3.: Upper and lower bounds given by SDDP, DADP and PADP.

Simulation values. We give the results obtained by simulation in Table 8.4. SDDP, DADP and PADP values correspond to the values obtained by simulating the strategies induced by (8.58) on $N^{sim} = 5,000$ scenarios. The notation $\pm$ corresponds to the 95% interval $\pm 1.96 \frac{\sigma}{\sqrt{N^{sim}}}$. We use as a reference the cost obtained by the SDDP strategy (a positive gap meaning that the decomposed strategies are better than SDDP's strategy).

Problem	**3-Nodes**	**6-Nodes**	**12-Nodes**	**24-Nodes**	**48-Nodes**
SDDP value	2.26 $\pm$ 0.006	4.71 $\pm$ 0.008	9.36 $\pm$ 0.011	18.59 $\pm$ 0.016	35.50 $\pm$ 0.023
DADP value	2.28 $\pm$ 0.006	4.64 $\pm$ 0.008	9.23 $\pm$ 0.012	18.39 $\pm$ 0.016	34.90 $\pm$ 0.023
Gap	- 0.8 %	+ 1.5 %	+1.4%	+1.1%	+1.7%
PADP value	2.29 $\pm$ 0.006	4.71 $\pm$ 0.008	9.31 $\pm$ 0.011	18.56 $\pm$ 0.016	35.08 $\pm$ 0.022
Gap	-1.3%	0.0%	+0.5%	+0.2%	+1.2%

Table 8.4.: Simulation results for SDDP, DADP and PADP.

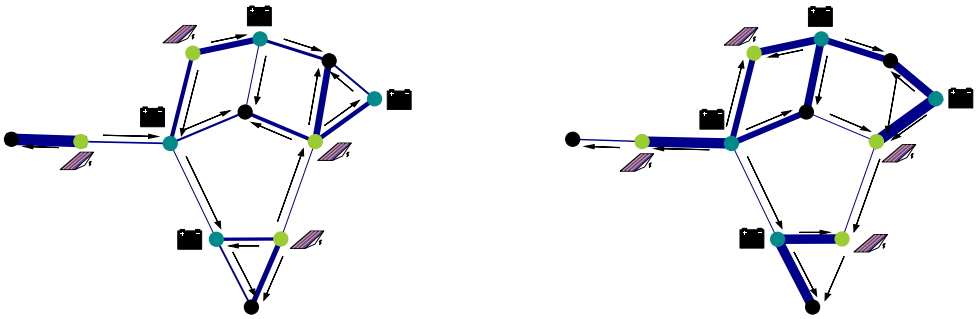
We make the following observations.

- If the number of nodes is greater than 6, DADP and PADP strategies beat SDDP strategy.
- The DADP strategy gives better results than the PADP strategy.
- Comparing with Table 8.3, the upper bounds obtained by the three strategies in simulation are closer to SDDP and DADP lower-bounds than PADP upper-bounds. By assuming that the resource allocation is constant in PADP, we obtain constant importation flows for every possible realization of uncertainties, thus impairing the accuracy of the PADP algorithm. We believe that using a non-constant allocation process in PADP would greatly enhance the quality of the upper-bound.

Simulation trajectories. We now analyze the average flows through the network, as returned by DADP's admissible strategy (8.58).

Figure 8.6 displays the average flows for the **12-Nodes** problem. The colors of the nodes denote the devices in each node, and follow the same color legend as in Figure 8.5 (blue for the node equipped with a battery, green for the node equipped with solar panels and black for the node without any device). The width of the edges is proportional to the intensity of the average flow through the edges: the wider, the larger is the average flow. The arrow gives the sense of the flow, from the exporting node to the importing node.

We give the average flows at two different moments in the day: (a) displays the average flows at midday — where the production of the solar panels is maximal — whereas (b) displays the average flows at 9pm, during twilight. We observe that at midday the nodes equipped with solar panels export their energy surplus to adjacent nodes. On the contrary, at 9pm, the exporting nodes become the nodes equipped with battery, as they are able to export the energy they stored during day. Thus, the directions of the flows at different moments in the day are coherent.



(a) (b)

Figure 8.6.: Average exchanges through the **12-Nodes** problem at 12am (a) and 9pm (b)

Impact of self-consumption. Eventually, we are able to assess the impact of self-consumption on the local network. We compare the electricity importation from the regional distribution grid. We display the average importation in Figure 8.7. for two different cases.

1. First, in blue, we depict the importation observed in the case previously described, where the different buildings exchange energy between each others.
2. Second, in red, we depict the importation observed without local network, that is, without exchange between the different buildings. The second case is a simplification of the former by considering a network without edges.

We observe the following points.

- By allowing exchange, the importations from the regional distribution grid are minimized (the blue curve is always lower than the red curve when importation is greater than 0).
- The energy losses during day, corresponding to negative importations, are also minimized by allowing buildings to exchange energy locally.

Thus, we deduce that self-consumption allows to decrease both the importation from the regional distribution grid and the energy losses.

8.6. Beyond price and resource decompositions

We have presented in §8.4 the price and resource decomposition algorithms and applied them succesfully on a practical problem in §8.5. The aim of this last section is solely to give some hints concerning the extension of price and resource decomposition schemes to other decomposition schemes. We will sketch primal-dual decomposition in §8.6.2 and stochastic proximal methods in §8.6.3. Such decomposition schemes will allow to design new resolution algorithms to solve Problem (8.11).

8.6.1. A new look on price and resource decompositions

We reframe the algorithms introduced in Section 8.4 by using the Fenchel conjugate.

Let (X, Y) be two spaces paired by a bilinear form $\langle \cdot\,, \cdot \rangle$. For all function $f : X \to \overline{\mathbb{R}}$, we define the Fenchel conjugate as

$$f^\star(y) = \sup_{x \in X} \langle y\,, x \rangle - f(x)\,, \quad \forall y \in Y\,. \tag{8.60}$$

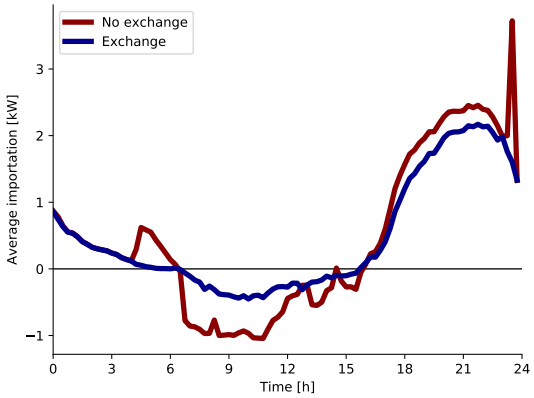


Figure 8.7.: Impact of self-consumption onto energy importation. Negative importation corresponds to losses, as we cannot export electricity onto the distribution grid.

Price decomposition. First, we have seen that the price value function in (8.19) writes, for all $\boldsymbol{\lambda} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N)$,

$$\underline{V}[\boldsymbol{\lambda}] = \min_{\boldsymbol{F},\boldsymbol{Q}} \; J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \left\langle \boldsymbol{\lambda} \, , A\boldsymbol{Q} + \boldsymbol{F} \right\rangle .$$

By using the Fenchel conjugates (8.60) of $J_{\mathcal{V}}$ and $J_{\mathcal{E}}$, we obtain

$$\underline{V}[\boldsymbol{\lambda}] = -J_{\mathcal{V}}^{\star}(-\boldsymbol{\lambda}) - J_{\mathcal{E}}^{\star}(-A^{\top}\boldsymbol{\lambda}) \, . \tag{8.61}$$

We suppose that $J_{\mathcal{V}}^{\star}$ and $J_{\mathcal{E}}^{\star}$ are differentiable w.r.t. $\boldsymbol{\lambda}$. Then, $\underline{V}[\boldsymbol{\lambda}]$ is differentiable, with

$$\nabla \underline{V}[\boldsymbol{\lambda}] = \nabla J_{\mathcal{V}}^{\star}(-\boldsymbol{\lambda}) + A \nabla J_{\mathcal{E}}^{\star}(-A^{\top}\boldsymbol{\lambda}) \, , \tag{8.62}$$

which is similar to Equation (8.41) by setting

$$\boldsymbol{F} = \nabla J_{\mathcal{V}}^{\star}(-\boldsymbol{\lambda}) \, , \quad \boldsymbol{Q} = \nabla J_{\mathcal{E}}^{\star}(-A^{\top}\boldsymbol{\lambda}) \, . \tag{8.63}$$

That allows to rewrite price decomposition in a more compact manner.

Resource decomposition. Resource decomposition is a symmetric version of price decomposition, but in the primal. The resource value function (8.27) rewrites in a compact manner, for all resource process $\boldsymbol{Q} \in \mathbb{L}^0(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^L)$,

$$\overline{V}[\boldsymbol{Q}] = J_{\mathcal{V}}(-A\boldsymbol{Q}) + J_{\mathcal{E}}(\boldsymbol{Q}) \, . \tag{8.64}$$

We suppose, as in Proposition 8.4.3, that the mappings $J_{\mathcal{V}}$ and $J_{\mathcal{E}}$ are differentiable. Then, $\overline{V}[\boldsymbol{Q}]$ is differentiable w.r.t. $\boldsymbol{Q}$, and its gradient is

$$\nabla \overline{V}[\boldsymbol{Q}] = -A^{\top} \nabla J_{\mathcal{V}}(-A\boldsymbol{Q}) + \nabla J_{\mathcal{E}}(\boldsymbol{Q}) \, , \tag{8.65}$$

which is similar to Equation (8.48) by setting

$$\boldsymbol{\lambda} = \nabla J_{\mathcal{V}}(-A\boldsymbol{Q}) \, , \quad \boldsymbol{\mu} = \nabla J_{\mathcal{E}}(\boldsymbol{Q}) \, . \tag{8.66}$$

We highlight the symmetry behind the primal and the dual decomposition algorithms in Figure (8.8).

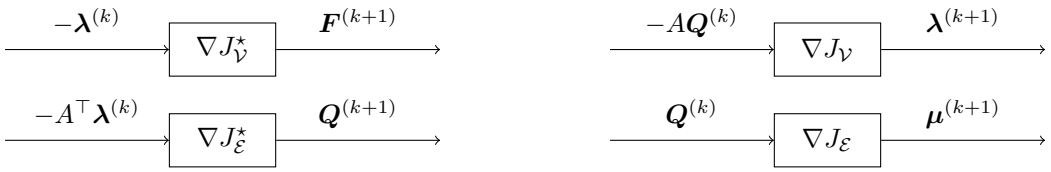


Figure 8.8.: Illustrating the symmetry between price and resource decomposition

8.6.2. Interaction prediction principle

We want now to mix the primal and dual decomposition schemes together, to combine the respective quality of these two algorithms. We recover the interaction-prediction algorithms described in Cohen (2004).

We define the Lagrangian $\mathcal{L} : \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N) \times \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^L) \times \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{R}^N) \to \mathbb{R}$ corresponding to Problem (8.11):

$$\mathcal{L}(\boldsymbol{F}, \boldsymbol{Q},\ \boldsymbol{\lambda}) = J_{\mathcal{V}}(\boldsymbol{F}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \left\langle \boldsymbol{\lambda}\ , A\boldsymbol{Q} + \boldsymbol{F} \right\rangle . \tag{8.67}$$

$\mathcal{L}(\cdot, \cdot, \cdot)$ is convex w.r.t. $(\boldsymbol{F}, \boldsymbol{Q})$, concave w.r.t. $\boldsymbol{\lambda}$. We aim at finding a possible saddle point $(\boldsymbol{F}^{\sharp}, \boldsymbol{Q}^{\sharp}, \boldsymbol{\lambda}^{\sharp})$ of the Lagrangian $\mathcal{L}$.

We propose here a mix between the price and resource decomposition algorithms.

We solve one of the problem (the nodal or the edge problem) in the dual and the other in the primal. That yields two possible primal-dual algorithms:

- The first algorithm dualizes the nodal problem (8.7) and considers as criterion

$$V_1(\boldsymbol{Q}, \boldsymbol{\lambda}) = \min_{\boldsymbol{F}}\ \mathcal{L}(\boldsymbol{F}, \boldsymbol{Q}, \boldsymbol{\lambda}) = -J_{\mathcal{V}}^{\star}(-\boldsymbol{\lambda}) + J_{\mathcal{E}}(\boldsymbol{Q}) + \left\langle \boldsymbol{\lambda}\ , A\boldsymbol{Q} \right\rangle . \tag{8.68}$$

  This scheme considers the couple $(\boldsymbol{Q}, \boldsymbol{\lambda})$ as decomposition variables.

- The second scheme dualizes the edge problem (8.9) and considers as criterion

$$V_2(\boldsymbol{F}, \boldsymbol{\lambda}) = \min_{\boldsymbol{Q}}\ \mathcal{L}(\boldsymbol{F}, \boldsymbol{Q}, \boldsymbol{\lambda}) = J_{\mathcal{V}}(\boldsymbol{F}) - J_{\mathcal{E}}^{\star}(-A^{\top}\boldsymbol{\lambda}) + \left\langle \boldsymbol{\lambda}\ , \boldsymbol{F} \right\rangle . \tag{8.69}$$

  This scheme considers the couple $(\boldsymbol{F}, \boldsymbol{\lambda})$ as decomposition variables.

We detail hereunder the two algorithms that work respectively with V_1 and V_2.

As explained in Cohen and Miara (1990), the primal-dual algorithms can be solved either with a fixed point algorithm or with the Arrow-Hurwicz algorithm (Arrow and Hurwicz, 1958).

8.6.2.1. Dualizing the nodal problem

We consider the function V_1 defined in Equation (8.68). We fix a resource $\boldsymbol{Q}^{(k)}$ and a price $\boldsymbol{\lambda}^{(k)}$.

Fixed point algorithm. The fixed point algorithm takes into account the couple $(\boldsymbol{Q}^{(k)}, \boldsymbol{\lambda}^{(k)})$ and computes

$$\boldsymbol{F}^{(k+1)} = \nabla J_{\mathcal{V}}^{\star}(-\boldsymbol{\lambda}^{(k)})\ ,\ \ \boldsymbol{\mu}^{(k+1)} = \nabla J_{\mathcal{E}}(\boldsymbol{Q}^{(k)})\ . \tag{8.70}$$

But the update of $\boldsymbol{Q}^{(k)}, \boldsymbol{\lambda}^{(k)}$ is not straightforward, as we do not have necessarily $\boldsymbol{F}^{(k+1)} \in im(A)$.

Arrow-Hurwicz algorithm. To overcome the issue encountered in the fixed point algorithm, we use an Arrow-Hurwicz algorithm (Arrow and Hurwicz, 1958). The gradient of V_1 at $(\boldsymbol{Q}^{(k)}, \boldsymbol{\lambda}^{(k)})$ writes:

$$\begin{cases} \nabla_q V_1(\boldsymbol{Q}^{(k)}, \boldsymbol{\lambda}^{(k)}) = \nabla J_{\mathcal{E}}(\boldsymbol{Q}^{(k)}) + A^\top \boldsymbol{\lambda}^{(k)} \\ \nabla_\lambda V_1(\boldsymbol{Q}^{(k)}, \boldsymbol{\lambda}^{(k)}) = \nabla J^\star_{\mathcal{V}}(-\boldsymbol{\lambda}^{(k)}) + A\boldsymbol{Q}^{(k)} \ . \end{cases} \tag{8.71}$$

We are able to find a saddle point with the Arrow-Hurwicz algorithm. By setting $\boldsymbol{\mu}^{(k+1)} = \nabla J_{\mathcal{E}}(\boldsymbol{Q}^{(k)})$ and $\boldsymbol{F}^{(k+1)} = \nabla J^\star_{\mathcal{V}}(-\boldsymbol{\lambda}^{(k)})$ we define

$$\begin{aligned} \boldsymbol{Q}^{(k+1)} &= \boldsymbol{Q}^{(k)} - \rho_1 \nabla_q V_1(\boldsymbol{Q}^{(k)}, \boldsymbol{\lambda}^{(k)}) && \text{(8.72a)} \\ &= \boldsymbol{Q}^{(k)} - \rho_1 \big(\boldsymbol{\mu}^{(k+1)} + A^\top \boldsymbol{\lambda}^{(k)}\big) \ , && \text{(8.72b)} \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{\lambda}^{(k+1)} &= \boldsymbol{\lambda}^{(k)} + \rho_2 \nabla_\lambda V_1(\boldsymbol{Q}^{(k)}, \boldsymbol{\lambda}^{(k)}) && \text{(8.72c)} \\ &= \boldsymbol{\lambda}^{(k)} + \rho_2 \big(\boldsymbol{F}^{(k+1)} + A\boldsymbol{Q}^{(k)}\big) \ . && \text{(8.72d)} \end{aligned}$$

We note that we are able to extend the Arrow-Hurwicz algorithm (8.72) by using implicit scheme and over and under relaxation. We refer to Cohen and Miara (1990) for a broader description.

8.6.2.2. Dualizing the edge problem

We now consider the function V_2 defined in Equation (8.69). We fix a resource $\boldsymbol{F}^{(k)}$ and a price $\boldsymbol{\lambda}^{(k)}$.

Fixed point algorithm. The fixed point algorithm looks at the couple $(\boldsymbol{F}^{(k)}, \boldsymbol{\lambda}^{(k)})$, computes

$$\boldsymbol{\lambda}^{(k+1)} = \nabla J_{\mathcal{V}}(\boldsymbol{F}^{(k)}) \ , \ \ \boldsymbol{Q}^{(k+1)} = \nabla J^\star_{\mathcal{E}}(-A^\top \boldsymbol{\lambda}^{(k)}) \ , \tag{8.73}$$

and sets $\boldsymbol{F}^{(k+1)} = -A\boldsymbol{Q}^{(k+1)}$.

Arrow-Hurwicz algorithm. We now adapt the Arrow-Hurwicz algorithm. The algorithm computes the gradient of V_2 at $(\boldsymbol{F}^{(k)}, \boldsymbol{\lambda}^{(k)})$:

$$\begin{cases} \nabla_f V_2(\boldsymbol{F}^{(k)}, \boldsymbol{\lambda}^{(k)}) = \nabla J_{\mathcal{V}}(\boldsymbol{F}^{(k)}) + \boldsymbol{\lambda}^{(k)} \\ \nabla_\lambda V_2(\boldsymbol{F}^{(k)}, \boldsymbol{\lambda}^{(k)}) = A \nabla J^\star_{\mathcal{E}}(-A^\top \boldsymbol{\lambda}^{(k)}) + \boldsymbol{F}^{(k)} \ . \end{cases} \tag{8.74}$$

By setting $\boldsymbol{\mu}^{(k+1)} = \nabla J_{\mathcal{V}}(\boldsymbol{F}^{(k)})$ and $\boldsymbol{Q}^{(k+1)} = \nabla J^\star_{\mathcal{E}}(-A^\top \boldsymbol{\lambda}^{(k)})$ we obtain the following algorithm:

$$\begin{aligned} \boldsymbol{F}^{(k+1)} &= \boldsymbol{F}^{(k)} - \rho_1 \nabla_q V_1(\boldsymbol{Q}^{(k)}, \boldsymbol{\lambda}^{(k)}) && \text{(8.75a)} \\ &= \boldsymbol{F}^{(k)} - \rho_1 \big(\boldsymbol{\mu}^{(k+1)} + A^\top \boldsymbol{\lambda}^{(k)}\big) \ , && \text{(8.75b)} \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{\lambda}^{(k+1)} &= \boldsymbol{\lambda}^{(k)} + \rho_2 \nabla_\lambda V_1(\boldsymbol{F}^{(k)}, \boldsymbol{\lambda}^{(k)}) && \text{(8.75c)} \\ &= \boldsymbol{\lambda}^{(k)} + \rho_2 \big(\boldsymbol{F}^{(k)} + A\boldsymbol{Q}^{(k+1)}\big) \ . && \text{(8.75d)} \end{aligned}$$

Again, we are able to solve this primal-dual decomposition scheme with the Arrow-Hurwicz algorithm.

8.6.3. Towards proximal methods

We supposed in Proposition 8.4.2 that the costs L_t were strongly convex to ensure the convergence of the algorithms. By relaxing this assumption, proximal decomposition methods are more generic than the classical price and resource decomposition. We give here a short description of proximal decomposition algorithms and frame the well-known ADMM algorithm (see Boyd et al. (2011) for a survey) to decompose Problem (8.11). We refer to Lenoir and Mahey (2017) for a survey on operator splitting and decomposition.

We define the *prox operator* applied to a convex function f as, for all $c > 0$,

$$\text{prox}_f^c(x) = \arg\min_y f(y) + \frac{c}{2}\|x - y\|^2 \ . \tag{8.76}$$

We aim to compute the prox of the function $J_{\mathcal{V}}$. We have the following result.

Proposition 8.6.1. *Let $\boldsymbol{R}$ be a $\mathcal{D}_{y_0,h,\psi}$-Markovian process. Then,*

$$prox_{J_{\mathcal{V}}}^c(\boldsymbol{R}) = \arg\min_{\boldsymbol{F}} \ J_{\mathcal{V}}(\boldsymbol{F}) + \frac{c}{2}\left\|\boldsymbol{F} - \boldsymbol{R}\right\|^2 \ , \tag{8.77}$$

decomposes node by node and is computable by Dynamic Programming.

Proof. We have that:

$$J_{\mathcal{V}}(\boldsymbol{F}) + \frac{c}{2}\left\|\boldsymbol{F} - \boldsymbol{R}\right\|^2 = \sum_{i=1}^{N}\left(J_{\mathcal{V}}^i(\boldsymbol{F}^i) + \frac{c}{2}\left\|\boldsymbol{F}^i - \boldsymbol{R}^i\right\|^2\right) , \tag{8.78}$$

hence the straightforward spatial decomposition.

We now aim to compute the local prox

$$\min_{\boldsymbol{F}^i} J_{\mathcal{V}}^i(\boldsymbol{F}^i) + \frac{c}{2}\left\|\boldsymbol{F}^i - \boldsymbol{R}^i\right\|^2 \ , \tag{8.79}$$

by Dynamic Programming with the $\mathcal{D}$-Markovian process $\boldsymbol{R}^i$. We have that

$$\begin{aligned}
\arg\min_{\boldsymbol{F}^i} J_{\mathcal{V}}^i(\boldsymbol{F}^i) + \frac{c}{2}\left\|\boldsymbol{F}^i - \boldsymbol{R}^i\right\|^2 = \min_{\boldsymbol{X}^i,\boldsymbol{U}^i,\boldsymbol{F}^i} \ & \mathbb{E}\big[\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) + \frac{c}{2}\left\|\boldsymbol{F}_t^i - \boldsymbol{R}_t^i\right\|^2 + K^i(\boldsymbol{X}_T^i)\big] \ , \\
\text{s.t. } & \boldsymbol{X}_{t+1}^i = g_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_{t+1}^i) \ , \ \ \boldsymbol{X}_0^i = x_0^i \ , \\
& \boldsymbol{Y}_{t+1} = h_t(\boldsymbol{Y}_t, \boldsymbol{W}_{t+1}) \ , \ \ \boldsymbol{Y}_0 = y_0 \\
& \boldsymbol{R}_t^i = \psi_t^i(\boldsymbol{Y}_t) \\
& \Delta_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{F}_t^i) \in \Gamma_t \ , \\
& \sigma(\boldsymbol{U}_t^i) \subset \sigma(\boldsymbol{W}_0, \cdots, \boldsymbol{W}_t) \ .
\end{aligned} \tag{8.80}$$

As noises $\{\boldsymbol{W}_t\}_{t\in\{0,T\}}$ are time independent, we can solve Problem (8.80) with the Dynamic Programming equations

$$V_T^i(x_T^i, y_T) = K^i(x_T^i) \ , \tag{8.81a}$$

$$\begin{aligned}
V_t^i(x_t^i, y_t) = \min_{u^i,\ f^i} \ & \mathbb{E}\big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}) + \\
& \frac{c}{2}\left\|f_t^i - \psi_t^i(y_t)\right\|^2 + V_{t+1}^i(f(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i), h_t(y_t, \boldsymbol{W}_{t+1}))\big] \ .
\end{aligned} \tag{8.81b}$$

Hence the results. □

Example. *ADMM is a splitting method related to the Douglas-Rachford algorithm. It solves iteratively two subproblems in the primal and updates the multiplier corresponding to the coupling constraint with a constant step-size. The main advantage of ADMM is that the problems are decoupled during the resolution. We refer to Boyd et al. (2011) for an extended overview of the ADMM method.*

At step k, the algorithm solves iteratively

$$\boldsymbol{F}^{(k+1)} = \arg\min_{\boldsymbol{F}} J_{\mathcal{V}}(\boldsymbol{F}) + \left\langle \boldsymbol{\lambda}^{(k)}, \boldsymbol{F} \right\rangle + \frac{c}{2} \left\| \mathbb{E}\big[A\boldsymbol{Q}^{(k)} \,|\boldsymbol{Y}\big] + \boldsymbol{F} \right\|^2 \tag{8.82a}$$

$$\boldsymbol{Q}^{(k+1)} = \arg\min_{\boldsymbol{Q}} J_{\mathcal{E}}(\boldsymbol{Q}) + \left\langle \boldsymbol{\lambda}^{(k)}, A\boldsymbol{Q} \right\rangle + \frac{c}{2} \left\| A\boldsymbol{Q} + \mathbb{E}\big[\boldsymbol{F}^{(k+1)} \mid \boldsymbol{Y}\big] \right\|^2 \tag{8.82b}$$

$$\boldsymbol{\lambda}_t^{(k+1)} = \boldsymbol{\lambda}_t^{(k)} + c\,(A\boldsymbol{Q}_t^{(k+1)} + \boldsymbol{F}_t^{(k+1)})\,. \tag{8.82c}$$

The update defined in Equations (8.82) *follows a Gauss-Seidel scheme.* △

8.6.3.1. Shortcoming of proximal methods

We reconsider the relaxed problem (8.39) with the relaxed coupling constraint:

$$\mathbb{E}\Big[A\boldsymbol{Q}_t + \boldsymbol{F}_t \mid \boldsymbol{Y}_t\Big] = 0\,. \tag{8.83}$$

A well-known result states that if we consider any multiplier $\boldsymbol{\mu}_t$ we have:

$$\begin{aligned} \mathbb{E}\Big[\boldsymbol{\mu}_t \cdot \mathbb{E}\big[A\boldsymbol{Q}_t + \boldsymbol{F}_t \mid \boldsymbol{Y}_t\big]\Big] &= \mathbb{E}\Big[\mathbb{E}\big[\boldsymbol{\mu}_t \mid \boldsymbol{Y}_t\big] \cdot \mathbb{E}\big[A\boldsymbol{Q}_t + \boldsymbol{F}_t \mid \boldsymbol{Y}_t\big]\Big] \\ &= \mathbb{E}\Big[\mathbb{E}\big[\boldsymbol{\mu}_t \mid \boldsymbol{Y}_t\big] \cdot \big(A\boldsymbol{Q}_t + \boldsymbol{F}_t\big)\Big]\,, \end{aligned} \tag{8.84}$$

as the conditional expected value is self-adjoint. However, we fail to interpret in a similar manner the expected value of the quadratic term $\mathbb{E}\big[\,\big\|\mathbb{E}\big[A\boldsymbol{Q}_t + \boldsymbol{F}_t \mid \boldsymbol{Y}_t\big]\big\|^2\,\big]$ in Equation (8.82). We refer to (Girardeau, 2010, Chapter 3, p.59) for a description of how to linearize the augmented Lagrangian in a multistage stochastic optimization problem.

8.7. Discussion

We have presented in this chapter a way to decompose spatially optimization problems where coupling constraints correspond to interaction exchanges on a graph. We have presented two decomposition algorithms in Section 8.4, the first relying on price decomposition and the second on resource. The decomposition algorithms work in a decentralized manner and are fully parallelizable. We apply these algorithms on a specific case study in Section 8.5, considering a district microgrid with different prosumers exchanging energy altogether. Numerical results show the relevance of the approach, as the decomposition algorithms beat the reference Stochastic Dual Dynamic Programming (SDDP) for large-scale problems with more than 12 nodes. The decomposition algorithms were applied to problems with up to 48 nodes, and we observed that their performance scaled well as the number of nodes grew.

This study can be extended in several directions. First, we considered only deterministic price and resource processes. We believe that using more complex processes would allow to improve the performance of the algorithm. However, it remains to find a proper way to find an information scheme being a good trade-off between accuracy and numerical performance. Second, we considered in Section 8.5 an easy-to-solve edges subproblem. The decomposition approach should be

extended to more complex network problems, such as those encountered in optimal power flow. Eventually, an extension of stochastic decomposition to more complicated decomposition schemes remains under study, as Section 8.6 just provide sketches for new decomposition methods relying on proximal splitting methods.

Appendix

8.7.1. Background on graph theory

Introducing the incidence matrix.

Definition 8.7.1 (Directed graph)**.** *A directed graph $G = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$, such as:*

- *arcs are ordered pairs (i, j) of nodes,*
- *there is at most one arc from node i to node j,*
- *there are no loops (arc (i, i)).*

Definition 8.7.2. *The* node-arc incidence matrix *of a directed graph $(\mathcal{V}, \mathcal{E})$ is the matrix $A \in \mathbb{R}^{N\times L}$, where N is the number of nodes and L the number of edges, such that:*

$$A_{ij} = \begin{cases} 1 & \text{arc } j \text{ leaves node } i, \\ -1 & \text{arc } j \text{ enters node } i, \\ 0 & \text{otherwise.} \end{cases} \tag{8.85}$$

The adjacency matrix $M \in \mathbb{R}^{N\times N}$ *of a graph G is the matrix such that*

$$M_{ij} = \begin{cases} 1 & \text{node } i \text{ is connected to node } j \\ 0 & \text{otherwise.} \end{cases} \tag{8.86}$$

The degree matrix $D \in \mathbb{R}^{N\times N}$ *is the diagonal matrix such that*

$$D = diag(\deg(v_1), \cdots, \deg(v_N)) \tag{8.87}$$

where $\deg(v_i)$ is the number of edges connected to node i.

The Laplacian $L \in \mathbb{R}^{N\times N}$ *of the graph G is defined as*

$$L = D - M \ . \tag{8.88}$$

Proposition 8.7.3. *Let A be the node-arc incidence matrix of the directed graph $G = (\mathcal{V}, \mathcal{E})$. The Laplacian L is equal to*

$$L = AA^\top \ . \tag{8.89}$$

Proposition 8.7.4. *If the graph $G = (\mathcal{V}, \mathcal{E})$ is connected, the rank of the incidence matrix A is $rank(A) = N - 1$.*

We illustrate on an electrical network example how these different matrix relate.

Example (Incidence matrix and Kirchoff's laws.)**.** *Let $G = (\mathcal{V}, \mathcal{E})$ be an electrical network, and A its node-arc incidence matrix.*

Let i_L (resp. v_L) be the current (resp. the voltage) through the edges and i_N (resp. v_N) be the current (resp. the voltage) through the nodes. Then, the Kirchhoff's Current Law (KCL) writes

$$i_N = Ai_L \ , \tag{8.90}$$

and the Kirchhoff's Voltage Law (KVL) writes

$$v_L = A^\top v_N \ . \tag{8.91}$$

Let $Y \in \mathbb{R}^{N\times N}$ be the node admittance matrix *of the network. The Ohm's law relates the node voltages v_N with the nodes currents i_N, such as*

$$i_N = Y v_N \ . \tag{8.92}$$

The bus admittance matrix is related to the incidence matrix as follows

$$Y = A Y_d A^\top \ , \tag{8.93}$$

where $Y_d \in \mathbb{R}^{L\times L}$ is the primitive admittance matrix, that is, a diagonal matrix with the admittance of each line in its diagonal

$$Y_d = diag\big(y_1, \cdots, y_L\big) \ . \tag{8.94}$$

Y is a weighted Laplacian matrix for graph G. △

8.7.2. Extracting sensitivity in resource allocation.

We now detail the numerical computation of the sensitivities $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ in Equation (8.49).

The computation of the sensitivity $\boldsymbol{\mu}$ associated to the edge resource problem $\overline{V}_{\mathcal{E}}$ is straightforward, as $J_{\mathcal{E}}$ is quadratic. However, the computation of the sensitivity process $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_0, \cdots, \boldsymbol{\lambda}_{T-1})$ associated to the nodal resource problem $\overline{V}_{\mathcal{V}}$ is more cumbersome. We detail in the sequel the associated procedure.

Let $i \in \{1, .., N\}$ be a node in the graph, and let us detail the computation of the sensitivity $\boldsymbol{\lambda}_t^i$ for all time t.

At time $t \in \{0, .., T-1\}$, the Dynamic Programming equations (7.53) satisfied by the nodal resource value functions $\{\overline{V}_t^i\}_{t\in\{0,..,T\}}$ writes

$$\begin{aligned}
\overline{V}_t^i(x_t^i, y_t) = \min_{u_t^i, f_t^i, \boldsymbol{X}_{t+1}^i} \ & \mathbb{E}\Big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) + \overline{V}_{t+1}^i(\boldsymbol{X}_{t+1}^i, \boldsymbol{Y}_{t+1})\Big] \\
& \text{s.t. } \boldsymbol{X}_{t+1}^i = g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) \rightsquigarrow \boldsymbol{\delta}_{t+1}^i \\
& \quad \boldsymbol{Y}_{t+1} = h_t(y_t, \boldsymbol{W}_{t+1}) \\
& \quad \Delta_t^i(x_t^i, u_t^i, w_{t+1}^i, f_t^i) \leq 0 \rightsquigarrow \nu_t^i \\
& \quad f_t^i - r_t^i = 0 \quad \rightsquigarrow \lambda_t^i \ .
\end{aligned} \tag{8.95}$$

The sensitivity λ_t^i at point x_t^i corresponds to the multiplier of the constraint $f_t^i - r_t^i = 0$ in Problem (8.95). By dualizing the constraints in Problem (8.95), we obtain the dual problem:

$$\begin{aligned}
\overline{V}_t^i(x_t^i, y_t) = \max_{\lambda_t^i, \boldsymbol{\delta}_{t+1}^i, \nu_t^i} \min_{u_t^i, f_t^i, \boldsymbol{X}_{t+1}^i} \ & \mathbb{E}\Big[L_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}) + \overline{V}_{t+1}^i(\boldsymbol{X}_{t+1}^i, \boldsymbol{Y}_{t+1}) + \\
& \big\langle \nu_t^i \, , \Delta_t^i(x_t^i, u_t^i, w_{t+1}^i, f_t^i)\big\rangle + \\
& \big\langle \lambda_t^i \, , r_t^i - f_t^i\big\rangle + \big\langle \boldsymbol{\delta}_{t+1}^i \, , g_t^i(x_t^i, u_t^i, \boldsymbol{W}_{t+1}^i) - \boldsymbol{X}_{t+1}^i\big\rangle\Big] \ .
\end{aligned} \tag{8.96}$$

We write the first-order necessary conditions for Problem (8.96):

- The optimal control $u_t^\sharp$ satisfies

$$\mathbb{E}\Big[\nabla_u L_t^i(x_t^i, u_t^\sharp, \boldsymbol{W}_{t+1}^i)^\top + \nabla_u g_t^i(x_t^i, u_t^\sharp, \boldsymbol{W}_{t+1}^i)^\top \boldsymbol{\delta}_{t+1}^i\Big] = -\nabla_u \Delta_t^i(x_t^i, u_t^\sharp, w_{t+1}^i, f_t^\sharp)^\top \nu_t^i \ , \tag{8.97}$$

- The optimal allocation $f_t^\sharp$ satisfies

$$\nabla_f \Delta_t^i(x_t^i, u_t^\sharp, w_{t+1}^i, f_t^\sharp)^\top \nu_t^i = \lambda_t^i \ . \ , \tag{8.98}$$

- The optimal future state $\boldsymbol{X}_{t+1}^i$ satisfies

$$\boldsymbol{\delta}_{t+1}^i \in \partial \overline{V}_{t+1}^i(\boldsymbol{X}_{t+1}^i) \ . \tag{8.99}$$

Proposition 8.7.5. *The sensitivity w.r.t. an injection flow $\boldsymbol{R} = (\boldsymbol{R}_0, \cdots, \boldsymbol{R}_{T-1})$ satisfies, for all $t \in \{0, .., T-1\}$, for all $x_t^i \in \mathbb{X}_t^i$*

$$\lambda_t^i = \nabla_f \Delta_t^i(x_t^i, u_t^\sharp, w_{t+1}^i, f_t^\sharp)^\top \nu_t^i \ , \tag{8.100}$$

with $u_t^\sharp$, $f_t^\sharp$ solutions of Problem (8.95) and ν_t^i multiplier associated to the coupling constraint between controls and injection flows.

Remark 8.7.6. *The multiplier $\boldsymbol{\delta}_{t+1}$ is the co-state associated to the state $\boldsymbol{X}_{t+1}$.* ♢

Chapter 9.

Stochastic decomposition applied to large-scale hydro valleys management

This chapter is a transcription of the article Carpentier et al. (2018b). The author thanks his co-authors Pierre Carpentier, Jean-Philippe Chancelier and Vincent Leclère.

Contents

We are interested in optimally controlling a discrete time dynamical system that can be influenced by exogenous uncertainties. This is generally called a Stochastic Optimal Control (SOC) problem and the Dynamic Programming (DP) principle is one of the standard ways of solving it. Unfortunately, DP faces the so-called curse of dimensionality: the complexity of solving DP equations grows exponentially with the dimension of the variable that is sufficient to take optimal decisions (the so-called state variable). For a large class of SOC problems, which includes important practical applications in energy management, we propose an original way of obtaining near

optimal controls. The algorithm we introduce is based on Lagrangian relaxation, of which the application to decomposition is well-known in the deterministic framework. However, its application to such closed-loop problems is not straightforward and an additional statistical approximation concerning the dual process is needed. The resulting methodology is called Dual Approximate Dynamic Programming (DADP). We briefly present DADP, give interpretations and enlighten the error induced by the approximation. The chapter is mainly devoted to applying DADP to the management of large hydro valleys. The modeling of such systems is presented, as well as the practical implementation of the methodology. Numerical results are provided on several valleys, and we compare our approach with the state of the art SDDP method.

9.1. Introduction

9.1.1. Large-scale systems and energy applications

Consider a controlled dynamical system over a discrete and finite time horizon. This system may be influenced by exogenous noises that affect its behavior. Assume that, at every instant t, the decision maker designs a control based on all the observations of noises available up to time t. We are thus looking for strategies (or policies), that is, feedback functions that map every instant and every possible history of the system to a decision to be made.

We can find typical applications in the field of energy management. Consider a power producer that owns a certain number of power units. Each unit has its own local characteristics such as physical constraints that restrain the set of feasible decisions, and induces a production cost or a revenue. The power producer controls the power units so that an overall goal is met. A classical example is the so-called unit commitment problem (see Takriti et al. (1996)) where the producer has to satisfy a global power demand at every instant. The power demand, as well as other parameters such as unit breakdowns, are random. The producer is looking for strategies that minimize the overall expected production cost, over a given time horizon. Another application, which is considered in this chapter, is the management of a large-scale hydro valley: here the power producer manages a cascade of dams, and maximizes the revenue obtained by selling the energy produced by turbinating the water inside the dams. Both natural inflows in water reservoirs and energy prices are random. In all these problems, the number of power units and the number of time steps are usually large (see de Matos et al. (2015b)).

9.1.2. Standard resolution methods

One classical approach when dealing with stochastic dynamic optimization problems is to discretize the random inputs of the problem using a scenario tree. Such an approach has been widely studied within the stochastic programming community (see Heitsch and Römisch (2009), Shapiro et al. (2009)), and used to model and solve energy problems, e.g. by Pflug and Pichler (2014). One of the advantages of such a technique is that, as soon as the scenario tree is drawn, the derived problem can be treated by classical mathematical programming techniques. Thus, a number of decomposition methodologies have been proposed (see for instance Rockafellar and Wets (1991b), Carpentier et al. (1996), Ruszczyński (1997), (Ruszczyński and Shapiro, 2003, Chap. 3) and applied to energy planning problems (see Bacaud et al. (2001)). Ways to combine the discretization of the expected value together with the discretization of information in a general setting have been presented in Heitsch et al. (2006), Pflug and Pichler (2014) and Carpentier et al. (2015)). However, in a multi-stage setting, this methodology suffers from the drawback that arises with scenario trees: as it was pointed out by Shapiro (2006), the number of scenarios needed to achieve a given accuracy grows exponentially with the number of time steps of the problem.

The other natural approach to solve SOC problems is to rely on the Dynamic Programming (DP) principle (see Bellman (1957), Puterman (1994)). The core of the DP approach is the definition of

a state variable that is, roughly speaking, the variable that, in conjunction with the time variable, is sufficient to take an optimal decision at every instant. It does not have the drawback of the scenario trees concerning the number of time steps since strategies are, in this context, depending on a state variable whose space dimension does not grow with time (usually linked to the number of power units in the case of power management). However, DP suffers from another drawback which is the so-called *curse of dimensionality*: the complexity of solving the DP equation grows exponentially with the state space dimension. Hence, solving the DP equation by brute force is generally intractable when the state space dimension goes beyond several units. In Vezolle et al. (2009), the authors were able to solve DP on a 10 state variables energy management problem, using parallel computation coupled with adequate data distribution, but the DP limits are around 5 state variables in a straightforward use of the method.

Another popular idea is to represent the value functions (solutions of the DP equation) as a linear combination of a priori chosen basis functions (see Bertsekas and Tsitsiklis (1996)). This approach, called Approximate Dynamic Programming (ADP) has become very popular and the reader is referred to Powell (2007) and Bertsekas (2012) for a description of ADP. This approximation drastically reduces the complexity of solving the DP equation. However, in order to be practically efficient, such an approach requires some a priori information about the problem, in order to define a well suited functional subspace. Indeed, there is no systematic means to choose the basis functions and several choices have been proposed in the literature (see Tsitsiklis and Van Roy (1996)).

Last but not least is the popular DP-based method called Stochastic Dual Dynamic Programming (SDDP). Starting with the seminal work of Van Slyke and Wets (1969), the SDDP method has been designed in Pereira and Pinto (1991). It has been widely used in the energy management context and lately regained interest in the Stochastic Programming community (see Shapiro (2011) and references therein). The idea is to extend Kelley's cutting plane method to the case of multi-stage stochastic problems. Alternatively it can be seen as a multistage Benders (or L-shaped) decomposition method with sampling. It consists of a succession of forward (trajectory computation) and backward (Bellman function refining) passes that ultimately aims at approaching the Bellman function as the supremum of affine hyperplanes (cuts) generated during the backward passes.

9.1.3. Decomposition approach

When dealing with large-scale optimization problems, the decomposition-coordination approach aims at finding a solution to the original problem by iteratively solving subproblems of smaller dimension. In the deterministic case, several types of decomposition have been proposed (e.g. by prices, by quantities or by interaction prediction) and unified in Cohen (1980) using a general framework called Auxiliary Problem Principle. In the open-loop stochastic case, i.e. when controls do not rely on any observation, it is proposed in Cohen and Culioli (1990) to take advantage of both decomposition techniques and stochastic gradient algorithms. The natural extension of these techniques to the closed-loop stochastic case (see Barty et al. (2009)), i.e. when the control is a function of the available observations, fails to provide decomposed state dependent strategies. Indeed, the optimal strategy of a subproblem depends on the state of the whole system, and not only on the local state.

We recently proposed a way to use price decomposition within the closed-loop stochastic case. The coupling constraints, namely the constraints preventing the problem from being naturally decomposed, are dualized using a Lagrange multiplier (price). At each iteration, the price decomposition algorithm solves each subproblem using the current price, and then uses the solutions to update it. In the stochastic context, the price is a random process whose dynamics is not available, so the subproblems do not in general fall into the Markovian setting. However, in a specific instance of this problem (see Strugarek (2006)), the author exhibited a dynamics for the optimal

multiplier and showed that these dynamics were independent from the decision variables. Hence it was possible to come down to the Markovian framework and use DP to solve the subproblems. Following this idea, it is proposed in Barty et al. (2010b) to choose a parameterized dynamics for these multipliers in such a way that solving subproblems using DP becomes possible. While the approach, called Dual Approximate Dynamic Programming (DADP), showed promising results on numerical examples, it suffered from the fact that the induced restrained dual space is non-convex, leading to some numerical instabilities. Moreover, it was not possible to give convergence results for the algorithm. The method has then been improved both from the theoretical and from the practical point of view. The core idea is to replace the current Lagrange multiplier by its conditional expectation with respect to some *information process*, at every iteration. This information process has to be a priori chosen and adapted to the natural filtration. Moreover, if the information process is driven by a dynamics, the state in each subproblem then consists of the original state augmented by the information process, making the resolution of the subproblem tractable by DP. Interestingly, approximating the multipliers by their conditional expectations is equivalent to solving a relaxed primal problem where the almost-sure coupling constraint has been replaced by its conditional expectation with respect to the information variable, yielding a lower bound of the true optimal cost. Further, the solutions obtained by the DADP algorithm do not necessarily satisfy the initial almost-sure coupling constraint, so we must rely on a heuristic procedure to produce a feasible solution to the original problem.

9.1.4. Contents of the chapter

The main contribution of the chapter is to give a practical algorithm aiming at solving large scale stochastic optimal control problems and providing closed-loop strategies. The numerous approximations used in the algorithm, and especially the one allowing for feasible strategies, make it difficult to theoretically assess the quality of the solution finally adopted. Nevertheless, numerical implementation shows that the method is promising to solve large scale stochastic optimization problems such as those encountered in the field of energy management.

The chapter is organized as follows. In §9.2, we present the hydro valley management problem, the corresponding general SOC formulation and the DP principle. We then focus on spatial decomposition of such a problem and the difficulties of using DP at the subproblem level. In §9.3, we present the DADP method and give different interpretations. We then propose a way to recover an admissible solution from the DADP results and we briefly discuss the theoretical and practical questions associated to the convergence and implementation of the method. Finally, in §9.4, we apply the DADP method to the management of hydro valleys. Different examples, corresponding to either academic or realistic valleys, are described. A comparison of the method with SDDP is outlined.

9.1.5. Notations

We will use the following notations.

- $\mathrm{J}i, j\mathrm{K}$ is the set of integers between i and j;
- $(\Omega, \mathcal{A}, \mathbb{P})$ is a probability space;
- bold letters are used for random variables, normal font for their realizations;
- $\boldsymbol{X} \preceq \mathcal{F}_t$ (resp. $\boldsymbol{X} \preceq \boldsymbol{Y}$) means that the random variable $\boldsymbol{X}$ is measurable with respect to the σ-algebra $\mathcal{F}_t$ (resp. with respect to the σ-algebra generated by $\boldsymbol{Y}$, denoted by $\sigma(\boldsymbol{Y})$);
- x generally stands for the state, u for the control, w for an exogeneous noise, and z for the controlled inputs from upstream releases and spills;

- f_t stands for a dynamics, that is, a transition function modeling the system evolution along time, L_t stands for a cost function at time t, K stands for a final cost function;
- V_t represents a Bellman's value function at time t;
- the notation $\boldsymbol{X}^i$ (resp. $\boldsymbol{U}^i$ and $\boldsymbol{Z}^i$) stands for the discrete time state process $(\boldsymbol{X}_0^i, \dots, \boldsymbol{X}_T^i)$ (resp. the two control processes $(\boldsymbol{U}_0^i, \dots, \boldsymbol{U}_{T-1}^i)$ and $(\boldsymbol{Z}_0^i, \dots, \boldsymbol{Z}_{T-1}^i)$).

9.2. Mathematical formulation

In this section, we present the modeling of a hydro valley and the associated optimization framework.

9.2.1. A generic formulation

We are interested in solving a multistage stochastic optimal control problem over a discrete-time horizon $\mathrm{J}0, T\mathrm{K}$. In this problem we consider multiple stochastic systems indexed by $i \in \mathrm{J}1, N\mathrm{K}$, that follow independent dynamics but that must satisfy a coupling constraint.

More precisely, we address the following problem

$$\min_{(\boldsymbol{X}^i, \boldsymbol{U}^i)_{i \in \mathrm{J}1, N\mathrm{K}}} \mathbb{E}\Big(\sum_{i=1}^{N} \Big(\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t) + K^i(\boldsymbol{X}_T^i) \Big) \Big) \,, \tag{9.1a}$$

$$\text{s.t.} \quad \boldsymbol{X}_{t+1}^i = f_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t) \,, \quad \boldsymbol{X}_0^i \quad \text{given} \,, \tag{9.1b}$$

$$\boldsymbol{U}_t^i \preceq \sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) \,, \tag{9.1c}$$

$$\sum_{i=1}^{N} \Theta_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t) = 0 \,. \tag{9.1d}$$

Constraints (9.1b) represent the dynamics and constraints (9.1c) are the non-anticipativity constraints, that is, the fact that each control $\boldsymbol{U}_t^i$ at time t, considered as a random variable, has to be *measurable* with respect to the sigma-field $\sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t)$ generated by noises up to time t. The last constraints (9.1d) express the interactions between the production units i. They represent an *additive* coupling with respect to the different production units, which is termed the "spatial coupling of the problem". Such a general modeling covers other cases than the cascade problem, such that the unit commitment problem, or the problem of exchanging energy on a smart grid.

9.2.2. Dams management problem

We consider a hydro valley constituted of N cascaded dams as represented in Figure 9.1. The water turbinated at a dam produces energy which is sold on electricity markets, and then enters the nearest downstream dam. The overall goal of the decision maker is to maximize the profit obtained by selling the produced energy on a market. We consider that the hydro valley manager acts as a price follower, in the sense that the energy prices are independent of the energy produced by the hydro valley. Note that the valley geometry may be more complicated than a pure cascade: see for example the valleys represented at Figure 9.4.

The representative variables of dam i at time t are u_t^i for the turbinated water, x_t^i for the current water volume, a_t^i for the natural water inflow entering dam i, p_t^i for the market value of the water at dam i. The randomness is given by $w_t^i = (a_t^i, p_t^i)$. The modeling of a dam takes into account a possible overflow: the spilled water does not produce electricity, but enters the next downstream dam.

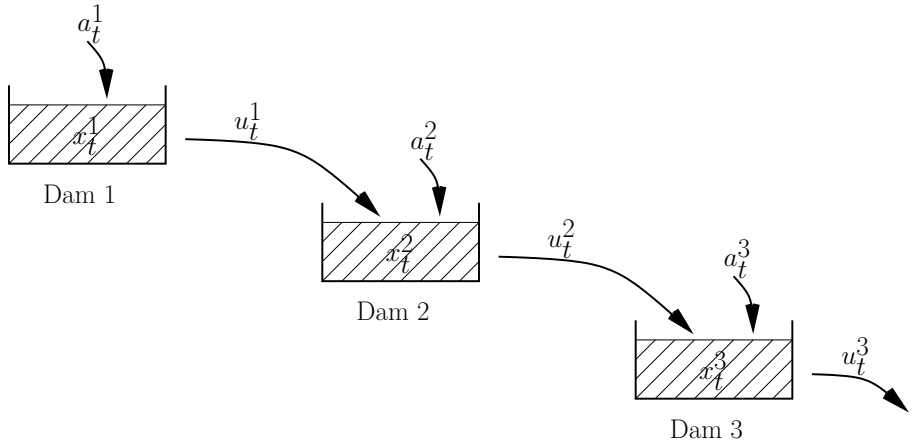


Figure 9.1.: Operating scheme of a hydro valley with 3 dams.

We now cast the problem in the generic framework presented at §9.2.1, with a slight abuse of notation ($\boldsymbol{U}_t^i$ stands for $(\boldsymbol{U}_t^i, \boldsymbol{Z}_t^i)$ here).

- The dam dynamics (corresponding to Equation (9.1b)) reads

$$x_{t+1}^i = x_t^i - u_t^i + a_t^i + z_t^i - s_t^i = f_t^i(x_t^i, (u_t^i, z_t^i), w_t^i) \,, \tag{9.2a}$$

  where s_t^i is the volume of water spilled by overflowing the dam:

$$s_t^i = \max\left\{0, x_t^i - u_t^i + a_t^i + z_t^i - \overline{x}^i\right\} \,. \tag{9.2b}$$

  The constant value $\overline{x}^i$ stands for the maximal capacity of dam i. The outflow of dam i, that is, the sum of the turbinated water and of the spilled water, is denoted by z_t^{i+1}:

$$z_t^{i+1} = u_t^i + s_t^i = g_t^i(x_t^i, (u_t^i, z_t^i), w_t^i) \,. \tag{9.2c}$$

  This last equation corresponds to Equation (9.1d) in the general framework. Note that the dynamic equations (9.2a) are nonlinear because of the max operator in the definition (9.2b) of the spilled water volume. We assume the *Hazard-Decision* information structure: the control u_t^i applied at time t is chosen once the noise w_t^i at time t has been observed. It is thus possible to ensure that the dam always remains above its minimal admissible volume $\underline{x}^i$ by limiting the control range: $\underline{u}^i \leq u_t^i \leq \min\left\{\overline{u}^i, x_t^i + a_t^i + z_t^i - \underline{x}^i\right\}$.

  Remark 9.2.1. *As will be seen in §9.4, the typical time step length we use is the month (with a time horizon of one year). It is thus reasonable to assume the Hazard-Decision framework, the control applied for a given month being in fact implemented each day taking into account the observed information on a daily basis.* ◊

- The objective function of dam i is the sum of different terms.
  - The cost at each time $t \in \mathrm{J}0, T-1\mathrm{K}$ is $L_t^i(x_t^i, (u_t^i, z_t^i), w_t^i) = -p_t^i u_t^i + \varepsilon(u_t^i)^2$. The first linear term corresponds to the opposite of the profit when selling the energy produced by the turbinated water on the energy market. The second term $\varepsilon(u_t^i)^2$ models the operating cost of the turbine as a quadratic term, and is usually small. This last term ensures the strong convexity of the cost function.

– The final cost at time T is $K^i\big(x_T^i\big) = \alpha^i \min\{0, \widehat{x}^i - x_T^i\}^2$. It corresponds to a quadratic penalization around a target value $\widehat{x}^i$ representing the desired water volume in the dam at the end of the time horizon.

Both functions appear in the cost (9.1a) in the generic problem formulation.

9.2.3. Dynamic Programming like approaches

In the remainder of the chapter, we assume that we are in the so-called *white noise* setting.

Assumption 9.2.2. *Noises $\boldsymbol{W}_0, \dots, \boldsymbol{W}_{T-1}$ are independent over time.*

This assumption can be alleviated, in the case where it is possible to identify a dynamics in the noise process (such as an ARMA model), and by incorporating this new dynamics in the state variables (see Maceira and Damazio (2006) on this topic).

Under Assumption 9.2.2, Dynamic Programming (DP) applies to Problem (9.1): there is no optimality loss to seek each control $\boldsymbol{U}_t^i$ at time t as a function of both the state and the noise at time t. Then, the Bellman functions V_t are obtained by solving the Dynamic Programming equation backwards in time

$$V_T(x_T) = \sum_{i=1}^{N} K^i(x_T^i) \ , \tag{9.3a}$$

$$V_t(x_t) = \mathbb{E}\bigg(\min_{u_t^1, \dots, u_t^N} \sum_{i=1}^{N} L_t^i(x_t^i, u_t^i, \boldsymbol{W}_t) + V_{t+1}\big(f_t(x_t, u_t, \boldsymbol{W}_t)\big)\big)\bigg) \ . \tag{9.3b}$$

where $x_t = (x_t^1, \dots, x_t^N)$, $u_t = (u_t^1, \dots, u_t^N)$ and $f_t(x_t, u_t, \boldsymbol{W}_t)$ is the collection of new states $f_t^i(x_t^i, u_t^i, \boldsymbol{W}_t)$.

The DP equation is agnostic to whether the state and control variables are continuous or discrete, whether the constraints and the cost functions are convex or not, etc. However, in order to exhaustively solve the DP equation, we need to have discrete state, and to be able to solve each equation to optimality. In practice, the method is subject to the curse of dimensionality and cannot be used for large-scale optimization problems. For example, applying DP to dams management problems is practically untractable for more than five dams (see the results given at §9.4.3).

Another way to compute the Bellman functions associated to Problem (9.1) is to use the Stochastic Dual Dynamic Programming (SDDP) method. The method has been first described in Pereira and Pinto (1991), and its convergence has been analyzed in Philpott and Guan (2008) for the linear case and in Girardeau et al. (2014) for the general convex case. SDDP recursively constructs an approximation of each Bellman function as the supremum of a number of affine functions, thus exploiting the convexity of the Bellman functions (arising from the convexity of the cost and constraint functions and the linear dynamics). SDDP has been used for a long time for solving large-scale hydrothermal problems (see de Matos et al. (2015b) and the references therein) and allows to push the limits of DP in terms of state dimension (see the results given at §9.4.4).

9.2.4. Spatial coupling and approach by duality

A standard way to tackle large-scale optimization problems is to use Lagrange relaxation in order to split the original problem into a collection of smaller subproblems by dualizing coupling constraints. As far as Problem (9.1) is concerned, we have in mind to use DP for solving the subproblems and thus we want to dualize the spatial coupling constraints (9.1d) in order to formulate subproblems,

each incorporating a single dam. The associated Lagrangian $\mathcal{L}$ is accordingly

$$\mathcal{L}(\boldsymbol{X},\boldsymbol{U},\boldsymbol{\lambda}) = \mathbb{E}\Bigg(\sum_{i=1}^{N}\Bigg(\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_t) + K^i(\boldsymbol{X}_T^i) + \sum_{t=0}^{T-1}\boldsymbol{\lambda}_t\cdot\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_t)\Bigg)\Bigg)\,,$$

where the multiplier $\boldsymbol{\lambda}_t$ associated to Constraint (9.1d) is a random variable. From the measurability of the variables $\boldsymbol{X}_t^i$, $\boldsymbol{U}_t^i$ and $\boldsymbol{W}_t$, we can assume without loss of optimality that the multipliers $\boldsymbol{\lambda}_t$ are $\sigma(\boldsymbol{W}_0,\dots,\boldsymbol{W}_t)$-measurable random variables.

In order to be able to apply duality theory to the problem (which is mandatory for algorithmic resolution), we make the two following assumptions.

Assumption 9.2.3. *A saddle point of the Lagrangian $\mathcal{L}$ exists.*

Assumption 9.2.4. *The Uzawa algorithm applies to compute a saddle-point of $\mathcal{L}$ (see (Ekeland and Temam, 1999, Chap. VII) for a complete presentation).*

Assumption 9.2.3 corresponds to a Constraint Qualification condition and ensures the existence of an optimal multiplier. Assumption 9.2.4 allows to use a (dual) gradient ascent algorithm to compute the optimal multiplier. An important question in order to be able to satisfy these two assumptions is the choice of the spaces where the various random variables of the problem are living in. Duality theory and associated algorithms have been extensively studied in the framework of Hilbert spaces (see Ekeland and Temam (1999)), but the transition to the framework of stochastic optimal control poses difficult challenges (Rockafellar (1968, 1971)), which will be briefly presented at §9.3.4. One way to get rid of these difficulties is to assume that the space Ω is finite, assumption also needed for the convergence of SDDP.[1]

When using the Uzawa algorithm to compute a saddle-point of the Lagrangian, the minimization step with respect to $(\boldsymbol{X}^i,\boldsymbol{U}^i)_{i\in\mathrm{J1},N\mathrm{K}}$ splits in N independent subproblems each depending on a single pair $(\boldsymbol{X}^i,\boldsymbol{U}^i)$, and therefore allows for a dam by dam decomposition. More precisely, the k-th iteration of the Uzawa algorithm consists of the two following steps.

1. Solve Subproblem i, $i\in\mathrm{J1},N\mathrm{K}$, with fixed $\boldsymbol{\lambda}^{(k)}$:

$$\min_{\boldsymbol{X}^i,\boldsymbol{U}^i}\ \mathbb{E}\Big(\sum_{t=0}^{T-1} L_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_t) + \boldsymbol{\lambda}_t^{(k)}\cdot\Theta_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_t) + K^i(\boldsymbol{X}_T^i)\Big) \tag{9.4a}$$

$$s.t.\quad \boldsymbol{X}_{t+1}^i = f_t^i(\boldsymbol{X}_t^i,\boldsymbol{U}_t^i,\boldsymbol{W}_t)\,,\boldsymbol{X}_0^i \quad \text{given} \tag{9.4b}$$

$$\boldsymbol{U}_t^i \preceq \sigma(\boldsymbol{W}_0,\dots,\boldsymbol{W}_t)\,, \tag{9.4c}$$

   whose solution is denoted $\big(\boldsymbol{U}^{i,(k)},\boldsymbol{X}^{i,(k)}\big)$.

2. Use a gradient step to update the multipliers $\boldsymbol{\lambda}_t$:

$$\boldsymbol{\lambda}_t^{(k+1)} = \boldsymbol{\lambda}_t^{(k)} + \rho_t\Big(\sum_{i=1}^{N}\Theta_t^i\big(\boldsymbol{X}_t^{i,(k)},\boldsymbol{U}_t^{i,(k)},\boldsymbol{W}_t\big)\Big)\,. \tag{9.5}$$

Note that even if Subproblem (9.4) only involves the "physical" state variable $\boldsymbol{X}_t^i$ and the control variable $\boldsymbol{U}_t^i$, a situation which seems favorable to DP, it also involves two exogenous random

[1] Recall that the aim of the present chapter is mainly to present numerical results. The reader is referred to Leclère (2014) for these difficult theoretical questions.

processes, namely $\boldsymbol{W}$ and $\boldsymbol{\lambda}^{(k)}$. The white noise Assumption 9.2.2 applies for the first process $\boldsymbol{W}$, but not for the second one $\boldsymbol{\lambda}^{(k)}$, so that the state of the system cannot be summarized by the physical state $\boldsymbol{X}_t^i$! Moreover if we just use the fact that $\boldsymbol{\lambda}_t^{(k)}$ is measurable with respect to the past noises, the state of the system must incorporate all noises prior to time t, that is, $(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t)$. The state size of the subproblem increases with time. Without some additional knowledge on the process $\boldsymbol{\lambda}^{(k)}$, DP cannot be applied in a straightforward manner: something has to be compressed in order to use Dynamic Programming.

9.3. Dual Approximate Dynamic Programming

In Strugarek (2006), for a very specific instance of Problem (9.1), the author exhibited the dynamics of the optimal multiplier of the coupling constraint (9.1d). Hence it was possible to come down to the Markovian framework and to use DP to solve the subproblems (9.4) with an augmented space, namely the "physical" state $\boldsymbol{X}_t^i$ and the state associated with the multiplier's dynamics. Following this idea for a general Problem (9.1), Barty et al. (2010b) proposed to choose a parameterized dynamics for the multiplier: then solving the subproblems using DP became possible, the parameters defining the multiplier dynamics being updated at each iteration of the Uzawa algorithm. This new approach, called Dual Approximate Dynamic Programming (DADP), has then largely improved through a series of PhD theses (Girardeau (2010), Alais (2013) and Leclère (2014)) both from the theoretical and from the practical point of view. We give here a brief overview of the current DADP method.

9.3.1. DADP core idea and associated algorithm

In order to overcome the obstacle explained at §9.2.4 concerning the measurability of random variables $\boldsymbol{\lambda}_t^{(k)}$, we choose a random variable $\boldsymbol{Y}_t$ at each time t, each $\boldsymbol{Y}_t$ being measurable with respect to the noises $\big(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t\big)$ up to time t. We call $\boldsymbol{Y} = \big(\boldsymbol{Y}_0, \dots, \boldsymbol{Y}_{T-1}\big)$ the *information process* associated to Problem (9.1).

9.3.1.1. Method foundation

The core idea of DADP is to replace the multiplier $\boldsymbol{\lambda}_t^{(k)}$ by its conditional expectation $\mathbb{E}(\boldsymbol{\lambda}_t^{(k)} \mid \boldsymbol{Y}_t)$ with respect to $\boldsymbol{Y}_t$. From an intuitive point of view, the resulting optimization problem will be a good approximation of the original one if $\boldsymbol{Y}_t$ is close to the random variable $\boldsymbol{\lambda}_t^{(k)}$. Note that we require that the information process is not influenced by controls because introducing a dependency of the conditioning term with respect to the control would lead to very serious difficulties for optimization.

Using this core idea, we replace Subproblem (9.4) by:

$$\min_{\boldsymbol{X}^i, \boldsymbol{U}^i} \quad \mathbb{E}\Bigg(\sum_{t=0}^{T-1} \Big(L_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t) + K^i(\boldsymbol{X}_T^i) \Big) + \mathbb{E}(\boldsymbol{\lambda}_t^{(k)} \mid \boldsymbol{Y}_t) \cdot \Theta_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t) \Bigg) , \tag{9.6a}$$

$$\text{s.t.} \quad \boldsymbol{X}_{t+1}^i = f_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t) , \qquad \boldsymbol{X}_0^i \quad \text{given} , \tag{9.6b}$$

$$\boldsymbol{U}_t^i \preceq \sigma(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t) . \tag{9.6c}$$

According to the Doob property (Dellacherie and Meyer, 1975, Chapter 1, p. 18), the $\boldsymbol{Y}_t$-measurable

random variable $\mathbb{E}(\boldsymbol{\lambda}_t^{(k)} \mid \boldsymbol{Y}_t)$ can be represented by a measurable mapping $\mu_t^{(k)}$, that is,

$$\mu_t^{(k)}(y) = \mathbb{E}\big(\boldsymbol{\lambda}_t^{(k)} \;\big|\; \boldsymbol{Y}_t = y\big) \;, \tag{9.7}$$

so that Subproblem (9.6) in fact involves the two fixed random processes $\boldsymbol{W}$ and $\boldsymbol{Y}$. If the process $\boldsymbol{Y}$ follows a non-controlled Markovian dynamics driven by the noise process $\boldsymbol{W}$, i.e. if there exist functions h_t such that $\boldsymbol{Y}_{t+1} = h_t(\boldsymbol{Y}_t, \boldsymbol{W}_t)$ then $(\boldsymbol{X}_t^i, \boldsymbol{Y}_t)$ is a valid state for the subproblem and DP applies.

9.3.1.2. DADP algorithm

We assume that the process $\boldsymbol{Y}$ follows the dynamics $\boldsymbol{Y}_{t+1} = h_t(\boldsymbol{Y}_t, \boldsymbol{W}_t)$.

- The first step of the DADP algorithm at iteration k consists of solving all the subproblems (9.6) with $\boldsymbol{\lambda}_t^{(k)}$ fixed, that is, with $\mu_t^{(k)}(\cdot)$ given. It is done by solving the Bellman functions associated to each subproblem i, that is,

$$\begin{aligned} V_T^{i,(k)}(x^i, y) &= K^i(x) \;, \\ V_t^{i,(k)}(x^i, y) &= \mathbb{E}\big(Q_t^{i,(k)}(x^i, y, \boldsymbol{W}_t)\big) \;, \end{aligned}$$

  where $Q_t^{i,(k)}(x^i, y, w_t)$ is the value of

$$\begin{aligned} \min_{u^i} \quad & L_t^i(x^i, u^i, w_t) + \mu_t^{(k)}(y) \cdot \Theta_t^i(x^i, u^i, w_t) + V_{t+1}^{i,(k)}\big(x_{t+1}^i, y_{t+1}^i\big) \\ \text{s.t.} \quad & x_{t+1}^i = f_t^i(x^i, u^i, w_t) \;, \\ & y_{t+1} = h_t(y, w_t) \;. \end{aligned}$$

  Storing the argmin obtained during the Bellman resolution, we obtain the optimal feedback laws $\gamma_t^{i,(k)}$ as functions of both the state (x^i, y) and the noise w_t at time t. These functions allow to compute the optimal state and control processes $\big(\boldsymbol{U}^{i,(k)}, \boldsymbol{X}^{i,(k)}\big)$ of subproblem i at iteration k. Starting from $\boldsymbol{X}_0^{i,(k)} = \boldsymbol{X}_0^i$ the optimal control and state variables are obtained by applying the optimal feedback laws from $t = 0$ up to $T - 1$:

$$\begin{aligned} \boldsymbol{U}_t^{i,(k)} &= \gamma_t^{i,(k)}(\boldsymbol{X}_t^{i,(k)}, \boldsymbol{Y}_t, \boldsymbol{W}_t) \;, \\ \boldsymbol{X}_{t+1}^{i,(k)} &= f_t^i(\boldsymbol{X}_t^{i,(k)}, \boldsymbol{U}_t^{i,(k)}, \boldsymbol{W}_t) \;. \end{aligned}$$

- The second step of the DADP algorithm consists of updating the multiplier process $\boldsymbol{\lambda}^{(k)}$. Instead of updating the multipliers themselves by the standard gradient formula

$$\boldsymbol{\lambda}_t^{(k+1)} = \boldsymbol{\lambda}_t^{(k)} + \rho_t\Big(\sum_{i=1}^N \Theta_t^i\big(\boldsymbol{X}_t^{i,(k)}, \boldsymbol{U}_t^{i,(k)}, \boldsymbol{W}_t\big)\Big) \;, \tag{9.8}$$

  it is sufficient to deal with their conditional expectations w.r.t. $\boldsymbol{Y}_t$. Using the optimal processes $\boldsymbol{X}^{i,(k)}$ and $\boldsymbol{U}^{i,(k)}$ obtained at the previous step of the algorithm for all subproblems, the *conditional* deviation from the coupling constraint is represented by a measurable mapping $\Delta_t^{(k)}$:

$$\Delta_t^{(k)}(y_t) = \mathbb{E}\Big(\sum_{i=1}^N \Theta_t^i\big(\boldsymbol{X}_t^{i,(k)}, \boldsymbol{U}_t^{i,(k)}, \boldsymbol{W}_t\big) \;\Big|\; \boldsymbol{Y}_t = y_t\Big) \;. \tag{9.9}$$

Gathering the functional representations (9.7) and (9.9) of the conditional multiplier and of the conditional deviation, the gradient update reduces to the following functional expression:

$$\mu_t^{(k+1)}(\cdot) = \mu_t^{(k)}(\cdot) + \rho_t \Delta_t^{(k)}(\cdot) \ . \tag{9.10}$$

This last equation is equivalent to the multipliers conditional expectation update:

$$\mathbb{E}\big(\boldsymbol{\lambda}_t^{(k+1)} \ \big| \ \boldsymbol{Y}_t\big) = \mathbb{E}\big(\boldsymbol{\lambda}_t^{(k)} \ \big| \ \boldsymbol{Y}_t\big) + \rho_t \, \mathbb{E}\Big(\sum_{i=1}^{N} \Theta_t^i\big(\boldsymbol{X}_t^{i,(k)}, \boldsymbol{U}_t^{i,(k)}, \boldsymbol{W}_t\big) \ \Big| \ \boldsymbol{Y}_t \Big) \ . \tag{9.11}$$

From a practical point of view, computing the gradients using Formula (9.9), instead of (9.8) opens the way to important numerical improvements in the DADP algorithm. Indeed, instead of a gradient formula in a large space, we can use more sophisticated direction descent algorithms: as a matter of fact, if the support of the random variable $\boldsymbol{Y}_t$ is finite, it becomes possible to efficiently implement a quasi-Newton method, thus obtaining a much faster convergence than the one of the standard gradient ascent method (see §9.4.3.2 for details).

DADP algorithm is depicted in Figure 9.2.

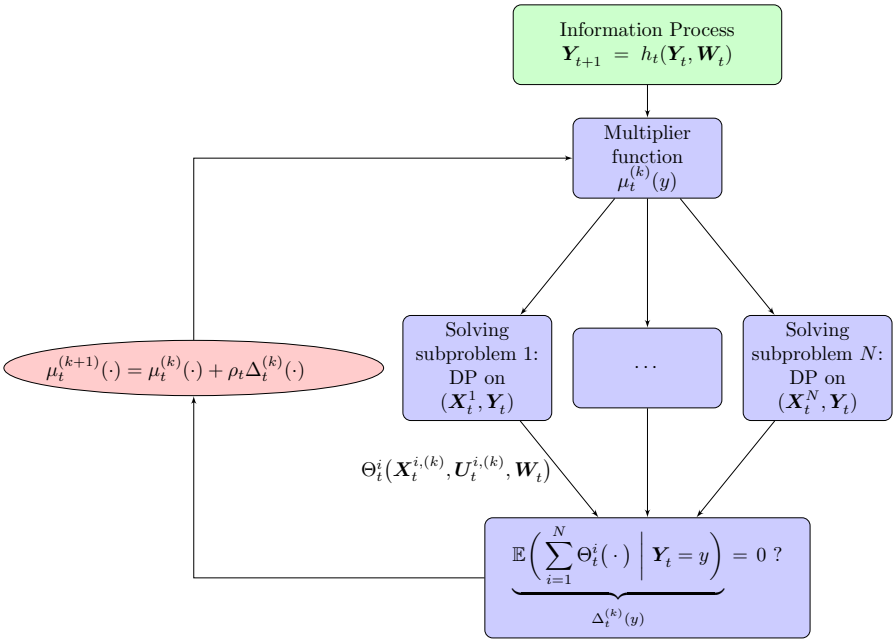


Figure 9.2.: DADP flowchart.

9.3.2. DADP interpretations

The DADP method, as it has been presented up to now, makes use of an *approximation* of the optimal multiplier, that is, the multiplier $\boldsymbol{\lambda}_t$ is replaced by its conditional expectation $\mathbb{E}\big(\boldsymbol{\lambda}_t \ \big| \ \boldsymbol{Y}_t\big)$. Such an approximation is equivalent to a *decision-rule* approach for the dual problem (see also Kuhn et al. (2011)), obtained by imposing that the dual variable $\boldsymbol{\lambda}_t$ is measurable with respect to $\boldsymbol{Y}_t$.

DADP may also be viewed as a *relaxation* of the constraints in the primal problem. More precisely, we replace the almost sure coupling constraint (9.1d) by the following conditional expectation constraint

$$\mathbb{E}\Big(\sum_{i=1}^{N} \Theta_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t) \;\Big|\; \boldsymbol{Y}_t\Big) = 0\ . \tag{9.12}$$

Proposition 9.3.1. *Assume that the Lagrangian associated with this relaxed problem has a saddle point. Then the DADP algorithm on Problem* (9.1) *can be interpreted as the Uzawa algorithm applied to the relaxed Problem.*

Proof. Consider the duality term $\mathbb{E}\big(\mathbb{E}(\boldsymbol{\lambda}_t^{(k)} \mid \boldsymbol{Y}_t) \cdot \Theta_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t)\big)$ which appears in the cost function of subproblem i in DADP. This term can be written equivalently $\mathbb{E}\big(\boldsymbol{\lambda}_t^{(k)} \cdot \mathbb{E}(\Theta_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t) \mid \boldsymbol{Y}_t)\big)$, which corresponds to the dualization of the coupling constraint handled in the relaxed problem. □

DADP thus consists of replacing an *almost-sure* constraint by its *conditional expectation* w.r.t. the information variable $\boldsymbol{Y}_t$. From this interpretation, we deduce that the optimal value provided by DADP is a guaranteed *lower bound* of the optimal value of Problem (9.1).

9.3.3. Admissibility recovery

Solving the relaxed problem, that is, Problem (9.1) where constraints (9.1d) are replaced by the less binding constraints (9.12), does not necessarily yield a solution admissible for Problem (9.1). Nevertheless it produces at each time t a set of N local Bellman functions $V_t^{i,\infty}$, each depending on the extended state (x_t^i, y_t). We use these functions to produce an *approximation* V_t^∞ of the "true" Bellman function V_t of the global state $\big(x_t^1, \ldots, x_t^N\big)$ by simply summing the local Bellman functions:

$$V_t^\infty\big(x_t^1, \ldots, x_t^N, y_t\big) = \sum_{i=1}^{N} V_t^{i,\infty}\big(x_t^i, y_t\big)\ .$$

We then obtain an admissible feedback policy for Problem (9.1): for any value of the state $\big(x_t^1, \ldots, x_t^N\big)$, any value of the information y_t and any value of the noise w_t at time t, the control value is obtained by solving the following one-step DP problem

$$\begin{aligned}
\min_{(u_t^1, \ldots, u_t^N)} \quad & \sum_{i=1}^{N} L_t^i\big(x_t^i, u_t^i, w_t^i\big) + V_{t+1}^\infty\big(x_{t+1}^1, \ldots, x_{t+1}^N, y_{t+1}\big)\ , \\
\text{s.t.} \quad & x_{t+1}^i = f_t^i\big(x_t^i, u_t^i, w_t^i\big)\ ,\ \ i \in \mathrm{J1}, N\mathrm{K}\ , \\
& y_{t+1} = h_t\big(y_t^i, w_t^i\big)\ , \\
& \sum_{i=1}^{N} \Theta_t^i(x_t^i, u_t^i, w_t) = 0\ .
\end{aligned}$$

In this framework, DADP can be viewed as a tool allowing to compute approximated Bellman functions for Problem (9.1) which in turns yields an online admissible feedback policy for Problem (9.1).

Applying this online feedback policy along a bunch of noises scenarios allows to compute a Monte Carlo approximation of the cost, which is accordingly a stochastic *upper bound* of the optimal value of Problem (9.1).

9.3.4. Theoretical and practical questions

Theoretical questions linked to DADP are addressed in Leclère (2014), and practical ones in Girardeau (2010) and Alais (2013).

9.3.4.1. Theoretical questions

In the DADP approach, we treat the coupling constraints of a stochastic optimization problem by duality methods and solve it using the Uzawa algorithm. The Uzawa algorithm is a dual ascent method which is usually described in an Hilbert space such as $\mathrm{L}^2(\Omega, \mathcal{A}, \mathbb{P}, \mathbb{R}^n)$, but we cannot guarantee the existence of an optimal multiplier in such a space. To overcome the difficulty, the approach consists of extending the setting to the non-reflexive Banach space $\mathrm{L}^\infty(\Omega, \mathcal{A}, \mathbb{P}, \mathbb{R}^n)$, to give conditions for the existence of an optimal multiplier in $\mathrm{L}^1\big(\Omega, \mathcal{A}, \mathbb{P}; \mathbb{R}^n\big)$ (rather than in the dual space of L^∞) and to study the Uzawa algorithm convergence in this space.

9.3.4.2. Practical questions

An important practical question is the choice of the information variables $\boldsymbol{Y}_t$. We present here some possibilities.

1. *Perfect memory*: $\boldsymbol{Y}_t = \big(\boldsymbol{W}_0, \dots, \boldsymbol{W}_t\big)$.
   From the measurability properties of $\boldsymbol{\lambda}_t^{(k)}$, we have $\mathbb{E}(\boldsymbol{\lambda}_t^{(k)} \mid \boldsymbol{Y}_t) = \boldsymbol{\lambda}_t^{(k)}$, that is, there is no approximation! Indeed a valid state for each subproblem is $\big(\boldsymbol{X}_t, \boldsymbol{W}_0, \dots, \boldsymbol{W}_t\big)$: the state is growing with time.

2. *Minimal information*: $\boldsymbol{Y}_t = 0$.
   Here $\boldsymbol{\lambda}_t^{(k)}$ is approximated by its expectation $\mathbb{E}(\boldsymbol{\lambda}_t^{(k)})$. The information variable does not deliver any online information, and a valid state for subproblem i is $\boldsymbol{X}_t^i$.

3. *Dynamic information*: $\boldsymbol{Y}_{t+1} = h_t\big(\boldsymbol{Y}_t, \boldsymbol{W}_{t+1}\big)$.
   This choice corresponds to a number of possibilities, as mimicking the state of another unit, or adding a hidden dynamics. A valid state for subproblem i is $\big(\boldsymbol{X}_t^i, \boldsymbol{Y}_t\big)$.

The question of accelerating the DADP algorithm by using a more sophisticated method than the simple gradient ascent method in the multiplier update step has been discussed at the end of §9.3.1.2. Numerical experiments have shown that it has a great impact on the convergence speed of the method (see §9.4.3.2). Another improvement would be to replace the standard Lagrangian by an augmented Lagrangian.

9.4. Numerical experiments

In this section, we present numerical results obtained on a large selection of hydro valleys. Some of these valleys (see Figure 9.4) correspond to academic examples, in the sense that their characteristics (size of dams, range of controls, inflows values) do not rely on existing valleys. These examples allow us to quantify the performance of different optimization methods (DP, DADP and SDDP) on problems of increasing size, from a valley incorporating 4 dams, and thus solvable by DP, up to a valley with 30 dams, and thus facing the curse of dimensionality (§9.4.3 and §9.4.4). We also present two instances corresponding to more realistic hydro valleys, where the models respect the orders of magnitude of the dam sizes of existing valleys (§9.4.5).

All the results presented here have been obtained using a 3.4GHz, 4 cores – 8 threads Intel® Xeon® E3 based computer.

9.4.1. Application of DADP to a hydro valley

We go back to the problem formulation presented at §9.2.2. In order to implement the DADP algorithm, we dualize the coupling constraints

$$\boldsymbol{Z}_t^{i+1} - g_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t^i, \boldsymbol{Z}_t^i) = 0\ , \tag{9.13}$$

and we denote by $\boldsymbol{\lambda}_t^{i+1}$ the associated multiplier (random variable).

When minimizing the dual problem at iteration k of the algorithm, the product of (9.13) with a given multiplier by $\boldsymbol{\lambda}_t^{i+1,(k)}$ is additive with respect to the dams, that is, the term $-\boldsymbol{\lambda}_t^{i+1,(k)} \cdot g_t^i\big(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t^i, \boldsymbol{Z}_t^i\big)$ pertains to dam i subproblem, whereas the term $\boldsymbol{\lambda}_t^{i+1,(k)} \cdot \boldsymbol{Z}_t^{i+1}$ pertains to dam $i+1$ subproblem, hence leading to a dam by dam decomposition for the dual problem maximization in $(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{Z})$ at $\boldsymbol{\lambda}_t^{i+1,(k)}$ fixed. This decoupling is illustrated in Figure 9.3.

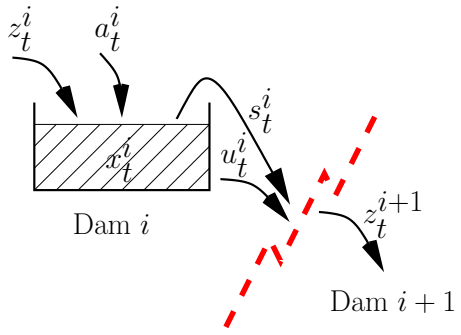


Figure 9.3.: Decomposition by dam.

9.4.1.1. DADP implementation

The DADP method consists of choosing a multiplier process $\boldsymbol{Y}$ and then replacing the coupling constraints by their conditional expectations with respect to $\boldsymbol{Y}_t$. Here we adopt the choice $\boldsymbol{Y}_t = 0$ (minimal information), so that Constraints (9.13) are replaced in the approximated problem by their expectations:

$$\mathbb{E}\big(\boldsymbol{Z}_t^{i+1} - g_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t^i, \boldsymbol{Z}_t^i)\big) = 0\ . \tag{9.14}$$

The expression of Subproblem (9.6) attached to dam i reads

$$\min_{\boldsymbol{U}^i, \boldsymbol{Z}^i, \boldsymbol{X}^i} \quad \mathbb{E}\Bigg(\sum_{t=0}^{T-1} \Big(L_t^i\big(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t^i, \boldsymbol{Z}_t^i\big) \ + \ \mathbb{E}\big(\boldsymbol{\lambda}_t^{i,(k)}\big) \cdot \boldsymbol{Z}_t^i \tag{9.15a}$$

$$- \ \mathbb{E}\big(\boldsymbol{\lambda}_t^{i+1,(k)}\big) \cdot g_t^i\big(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t^i, \boldsymbol{Z}_t^i\big)\Big) + K^i\big(\boldsymbol{X}_T^i\big)\Bigg)\ , \tag{9.15b}$$

$$s.t. \qquad \boldsymbol{X}_{t+1}^i = f_t^i(\boldsymbol{X}_t^i, \boldsymbol{U}_t^i, \boldsymbol{W}_t)\ , \qquad \boldsymbol{X}_0^i \quad \text{given} \tag{9.15c}$$

$$\boldsymbol{U}_t^i \ \preceq \sigma(\boldsymbol{W}_0, \ldots, \boldsymbol{W}_t)\ . \tag{9.15d}$$

Because of the crude relaxation due to a constant $\boldsymbol{Y}_t^i$, the multipliers $\boldsymbol{\lambda}_t^{i,(k)}$ appear only in the subproblems by means of their expectations $\mathbb{E}(\boldsymbol{\lambda}_t^{i,(k)})$, so that all subproblems involve a 1-dimensional state variable, that is, the dam stock $\boldsymbol{X}_t^i$, and hence are easily solvable by Dynamic Programming. We denote by $\big(\boldsymbol{U}^{i,(k)}, \boldsymbol{Z}^{i,(k)}, \boldsymbol{X}^{i,(k)}\big)$ the optimal solution of each subproblem i, and by $V_t^{i,(k)}(x^i)$

the Bellman function obtained for each dam i at time t.

With the choice of constant information variables $\boldsymbol{Y}_t^i$, the coordination update step (9.11) reduces to

$$\mathbb{E}\big(\boldsymbol{\lambda}_t^{i,(k+1)}\big) = \mathbb{E}\big(\boldsymbol{\lambda}_t^{i,(k)}\big) + \rho_t \mathbb{E}\Big(\boldsymbol{Z}_t^{i+1,(k)} - g_t^i\big(\boldsymbol{X}_t^{i,(k)}, \boldsymbol{U}_t^{i,(k)}, \boldsymbol{W}_t^i, \boldsymbol{Z}_t^{i,(k)}\big)\Big)\,, \tag{9.16}$$

that is, a collection of deterministic equations involving the expectation of (9.13) which is easily estimated by a Monte Carlo approach.

Assume that DADP converges, leading to Bellman functions $V_t^{i,\infty}$. We know that the initial almost-sure coupling constraints (9.13) are not satisfied. To recover admissibility, we use the heuristic rule proposed at §9.3.3, solving the following deterministic one-step DP problem:

$$\min_{(u^1,\dots,u^N)} \quad \sum_{i=1}^{N} L_t^i\big(x^i, u^i, w_t^i, z^i\big) + V_{t+1}^{\infty}\big(x_{t+1}^1, \dots, x_{t+1}^N\big)\,, \tag{9.17a}$$

$$s.t. \quad x_{t+1}^i = f_t^i\big(x^i, u^i, w_t^i, z^i\big) \quad \forall i\,, \tag{9.17b}$$

$$z^{i+1} = g_t^i(x^i, u^i, w_t^i, z^i) \quad \forall i\,. \tag{9.17c}$$

9.4.1.2. Complete process

We can summarize the whole process as follows. In the *optimization stage*, we first compute the local Bellman functions $V_t^{i,\infty}$ and form the approximate global Bellman functions V_t^{∞} by summing the local ones. In the *simulation stage*, we evaluate by Monte-Carlo the strategy induced by V_t^{∞}. We draw a large number of noise scenario, and compute the admissible control values along each scenario by solving Problem (9.17), from $t = 0$ to $t = T - 1$, and storing payoffs.

9.4.2. SDDP implementation

As explained in §9.4.3, the controls of the original problem are discrete, which is a difficulty for SDDP implementation (although a recent extension has been proposed in Zou et al. (2017)). In the *optimization stage* we relax the integrity constraints to obtain relaxed Bellman value functions V_t^{∞}. Furthermore, we consider that the spillage is a control variable, so as to render the dynamics linear (convex framework of SDDP). Then, in the *simulation stage*, we use these relaxed Bellman value functions to design policies taking into account the discrete controls by solving problems akin to Problem (9.17).

The whole process of SDDP is as follows. In the optimization stage, lower approximations of Bellman functions V_t are built iteratively. At iteration k, the procedure consists of two passes.

- During the *forward pass*, we sample a scenario of noise. We then simulate a stock trajectory by using the current approximation of the Bellman functions. This is done by successively solving one-step DP problem, akin to Problem (9.17), where V_{t+1}^{∞} is replaced by its current piecewise linear outer-approximation, to determine the next stock value. Note that each of these one-step DP problem is a continuous quadratic programming (QP) problem.

- In the *backward pass*, duality theory allows to find subgradient of lower approximations of the Bellman functions. This subgradients are computed along the trajectory obtained during the forward pass, and used to construct valid cuts, that is, hyperplanes that are lower than the Bellman functions. Those cuts are then added to the current outer-approximations of the Bellman functions.

In order to assess the convergence of the SDDP algorithm, we compute (say every 20 iterations of SDDP) a Monte Carlo approximation of the expected cost value with its associated 95% confidence interval, and we compare the upper value of the confidence interval with the lower bound provided by SDDP up to a given threshold in order to stop the algorithm (see Shapiro (2011)). In our

experiments, the Monte Carlo simulation has been made using 10,000 scenarios, and the relative convergence threshold was around 0.5%. The *simulation stage* is identical to the one described at §9.4.1.2 using the global Bellman's value function obtained by SDDP .

We have used a version of SDDP implemented in Julia (StochDynamicProgramming package[2]) built on top of the JuMP package used as a modeler (see Dunning et al. (2017)). The QP problems are solved using CPLEX 12.5. Every 10 iterations, redundant cuts are removed thanks to the limited memory level-1 heuristic described in Guigues (2017). Indeed, without cuts removal, the resolution of each QP becomes too slow as the number of cuts increases along iterations.

9.4.3. Results obtained for academic valleys

We model a first collection of hydro valleys including from 4 to 12 dams, with arborescent geometries (see Figure 9.4).

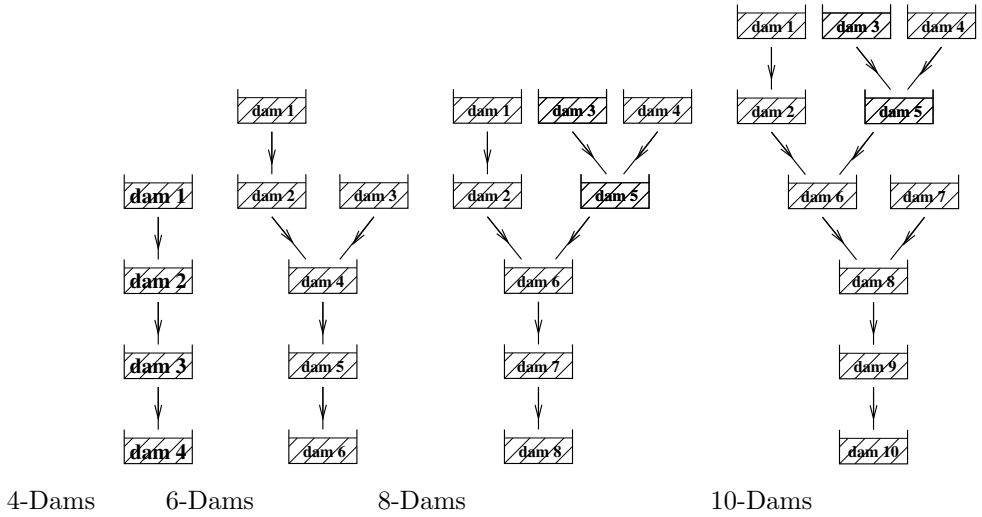


Figure 9.4.: Some academic examples of hydro valleys.

The optimization problem is stated on a time horizon of one year, with a monthly time step ($T = 12$). All the dams have more or less the same maximal volume. The maximal amount of turbinated water for each dam varies with the location of the dam in the valley (more capacity for a downstream dam than for an upstream dam), as well as the random inflows in a dam (more inflow for an upstream dam than for a downstream dam). We assume discrete probability laws with finite supports for the inflows, and deterministic market prices. We also assume that the available turbine controls are discrete, so that each dam is in fact modeled using a discrete Markov chain. These valleys do not correspond to realistic valleys, in the sense that a true valley incorporates dams of very heterogeneous size.

9.4.3.1. SDDP convergence

We first illustrate the convergence of the SDDP algorithm for the 8-Dams valley on Figure 9.5 (note that most of the valleys display a similar convergence pattern). As explained at §9.4.2, the exact lower bound given by SDDP (black curve) increases along the iterations, and the gap between this lower bound and the upper value of the confidence interval (red curve) is less than 0.5% at iteration 140.

[2]See the github link `https://github.com/JuliaOpt/StochDynamicProgramming.jl`.

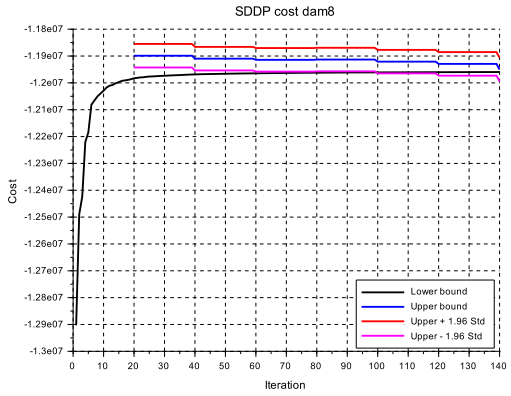


Figure 9.5.: Convergence of SDDP for the 8-Dams valley

Note that, in our experiments, this stopping criterion approximately matches the classical SDDP convergence stopping criterion proposed in Pereira and Pinto (1991) corresponding to the fact that the lower bound provided by SDDP becomes greater than the lower value of the confidence interval.

9.4.3.2. DADP convergence

Let us first detail the method used for the update of the multipliers involved by DADP. Thanks to the choice of constant information variables, the gradient expression involved in the update formula (9.16) is an expectation, that can be approximated by a Monte Carlo approach. We draw a collection of statistically independent scenarios of $\{\boldsymbol{W}_t\}$ and then compute at iteration k of DADP the optimal solutions $\left\{\boldsymbol{X}_t^{i,(k)}, \boldsymbol{U}_t^{i,(k)}, \boldsymbol{Z}_t^{i,(k)}\right\}$ of Subproblem (9.15) along each scenario. One has to note that this collection of scenarios is independent of the one used during the simulation stage of the complete process described at §9.4.1.2. We thus obtain realizations of $\left(\boldsymbol{Z}_t^{i+1,(k)} - g_t^i(\boldsymbol{X}_t^{i,(k)}, \boldsymbol{U}_t^{i,(k)}, \boldsymbol{W}_t^i, \boldsymbol{Z}_t^{i,(k)})\right)$, whose arithmetic mean gives the (approximated) gradient component at time t for the coupling between dam i and dam $i+1$. This gradient can be used either in the standard steepest ascent method such as in (9.16), or in a more sophisticated algorithm such as the conjugate gradient or the quasi-Newton method. We use in our numerical experiments a solver (limited memory BFGS) of the MODULOPT library from INRIA by Gilbert and Jonsson (2007). For all the valleys we studied, the convergence was fast (around 100 iterations regardless of the problem size). Figure 9.6 represents the evolution of the multipliers λ_t^i for the 8-Dams valley along the iterations of the algorithm.

The order of magnitude of the optimal multipliers decreases with the geographical position of the link in the hydro valley. Nevertheless, the convergence rate is very similar for all links: this practical consideration remains true for almost all valleys, and it explains why the number of iterations required for the DADP convergence does not vary too much with the size of the valley.

9.4.3.3. Methods comparison

We solve Problem (9.1) for the first collection of academic valleys by 3 different methods:

1. the standard Dynamic Programming method (DP), when possible,
2. the SDDP presented at §9.4.2,
3. the DADP method.

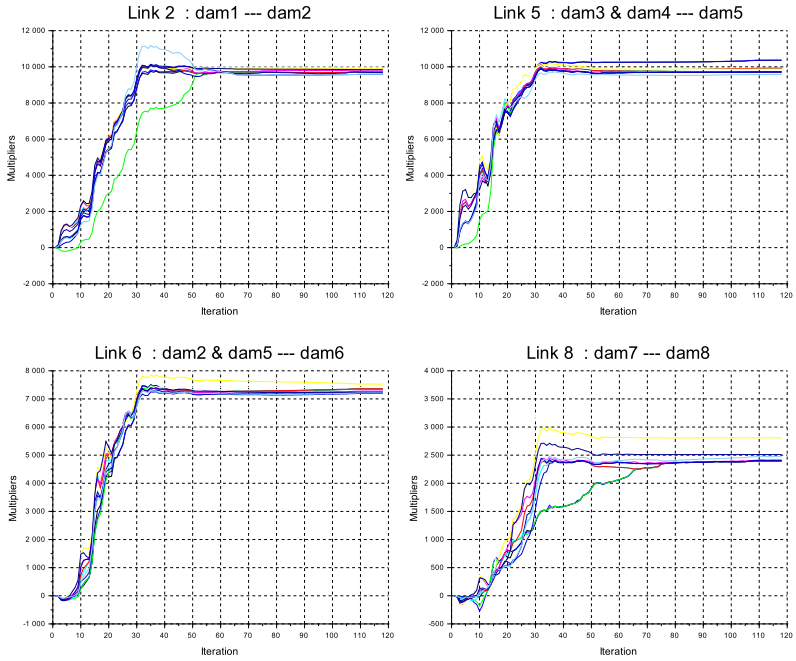


Figure 9.6.: 8-Dams multipliers: `dam1`→`dam2` , `dam3-4`→`dam5` , `dam2-5`→`dam6` , `dam7`→`dam8`

All these methods produce Bellman functions (optimization stage described at §9.4.1.2), whose quality is evaluated by the simulation stage of §9.4.1.2. The obtained results are given in Table 9.1. The lines "CPU time" correspond to the time (in minute) needed to compute the Bellman functions (optimization stage only), whereas the lines "value" indicate the cost obtained by Monte Carlo on the initial model (simulation stage, performed using a 100,000 scenarios sample, except for the 12-Dams valley for which a smaller sample set was used in order to reduce the computational load). The comparisons between the different cost values for the same valley are thus relevant. For both SDDP and DADP, we also give the lower bound corresponding to the Bellman value obtained at the end of the optimization stage.

Valley	**4-Dams**	**6-Dams**	**8-Dams**	**10-Dams**	**12-Dams**
DP CPU time	*1600'*	*$\sim 10^8$'*	$\sim \infty$	$\sim \infty$	$\sim \infty$
DP value	−3743	N.A.	N.A.	N.A.	N.A.
SDDP CPU time	*6'*	*10'*	*13'*	*50'*	*97'*
SDDP value	−3742	−7027	−11830	−17070	~ -17000
SDDP lower bound	*−3754*	*−7050*	*−11960*	*−17260*	*−19490*
DADP CPU time	*7'*	*12'*	*18'*	*24'*	*22'*
DADP value	−3667	−6816	−11570	−16760	~ -17000
DADP lower bound	*−3996*	*−7522*	*−12450*	*−17930*	*−20480*
Gap DADP/SDDP	**2.0%**	**3.0%**	**2.2%**	**1.8%**	?

Table 9.1.: Results obtained by DP, SDDP and DADP

We first note that a direct use of DP is only possible for the 4-Dams valley: it corresponds to the well-known curse of dimensionality inherent to DP. The value given by DP is the true optimal cost value for the 4-Dams valley and can be used as the reference value. The SDDP method, although relying on the integrity constraints relaxation in the optimization stage (hence a not so tight

lower bound), gives excellent results for the 4-Dams valley: we thus elect SDDP as the reference method in order to evaluate the DADP method. Note that the CPU time remains reasonable, the optimization problems inside SDDP corresponding to a continuous linear-quadratic formulation (here solved using the CPLEX commercial solver).

Remark 9.4.1. *Note however that all the methods we are comparing face the curse of dimensionality associated to the combinatorics of the control during the simulation stage, as the controls associated to the whole valley have to be enumerated at each time t along each scenario. This is the reason why the values obtained for the 12-Dams valley have been computed using* $1,000$ *scenarios (*$100,000$ *for the others valleys) and hence are not so accurate.* ◊

We now turn to the DADP method. We first notice that the lower bound given by the method is rather bad (as a consequence of solving a problem with relaxed coupling constraints in the optimization stage), but the values obtained in the simulation stage are reasonable compared to the ones given by SDDP (as indicated by the last line of Table 9.1). The most noticeable point is that the CPU time needed for the optimization stage seems to grow more slowly for DADP than for SDDP. This aspect will be highlighted in §9.4.4.

Let us finally materialize more finely the difference in the results between SDDP and DADP. Beyond average values given in Table 9.1, Figure 9.7 represents the payoff empirical probability laws (optimal cost over the time horizon), obtained by the simulation stage using 100,000 scenarios, for both SDDP and DADP. We observe that, although the expectations are fairly close, the shapes of the two distributions differ significantly.

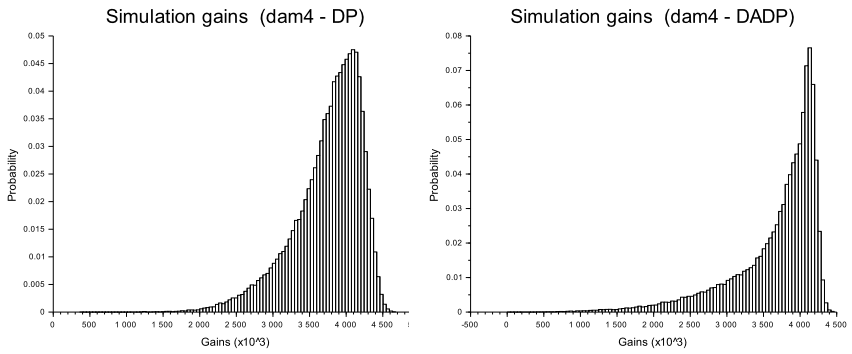


Figure 9.7.: 4-Dams payoff distributions: SDDP (left) — DADP (right)

9.4.4. Challenging the curse of dimensionality

The experiments made in §9.4.3 seem to indicate that DADP is less sensitive to the size of the valley than the SDDP method. In order to validate this observation, we design a new collection of academic hydro valleys incorporating from 14 up to 30 dams. It is of course no longer possible to perform the simulation stage for these instances: the combinatorics induced by the set of possible values of the controls is too large to allow simulation of the valley behavior along a large set of scenarios. We thus limit ourselves to the computation of the Bellman functions (optimization stage). The corresponding results are reported in Table 9.2.

It appears that the CPU time required for the DADP method grows linearly with the number of dams, while the growth rate of SDDP is more or less exponential. Figure 9.8 shows how the CPU time varies for the three methods. As expected, DP is only implementable for small instances, say

Valley	**14-Dams**	**18-Dams**	**20-Dams**	**25-Dams**	**30-Dams**
SDDP CPU time	*210'*	*585'*	*970'*	*1560'*	*2750'*
SDDP lower bound	*−32024*	*−46917*	*−61454*	*−79440*	*−100430*
DADP CPU time	*40'*	*50'*	*75'*	*140'*	*150'*
DADP lower bound	*−32981*	*−48095*	*−62802*	*−80993*	*−101990*

Table 9.2.: SDDP and DADP comparison for large academic valleys

up to 5 dams. Eventually, the limits of SDDP and DADP have not really be reached, but DADP displays a near linear rate of CPU time allowing to tackle instances of even greater size.

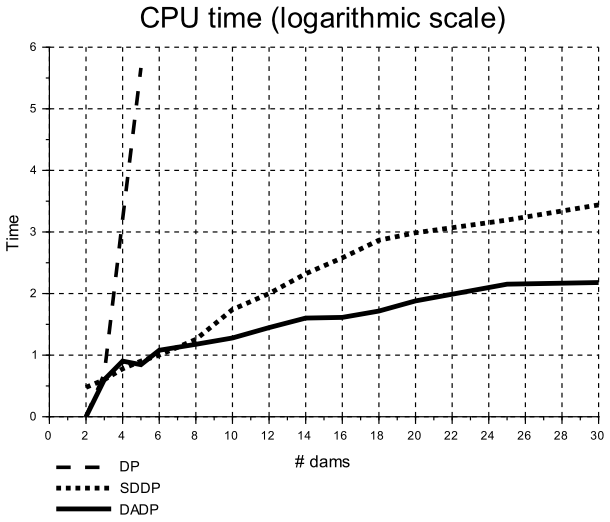


Figure 9.8.: CPU time comparison

9.4.5. Results for two realistic valleys

We finally model two hydro valleys corresponding to existing systems in France, namely the Vicdessos valley and the Dordogne river (see Figure 9.9).

The optimization problem is stated again on a one year horizon, with a monthly time step. What mainly differs here from the academic examples used at §9.4.3 are the characteristics of the dams. For example, the Dordogne river valley encompasses large dams (as "Bort" whose capacity is, say, 400) and small dams (as "Mareges" the capacity of which is equal to 35, that is, ten times smaller). This heterogeneity induces numerical difficulties, for example the requirement to have a wide range of possible controls for the small downstream reservoirs, or the need to use a fine discretization for the state grid in DP-like methods. We again assume discrete probability laws with finite support for the inflows, and we also assume that the available turbine controls are discrete.

The comparison results of SDDP and DADP are given in Table 9.3. As for the academic examples, SDDP displays the best results and is therefore used as the reference. The large number of possible discrete controls penalizes the CPU time of DADP, although the gap between SDDP and DADP remains limited.

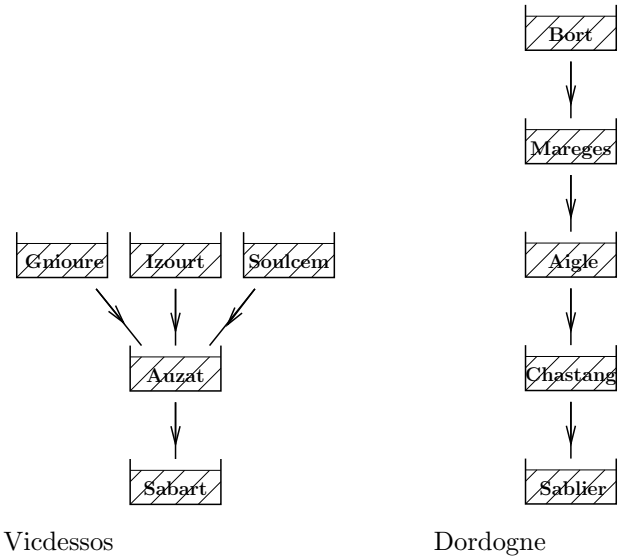


Figure 9.9.: Two realistic hydro valleys

Valley	**Vicdessos**	**Dordogne**
SDDP CPU time	*9'*	*17'*
SDDP value	−2244	−22150
SDDP lower bound	*−2258*	*−22310*
DADP CPU time	*9'*	*210'*
DADP value	−2238	−21650
DADP lower bound	*−2286*	*−22990*
Gap DADP/SDDP	**0.3**%	**2.2**%

Table 9.3.: Results obtained by SDDP and DADP

9.5. Conclusion

In this article, we have depicted a method called DADP which allows one to tackle large-scale stochastic optimal control problems in discrete time, such as the ones found in the field of energy management. We have presented the practical aspects of the method, without deepening in the theoretical issues arising in the foundations of the method. Lots of numerical experiments have been presented on hydro valley problems ("chained models"), which complements the ones already made on unit commitment problems ("flower models") Barty et al. (2010b). The main conclusions are that DADP converges fast and gives near-optimal results even when using a "crude" relaxation (here a constant information process $\boldsymbol{Y}$). More precisely, DADP allows one to deal with optimization problems that are out of the scope of standard Dynamic Programming, and beats SDDP for very large hydro valleys in terms of CPU time. We thus hope to be able to implement DADP for very large stochastic optimal control problems such as the ones encountered in smart management of urban districts, involving hundreds of houses and thus hundred of states variables. Such problems are formulated on a short-term time scale (typically a one day horizon with 15 minutes time steps), and incorporate on/off devices. In that new context, controls will have to be modeled using discrete variables (whereas this assumption was not mandatory for the study presented in this chapter). Moreover, on a short-term time scale, the randomness of the markets

prices plays an important role, and it will thus be necessary to take them into account as a noise in the problem.

We plan to extend this study in two directions. First to implement the DADP method for general spatial structures (not only "flower models" or "chain model", but "smart-grid models" involving a generic graph). The second direction is to implement more sophisticated decomposition methods than price decomposition. On the one hand we want to use decomposition schemes such that resource allocation or interaction prediction principle Cohen (1978). On the other hand we want to use augmented Lagrangian based methods such as alternating direction method of multiplier (ADMM) and proximal decomposition algorithm (PDA) for decomposition in order to obtain the nice convergence properties of this kind of methods (see Lenoir and Mahey (2017) for a survey).

Finally, let us mention that a theoretical work has begun in order to provide foundations of the method (Leclère (2014)). It includes conditions for existence of a multiplier in the L^1 space when the optimization problem is posed in L^∞ and conditions for convergence of the Uzawa algorithm in L^∞. A lot of work remains to be done on these questions, mainly to relax the continuity assumption in order to be able to deal with extended functions, and to obtain more general assumptions ensuring the convergence of the Uzawa algorithm.

Part III.

Contributions to Stochastic Dual Dynamic Programming

Abstract. In Part I, we applied Stochastic Dual Dynamic Programming (SDDP) to compute Bellman functions corresponding to small scale microgrid systems. In Part II, we studied the behavior of SDDP when applied to large scale systems, and illustrated its limits in term of convergence and computation time.
However, one of the main drawbacks of SDDP is the lack of proper stopping criterion: the algorithm usually stops when the lower bound becomes close to the upper bound, ordinarily estimated statistically. We propose in Chapter 10 a new algorithm to compute a deterministic upper bound by considering a dual version of SDDP. An abstract version of SDDP is presented, that allows to extend the classical SDDP algorithm. We give in Chapter 11 a proof of convergence for the abstract SDDP algorithm.
Chapter 10 is a copy of Leclère et al. (2018). The author is deeply indebted to his co-authors for this work. Chapter 11 is a joint work with Maël Forcier. The author thanks him warmly for his precious help.

Chapter 10.

Exact converging bounds for Stochastic Dual Dynamic Programming via Fenchel duality

This chapter is a transcription of the article Leclère et al. (2018). The author thanks his co-authors Pierre Carpentier, Jean-Philippe Chancelier, Arnaud Lenoir and Vincent Leclère.

Contents

The Stochastic Dual Dynamic Programming (SDDP) algorithm has become one of the main tools to address convex multistage stochastic optimal control problem. Recently a large amount of work has been devoted to improve the convergence speed of the algorithm through cut-selection and regularization, or to extend the field of applications to non-linear, integer or risk-averse problems. However one of the main downside of the algorithm remains the difficulty to give an upper bound

of the optimal value, usually estimated through Monte Carlo methods and therefore difficult to use in the algorithm stopping criterion.

In this chapter we present a dual SDDP algorithm that yields a converging exact upper bound for the optimal value of the optimization problem. Incidently we show how to compute an alternative control policy based on an inner approximation of Bellman value functions instead of the outer approximation given by the standard SDDP algorithm. We illustrate the approach on an energy production problem involving zones of production and transportation links between the zones. The numerical experiments we carry out on this example show the effectiveness of the method.

10.1. Introduction

In this paper, we consider a risk neutral multistage stochastic optimization problem, with continuous decision variables. We adopt the stochastic optimal control point of view, that is, we work with explicit control and state variables in order to deal with an explicit dynamics of the system and to obtain an interpretation of the multipliers associated to the dynamics.

10.1.1. Stochastic optimization problem in discrete time

Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space, where Ω is the set of possible outcomes, $\mathcal{A}$ the associated σ-field and $\mathbb{P}$ the probability measure. We denote by $\{0, .., T\}$ the discrete time span $\{0, 1, \dots, T\}$, and we define upon it three processes $\boldsymbol{X} = \left\{\boldsymbol{X}_t\right\}_{t\in\{0,..,T\}}$, $\boldsymbol{U} = \left\{\boldsymbol{U}_t\right\}_{t\in\{1,..,T-1\}}$ and $\boldsymbol{W} = \left\{\boldsymbol{W}_t\right\}_{t\in\{1,..,T\}}$ where for all t, $\boldsymbol{X}_t : \Omega \to \mathbb{R}^{n_x}$, $\boldsymbol{U}_t : \Omega \to \mathbb{R}^{n_u}$ and $\boldsymbol{W}_t : \Omega \to \mathbb{R}^{n_\xi}$ are random variables representing respectively the state, the control and the noise variables. The state process $\boldsymbol{X}$ is assumed to follow the linear dynamics

$$\begin{aligned} &\boldsymbol{X}_0 = x_0 \ , \\ &\boldsymbol{X}_{t+1} = A_t\boldsymbol{X}_t + B_{t+1}\boldsymbol{U}_{t+1} + C_{t+1}\boldsymbol{\xi}_{t+1} \qquad \forall t \in \{0, .., T-1\} \ , \end{aligned}$$

where x_0 is the given initial state at time 0 and where $A_t \in \mathbb{R}^{n_x\times n_x}$, $B_{t+1} \in \mathbb{R}^{n_x\times n_u}$ and $C_{t+1} \in \mathbb{R}^{n_x\times n_\xi}$ are given deterministic matrices. We moreover assume that both the control and the state variables are subject to bound constraints, that is, for all $t \in \{0, .., T-1\}$, $\underline{u}_{t+1} \le \boldsymbol{U}_{t+1} \le \overline{u}_{t+1}$, and $\underline{x}_{t+1} \le \boldsymbol{X}_{t+1} \le \overline{x}_{t+1}$ and satisfy a linear coupling constraint

$$D_t\boldsymbol{X}_t + E_{t+1}\boldsymbol{U}_{t+1} + G_{t+1}\boldsymbol{\xi}_{t+1} \le 0 \qquad \forall t \in \{0, .., T-1\} \ .$$

where $D_t \in \mathbb{R}^{n_c\times n_x}$, $E_{t+1} \in \mathbb{R}^{n_c\times n_u}$ and $G_{t+1} \in \mathbb{R}^{n_c\times n_\xi}$ are given matrices. In particular, state and control variables take values in compact subsets of $\mathbb{R}^{n_x}$ and $\mathbb{R}^{n_u}$ respectively.

We assume that the problem has a Hazard-Decision[1] information structure, that is, the decision at time t is taken knowing the noise that affects the system between t and $t+1$. Accordingly, the decision $\boldsymbol{U}_{t+1}$ is a function of the uncertainties up to time $t+1$, which means that $\boldsymbol{U}_{t+1}$ has to be measurable with respect to the σ-field $\mathcal{F}_{t+1}$ generated by the uncertainties $(\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_{t+1})$. We write this *non anticipativity constraint* as, $\boldsymbol{U}_{t+1} \preceq \mathcal{F}_{t+1}$, for all $t \in \{0, .., T-1\}$.

Finally, the cost incurred at each time $t \in \{0, .., T-1\}$ is a linear function $a_t^\top\boldsymbol{X}_t + b_{t+1}^\top\boldsymbol{U}_{t+1}$ with $a_t \in \mathbb{R}^{n_x}$ and $b_{t+1} \in \mathbb{R}^{n_u}$, and the cost incurred at the final time T is $K(\boldsymbol{X}_T)$ where K is polyhedral, hence a convex lower semi-continuous function. Note that the results obtained in this paper for a polyhedral final cost function can be adapted to the case where K is a convex lower semi-continuous function, Lipschitz-continuous on its domain.

[1] Wait-and-see in the Stochastic Programming terminology

Gathering all these elements, we get the following stochastic optimization problem:

$$\min_{\boldsymbol{X},\boldsymbol{U}} \mathbb{E}\left[\sum_{t=0}^{T-1}(a_t^\top \boldsymbol{X}_t + b_{t+1}^\top \boldsymbol{U}_{t+1}) + K(\boldsymbol{X}_T)\right], \tag{10.1a}$$

$$\text{s.t. } \boldsymbol{X}_0 = x_0\,, \tag{10.1b}$$

$$\boldsymbol{X}_{t+1} = A_t\boldsymbol{X}_t + B_{t+1}\boldsymbol{U}_{t+1} + C_{t+1}\boldsymbol{\xi}_{t+1} \qquad \forall t \in \{0,..,T-1\}\,, \tag{10.1c}$$

$$\underline{u}_{t+1} \leq \boldsymbol{U}_{t+1} \leq \overline{u}_{t+1} \qquad \forall t \in \{0,..,T-1\}\,, \tag{10.1d}$$

$$\underline{x}_{t+1} \leq \boldsymbol{X}_{t+1} \leq \overline{x}_{t+1} \qquad \forall t \in \{0,..,T-1\}\,, \tag{10.1e}$$

$$D_t\boldsymbol{X}_t + E_{t+1}\boldsymbol{U}_{t+1} + G_{t+1}\boldsymbol{\xi}_{t+1} \leq 0 \qquad \forall t \in \{0,..,T-1\}\,, \tag{10.1f}$$

$$\boldsymbol{U}_{t+1} \preceq \mathcal{F}_{t+1} \qquad \forall t \in \{0,..,T-1\}\,. \tag{10.1g}$$

We make the following assumption throughout the paper.

Assumption 10.1.1 (discrete white noise)**.** *The noise sequence* $\left\{\boldsymbol{\xi}_t\right\}_{t\in\{1,..,T\}}$ *is assumed to be a sequence of independent variables with finite support.*

As it is well known, independence is of paramount importance to obtain Dynamic Programming equation, while finiteness of the support is required both to be able to compute exactly the expectation and for theoretical convergence reasons.

10.1.2. Stochastic Dual Dynamic Programming and its weaknesses

Thanks to white noise Assumption 10.1.1, we can solve Problem (10.1) by the Dynamic Programming approach (see the two reference books Bellman (1957) and Bertsekas (2005a) for further details). This approach leads to the so-called Bellman's value functions V_t, such that $V_t(x)$ is the optimal value of the problem when starting at time t with state $\boldsymbol{X}_t = x$. These functions are obtained by solving the following recursive Bellman equation:

$$\begin{aligned}
V_T(x) =& K(x)\,,\\
V_t(x) =& \mathbb{E}\Big[\inf_{u_{t+1},x_{t+1}} a_t^\top x + b_{t+1}^\top u_{t+1} + V_{t+1}(x_{t+1})\Big]\,,\\
&\text{s.t.}\quad x_{t+1} = A_t x + B_{t+1}u_{t+1} + C_{t+1}\xi_{t+1}\,,\\
&\qquad \underline{u}_{t+1} \leq u_{t+1} \leq \overline{u}_{t+1}\,,\\
&\qquad \underline{x}_{t+1} \leq x_{t+1} \leq \overline{x}_{t+1}\,,\\
&\qquad D_t x + E_{t+1}u_{t+1} + G_{t+1}\xi_{t+1} \leq 0\,.
\end{aligned}$$

When the state variable takes a finite number of possible values, we can solve the Bellman equation by exhaustive exploration of the state, yielding the exact solution of the problem. In the continuous linear-convex case, we can rely on polyhedral approximations. The Stochastic Dual Dynamic Programming algorithm (SDDP) builds polyhedral approximations $\underline{V}_t$ of the functions V_t by using a sampled nested Benders decomposition (see Philpott and Guan (2008), Girardeau et al. (2014), Guigues (2016) and Guigues (2017) for the convergence of this approach). The polyhedral value functions $\underline{V}_t$ computed by SDDP are *outer approximations* of the functions V_t at each stage, that is, $\underline{V}_t \leq V_t$, so that the value $\underline{v}_0 = \underline{V}_0(x_0)$ is an exact (deterministic) lower bound on the optimal value $v_0 = V_0(x_0)$ of Problem (10.1).

Functions $\underline{V}_t$ can also be used to derive an admissible strategy, whose associated expected cost $\overline{v}_0$ gives an upper bound of the optimization problem value. Unfortunately, computing the expectation is usually out of reach, and we need to rely on some approximate computation. The most common way to perform that task is based on the Monte Carlo approach: it consists in simulating the control

strategy induced by functions $\underline{V}_t$ along a (large) number M of noise scenarios, and then computing the arithmetic mean $\widehat{v}_0^M$ of the scenarios cost and the associated empirical standard deviation $\widehat{\sigma}_0^M$. The value $\widehat{v}_0^M$ is an approximate (statistical) upper bound of the optimal value of Problem (10.1). Moreover, it is easy to obtain an asymptotic α-confidence interval $[\widehat{v}_0^M - z_\alpha \widehat{\sigma}_0^M, \widehat{v}_0^M + z_\alpha \widehat{\sigma}_0^M]$. Here $1-\alpha \in [0,1]$ is a chosen confidence level and $z_\alpha = \Phi^{-1}(1-\alpha)$, Φ being the cumulative distribution function of the standard normal distribution.

The classical way to use this statistical upper bound in SDDP, as presented in Pereira and Pinto (1991), consists in testing at each iteration of the algorithm if the available exact lower bound $\underline{v}_0$ is greater than the α-confidence lower bound $\widehat{v}_0^M - z_\alpha \widehat{\sigma}_0^M$, and to stop the algorithm in that case. Such a stopping criterion raises at least two difficulties: the Monte Carlo simulation increases the computational burden of SDDP, and the stopping test does not give any guarantee of convergence of the algorithm.

The first difficulty can be tackled by parallelizing the M simulations involved in the evaluation of the upper bound, and also by calculating the empirical mean $\widehat{v}_0$ over the last $\overline{k}$ iterations of the algorithm, thus enlarging the sample size from M to $\overline{k}M$ without additional computation (see (Shapiro et al., 2012, §3.2)). The second difficulty induced by this stopping criterion has been analyzed in Shapiro (2011): the larger the standard deviation $\widehat{\sigma}_0$ and the confidence $(1-\alpha)$ are, the sooner the algorithm will be stopped. The author proposes another criterion based on the difference between the α-confidence upper bound $\widehat{v}_0 + z_\alpha \widehat{\sigma}_0$ and the exact lower bound $\underline{v}_0$ up to a prescribed accuracy level ε. Note that this stopping test is not necessarily convergent, in the sense that the stopping criterion might not be met in finite time, for example if $\varepsilon < z_\alpha \widehat{\sigma}_0$. An interesting view on the class of stopping criteria in terms of statistical hypothesis tests has been given in Homem-de Mello et al. (2011): the authors compare different hypothesis tests of optimality[2] and so they find the stopping criteria proposed by Pereira and Pinto (1991), Shapiro (2011), as well as another one which ensures an upper bound on the probability of incorrectly claiming convergence (type II error). Moreover, the simulation scenarios are obtained using Quasi-Monte Carlo or Latin Hypercube Sampling rather than raw Monte carlo, so that the accuracy of the upper bound is increased. Nevertheless, all these stopping criteria are based on a statistical evaluation and thus give a probabilistic guarantee that the gap is smaller than some ε, not an almost sure one.

A different approach consists in building polyhedral inner approximations $\overline{V}_t$ of the Bellman functions V_t at each stage, that is, $\overline{V}_t \geq V_t$. A deterministic upper bound $\overline{V}_0(x_0)$ of the optimal value of Problem (10.1) thus becomes available, and it is then possible to perform a stopping test of SDDP on the almost-sure gap $\overline{V}_0(x_0) - \underline{V}_0(x_0)$. Such a test, giving a guarantee on the algorithm convergence has been investigated in Philpott et al. (2013). More precisely, starting from a polyhedral inner approximation $\overline{V}_{t+1}$ at time $t+1$, and choosing an arbitrary sequence of points $x_t^1, \ldots, x_t^{J_t}$, the authors show how to compute a value q_t^j at each point x_t^j such that $q_t^j \geq V_t(x_t^j)$. The inner polyhedral approximation $\overline{V}_t$ is then obtained from the pairs $\{(x_t^j, q_t^j)\}_{j \in \{1,..,J_t\}}$. A delicate issue when devising the inner approximations is the choice of the points x_t^j defining the polyhedral functions $\overline{V}_t$. The authors suggest to use points generated by some other algorithm, such as SDDP. Another approach involving inner and outer approximations of the Bellman functions is described in Baucke et al. (2017), whose main feature is that the algorithm presented herein is fully deterministic.

10.1.3. Contents of the paper

In Section 10.2, we introduce the formalism of linear Bellman operators for a large class of optimization problem, we define its dual linear Bellman operator and we enlighten the relationships between them thanks to the Fenchel conjugate. We also present the SDDP algorithm that applies to a sequence of function recursively defined through linear Bellman operators. We apply in Section 10.3 the conjugacy results obtained in Section 10.2 to obtain a recursion on the dual value

[2] such as: (H_0: $\overline{v}_0 = \underline{v}_0$) *against* ($H_1$: $\overline{v}_0 \neq \underline{v}_0$)

functions, on which we can apply the abstract SDDP algorithm, yielding a dual SDDP algorithm for solving Problem (10.1). The main result of this section is that we eventually obtain an exact upper bound over the value of Problem (10.1). In Section 10.4, we show how to build inner approximations of Bellman functions associated to Problem (10.1) thanks to outer approximations computed by the dual SDDP algorithm. This inner-approximation induces a control strategy converging toward an optimal one (see Theorem 10.4.3). Furthermore, the expected cost incurred by this strategy is shown to be lower than the exact upper bound obtained in Section 10.3. Ultimately, in Section 10.5, we illustrate all the presented methodology on an energy management problem inspired by Électricité de France, at the European scale. The results show, on the one hand that having at disposal an exact upper bound in SDDP allows to devise a more efficient stopping test for SDDP than the usual ones based on a Monte Carlo approach, and on the other hand that the strategy based on the inner approximation of the Bellman functions outperforms the usual strategy obtained using standard outer approximations.

10.1.4. Notations

- $\{i, .., j\}$ denotes the set of integer between i and j.
- Ω denotes a finite set of cardinality $|\Omega|$ supporting a probability distribution $\mathbb{P}$: $\Omega = \{\omega_1, \ldots, \omega_{|\Omega|}\}$ and $\mathbb{P} = (\pi_1, \ldots, \pi_{|\Omega|})$.
- Random variables are denoted using bold uppercase letters (such as $\boldsymbol{Z} : \Omega \to \mathbb{Z}$), and their realizations are denoted using lowercase letters ($z \in \mathbb{Z}$).
- $\boldsymbol{X} : \Omega \to \mathbb{X}$ corresponds to the state, $\boldsymbol{U} : \Omega \to \mathbb{U}$ to the control, $\boldsymbol{\xi} : \Omega \to \Xi$ to the noise.
- $[R]^\star$ denotes the Fenchel transform of an extended real-valued function R.
- $V_t : \mathbb{X}_t \to \mathbb{R}$ is the Bellman value function associated to Problem (10.1) at time t.
- $\mathcal{D}_t = [V_t]^\star$ is the dual value function associated to Problem (10.1) at time t.
- $\mathcal{B}$ is the Bellman operator associated to a generic linear problem, with associated solution operator $\mathcal{S}$, and dual operator denoted $\mathcal{B}^\ddagger$.
- $\mathcal{T}$ is the Bellman operator associated to Problem (10.1), its dual being denoted $\mathcal{T}^\ddagger$.
- Underlined notation (e.g. $\underline{V}$) corresponds to a lower approximation of a function (e.g V). Overlined notation (e.g. $\overline{V}$) denotes an upper approximation.

10.2. Linear Bellman operators

This self-contained section is devoted to the definition and properties of linear Bellman operators (LBOs). In §10.2.1 we present the abstract formalism of LBOs that allows to write Dynamic Programming equations in a compact manner. In §10.2.2 we show that the Fenchel transform of a LBO is also a LBO. In §10.2.3 we present an abstract version of the SDDP algorithm adapted to the LBO formulation.

10.2.1. Linear Bellman operator

We first introduce the notion of *linear Bellman operator*, which is a particular class of Bellman operators (see Bertsekas (2005a)) associated to stochastic optimal control problems where costs and constraints are linear.

We consider an abstract probability space $(\Omega, \mathcal{A}, \mathbb{P})$. Recall that Ω is a finite set (see Assumption 10.1.1) and assume that the σ-field $\mathcal{A}$ is generated by all the singletons of Ω. We denote by $\mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ the set of functions defined on $\mathbb{R}^{n_x}$ and taking values in $[-\infty, +\infty]$, and by $\mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x})$ the space of $\mathbb{R}^{n_x}$-valued measurable functions defined on $(\Omega, \mathcal{A}, \mathbb{P})$.

Remark 10.2.1. *Since we suppose that the set Ω is finite, every function defined on Ω is measurable and a property that holds almost surely is a property that holds for every $\omega \in \Omega$. In particular $\mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^n)$ is a finite-dimensional space whathever $n \in \mathbb{N}$.* $\lozenge$

Definition 10.2.2. *An operator $\mathcal{B} : \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}}) \to \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ is said to be a* linear Bellman operator *(LBO) if it is defined as follows*

$$\mathcal{B} : \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}}) \to \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$$

$$R \mapsto \mathcal{B}(R) : x \mapsto \inf_{(\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) \times \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u})} \mathbb{E}\Big[\boldsymbol{C}^\top \boldsymbol{U} + R(\boldsymbol{Y})\Big] \,, \tag{10.2a}$$

$$s.t. \quad Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) \leq \boldsymbol{H} \,, \tag{10.2b}$$

where $\mathcal{W}^u : \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u}) \to \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_c})$ and $\mathcal{W}_y : \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) \to \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_c})$ are two linear *operators. Here, $\boldsymbol{U}$ and $\boldsymbol{Y}$ are two decision random variables respectively defined on $\mathbb{R}^{n_u}$ and $\mathbb{R}^{n_x}$. The two random variables $\boldsymbol{C} : \Omega \to \mathbb{R}^{n_u}$ and $\boldsymbol{H} : \Omega \to \mathbb{R}^{n_c}$ are exogenous uncertainties in Problem* (10.2) *and we note $\boldsymbol{W} = (\boldsymbol{C}, \boldsymbol{H})$. Deterministic matrix $T \in \mathbb{R}^{n_c \times n_x}$ is given data.*

We denote by $\mathcal{S}(R)$ the set valued mapping giving, for a given $x \in \mathbb{R}^{n_x}$, the set of optimal solutions $\boldsymbol{Y}$ of Problem (10.2)*:*

$$\mathcal{S}(R) : \mathbb{R}^{n_x} \to \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) \tag{10.3a}$$

$$x \mapsto \operatorname*{arg\,min}_{\boldsymbol{Y} \in \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x})} \left(\inf_{\boldsymbol{U} \in \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u})} \mathbb{E}\Big[\boldsymbol{C}^\top \boldsymbol{U} + R(\boldsymbol{Y})\Big] \right) , \tag{10.3b}$$

$$s.t. \quad Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) \leq \boldsymbol{H} \,. \tag{10.3c}$$

Let $\mathcal{G} : \mathbb{R}^{n_x} \rightrightarrows \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_u}) \times \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x})$ be the set valued mapping defined by

$$\mathcal{G}(x) := \big\{ (\boldsymbol{U}, \boldsymbol{Y}) \mid Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) \leq \boldsymbol{H} \big\} \,. \tag{10.4a}$$

With domain $\mathrm{dom}(\mathcal{G}) = \big\{ x \in \mathbb{R}^{n_x} \mid \mathcal{G}(x) \neq \emptyset \big\}$*. Further, we say that $\mathcal{B}$ is* compact *if $\mathcal{G}$ is compact-valued with non-empty compact domain.*

Note that, using the definition of the set valued mapping $\mathcal{G}$, we have that

$$\mathcal{B}(R)(x) = \inf_{(\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) \times \mathbb{R}^{n_u})} \mathbb{E}\Big[\boldsymbol{C}^\top \boldsymbol{U} + \chi_{\mathcal{G}(x)}(\boldsymbol{U}, \boldsymbol{Y})\Big] + \mathbb{E}\Big[R(\boldsymbol{Y})\Big] \,,$$

where χ_A is the characteristic function of a set A:

$$\chi_A(x) = \begin{cases} 0 & \text{if } x \in A, \\ +\infty & \text{otherwise.} \end{cases} \tag{10.5}$$

Example. *We give some classical examples of operators $\mathcal{W}^u$ and $\mathcal{W}_y$ involved in Definition 10.2.2 of $\mathcal{B}$. We stress that $\mathcal{W}$ is a linear operator over a space of random variables, and we describe the associated adjoint operator, that is, the linear operator $\mathcal{W}^\dagger$ such that $\big\langle \boldsymbol{X} \,, \mathcal{W}(\boldsymbol{Y}) \big\rangle = \big\langle \mathcal{W}^\dagger(\boldsymbol{X}) \,, \boldsymbol{Y} \big\rangle$, with $\big\langle \boldsymbol{X} \,, \boldsymbol{Y} \big\rangle = \mathbb{E}\big[\boldsymbol{X}^\top \boldsymbol{Y}\big]$.*

- *Linear point-wise operator:*

$$\begin{array}{rcccc} \mathcal{W} & : & \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) & \to & \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_c}) \\ & & \big(\omega \mapsto \boldsymbol{Y}(\omega)\big) & \mapsto & \big(\omega \mapsto A\boldsymbol{Y}(\omega)\big) \,. \end{array}$$

Such an operator allows to encode an almost sure constraint*, and* $\mathcal{W}^\dagger(\boldsymbol{X}) = A^\top \boldsymbol{X}$.

- *Linear expected operator:*

$$\begin{array}{rccc} \mathcal{W} & : & \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) & \to & \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_c}) \\ & & \big(\omega \mapsto \boldsymbol{Y}(\omega)\big) & \mapsto & \big(\omega \mapsto A\,\mathbb{E}(\boldsymbol{Y})\big)\ . \end{array}$$

Such an operator allows to encode a constraint in expectation*, and* $\mathcal{W}^\dagger(\boldsymbol{X}) = A^\top \mathbb{E}(\boldsymbol{X})$.

- *Linear conditional operator: given a sub* σ*-field* $\mathcal{F}$ *of* $\mathcal{A}$*,*

$$\begin{array}{rccc} \mathcal{W} & : & \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_x}) & \to & \mathcal{L}^0(\Omega, \mathcal{A}; \mathbb{R}^{n_c}) \\ & & \big(\omega \mapsto \boldsymbol{Y}(\omega)\big) & \mapsto & \big(\omega \mapsto A\,\mathbb{E}[\boldsymbol{Y}\,|\mathcal{F}](\omega)\big)\ , \end{array}$$

Such an operator allows to encode, for example, measurability constraints and $\mathcal{W}^\dagger(\boldsymbol{X}) = A^\top \mathbb{E}[\boldsymbol{X}\,|\mathcal{F}]$.

Of course, any linear combination of these three kinds of operator is also linear. △

We define the key notion of *relatively complete recourse*, introduced in Rockafellar and Wets (1976).

Definition 10.2.3. *Let* $Q \in \mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ *and* $\mathcal{B}$ *a LBO. We say that the pair* $(\mathcal{B}, Q)$ *satisfies a* relatively complete recourse *(RCR) assumption if* $\mathrm{dom}(\mathcal{B}(Q)) = \mathrm{dom}(\mathcal{G})$*, that is, if*

$$\forall x \in \mathrm{dom}(\mathcal{G})\ ,\ \ \exists (\boldsymbol{U}, \boldsymbol{Y}) \in \mathcal{G}(x) \ \text{ such that } \ \mathbb{P}\big(\boldsymbol{Y} \in \mathrm{dom}(Q)\big) = 1\ . \tag{10.6}$$

Remark 10.2.4. *Note that if* $\mathcal{B}$ *is compact and* $(\mathcal{B}, Q)$ *satisfy the RCR assumption, then* $\mathcal{B}(Q)$ *is finite at some point* x_0. ♢

We now turn to properties of LBOs and polyhedral functions, proven in Appendix 10.6.2.

Proposition 10.2.5. *Let* R *be a function of* $\mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ *and let* $\mathcal{B}$ *be a LBO. Then we have the following properties.*

1. *If* R *is convex, then* $\mathcal{B}(R)$ *is convex.*
2. *If* R *is polyhedral, then* $\mathcal{B}(R)$ *is polyhedral.*
3. *If* $R \geq \tilde{R}$*, then* $\mathcal{B}(R) \geq \mathcal{B}(\tilde{R})$.

Proof. The probability set Ω being finite, we denote by u (resp. y, c, h) the vectors concatenating all possible values of $\boldsymbol{U}$ (resp. $\boldsymbol{Y}$, $\boldsymbol{C}$, $\boldsymbol{H}$) over the set Ω, that is $u = (u_1, \ldots, u_{|\Omega|})$. Then the extensive formulation of Constraint (10.2b) is

$$\widetilde{T}x + \widetilde{W}_u u + \widetilde{W}_y y \leq h\ ,$$

where $\widetilde{T}$, $\widetilde{W}_u$ and $\widetilde{W}_y$ are adequate matrices deduced from T, $\mathcal{W}^u$ and $\mathcal{W}_y$. Problem (10.2) rewrites

$$\mathcal{B}(R)(x) = \inf_{u,y}\ J(R)(x, u, y)\ ,$$

with

$$J(R)(x, u, y) = \sum_{\omega=1}^{|\Omega|} \pi_\omega \Big(c_\omega^\top u_\omega + R(y_\omega)\Big) + \chi_{\{\widetilde{T}x + \widetilde{W}_u u + \widetilde{W}_y y \leq h\}}(x, u, y)\ .$$

1. If R is convex, then $J(R)$ is jointly convex in (x, u, y) so that $\mathcal{B}(R)$ is a convex function.

2. If R is polyhedral, then $J(R)$ is polyhedral in (x, u, y), and thus $\mathcal{B}(R)$ is a polyhedral function (see (Borwein and Lewis, 2010, Prop 5.1.8)).

3. From $R \geq \tilde{R}$, we deduce that $J(R) \geq J(\tilde{R})$, and thus $\mathcal{B}(R) \geq \mathcal{B}(\tilde{R})$.

The proof is complete. □

Remark 10.2.6. *Assume that function R is proper and polyhedral. Then, under relatively complete recourse (see Definition 10.2.3), and if $\mathcal{B}(R)$ is finite at some point, $\mathcal{B}(R)$ is a proper polyhedral function. Furthermore, if $\mathcal{B}(R)(x)$ is finite, solving its (linear programming) dual generates a supporting hyperplane at point x of the function $\mathcal{B}(R)$, that is, a pair $(\lambda, \beta) \in \mathbb{R}^{n_x} \times \mathbb{R}$ such that*

$$\begin{cases} \langle \lambda \, , \cdot \rangle + \beta \leq \mathcal{B}(R)(\cdot) \\ \langle \lambda \, , x \rangle + \beta = \mathcal{B}(R)(x) \, . \end{cases}$$

Such hyperplanes, or cuts, are of paramount importance for the SDDP algorithm. ◊

The following proposition establish a link between the Lipschitz constants of R and $\mathcal{B}(R)$.

Proposition 10.2.7. *Let R be a proper polyhedral function of $\mathcal{L}^0(\mathbb{R}^{n_x}; \overline{\mathbb{R}})$ and let $\mathcal{B}$ be a LBO. Assume that $(\mathcal{B}, R)$ satisfies the RCR assumption and that $\mathcal{B}(R)$ is finite at some point. If R is Lipschitz (for the L_1-norm) with constant L_R, then $\mathcal{B}(R)$ is also Lipschitz on its domain (which is $\mathrm{dom}(\mathcal{G})$) with constant $\phi(L_R) = (\|\boldsymbol{C}\|_\infty + L_R)\kappa_W \|T\|_\infty$, where κ_W is a constant associated to the linear operator $(\mathcal{W}^u, \mathcal{W}^y)$.*

Proof. Consider x_1 (resp. x_2) an element in $\mathrm{dom}(\mathcal{B}(R))$, and denote by Z_1 (resp. Z_2) the polyhedron of optimal solutions of Problem (10.2). These polyhedrons are non-empty thanks to the RCR assumption. Let $(\boldsymbol{U}_1, \boldsymbol{Y}_1) \in Z_1$ be fixed, from (Shapiro et al., 2009, Theorem 7.12), there exists a positive constant κ_W such that

$$\inf_{(\boldsymbol{U}_2, \boldsymbol{Y}_2) \in Z_2} \|(\boldsymbol{U}_1, \boldsymbol{Y}_1) - (\boldsymbol{U}_2, \boldsymbol{Y}_2)\|_1 \leq \kappa_W \|T(x_1 - x_2)\|_1 \leq \kappa_W \|T\|_\infty \|x_1 - x_2\|_1 \, .$$

Let $(\boldsymbol{U}_2^\sharp, \boldsymbol{Y}_2^\sharp) \in Z_2$ be an optimal solution of the above minimization problem. Then we have that

$$\begin{aligned} \mathcal{B}(R)(x_1) &= \mathbb{E}\Big[\boldsymbol{C}^\top \boldsymbol{U}_1 + R(\boldsymbol{Y}_1)\Big] \\ &\leq \mathbb{E}\Big[\boldsymbol{C}^\top \boldsymbol{U}_2^\sharp + \boldsymbol{C}^\top (\boldsymbol{U}_1 - \boldsymbol{U}_2^\sharp) + R(\boldsymbol{Y}_2^\sharp) + L_R \|\boldsymbol{Y}_2^\sharp - \boldsymbol{Y}_1\|_1)\Big] \\ &\leq \mathcal{B}(R)(x_2) + \|\boldsymbol{C}\|_\infty \|\boldsymbol{U}_1 - \boldsymbol{U}_2^\sharp\|_1 + L_R \|\boldsymbol{Y}_1 - \boldsymbol{Y}_2^\sharp\|_1 \\ &\leq \mathcal{B}(R)(x_2) + (\|\boldsymbol{C}\|_\infty + L_R)\kappa_W \|T\|_\infty \|x_1 - x_2\|_1 \, . \end{aligned}$$

Exchanging x_1 and x_2 in the previous majoration leads to the reverse inequality, combining both leads to the desired Lipschitz property. □

10.2.2. Fenchel transform of a LBO

We now define the *dual linear Bellman operator* $\mathcal{B}^\ddagger$ of a linear Bellman operator $\mathcal{B}$.

Definition 10.2.8. *Let $\mathcal{B}$ be a LBO (see Definition (10.2.2)). We denote by $\mathcal{B}^\ddagger$ the dual LBO of*

$\mathcal{B}$, defined, for a given function $Q \in \mathcal{L}^0(\mathbb{R}^{n_x};\overline{\mathbb{R}})$ and for any $\lambda \in \mathbb{R}^{n_x}$, by

$$\mathcal{B}^\ddagger(Q)(\lambda) = \inf_{\boldsymbol{\mu}\in\mathcal{L}^0(\Omega,\mathcal{A};\mathbb{R}^{n_x}),\boldsymbol{\nu}\in\mathcal{L}^0(\Omega,\mathcal{A};\mathbb{R}^{n_c})} \quad \mathbb{E}\Big[-\boldsymbol{\mu}^\top \boldsymbol{H} + Q(\boldsymbol{\nu})\Big] \tag{10.7a}$$

$$s.t. \quad T^\top \mathbb{E}\big[\boldsymbol{\mu}\big] + \lambda = 0 \tag{10.7b}$$

$$\mathcal{W}_u^\dagger(\boldsymbol{\mu}) = \boldsymbol{C} \tag{10.7c}$$

$$\mathcal{W}_y^\dagger(\boldsymbol{\mu}) = \boldsymbol{\nu} \tag{10.7d}$$

$$\boldsymbol{\mu} \leq 0\,, \tag{10.7e}$$

where $\mathcal{W}_u^\dagger$ (resp. $\mathcal{W}_y^\dagger$) is the adjoint operator of $\mathcal{W}^u$ (resp. $\mathcal{W}_y$). We define the dual constraint set valued mapping

$$\mathcal{G}^\ddagger(\lambda) = \big\{(\boldsymbol{\mu},\boldsymbol{\nu}) \in \mathbb{R}^{n_x+n_c} \mid T^\top\mathbb{E}\big[\boldsymbol{\mu}\big] + \lambda = 0\,,\ \mathcal{W}_u^\dagger(\boldsymbol{\mu}) = \boldsymbol{C}\,,\ \mathcal{W}_y^\dagger(\boldsymbol{\mu}) = \boldsymbol{\nu}\,,\ \boldsymbol{\mu} \leq 0\big\}\,. \tag{10.8}$$

Note that straightforward computations show that $(\mathcal{B}^\ddagger)^\ddagger = \mathcal{B}$.

Calling $\mathcal{B}^\ddagger$ the dual of $\mathcal{B}$ is justified by the following Theorem.

Theorem 10.2.9. *Let R be a proper polyhedral function, $\mathcal{B}$ be a compact LBO (see Definition 10.2.2), such that the pair $(\mathcal{B}, R)$ satisfies the RCR assumption. Then $\mathcal{B}(R)$ is a proper polyhedral function and we have*

$$[\mathcal{B}(R)]^\star = \mathcal{B}^\ddagger\big([R]^\star\big) \quad . \tag{10.9}$$

Proof. First note that as $\mathcal{B}$ is compact, $\mathcal{G}$ has non-empty compact domain, and thus $\mathcal{B}(R)$ is finite at some point by the RCR assumption.

During the proof we denote $\big\langle \boldsymbol{X}\,, \boldsymbol{Y}\big\rangle = \mathbb{E}\big(\boldsymbol{X}^\top\boldsymbol{Y}\big)$, $\mathcal{R}(\boldsymbol{Y}) = \mathbb{E}\big(R(\boldsymbol{Y})\big)$, and

$$\mathcal{K}(x,\boldsymbol{Y}) = \min_{\boldsymbol{U}} \quad \big\{ \big\langle \boldsymbol{C}\,, \boldsymbol{U}\big\rangle \quad \mid \quad Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) \leq \boldsymbol{H}\big\}\,.$$

By definition, we have $\mathcal{B}(R)(x) = \inf_{\boldsymbol{Y}} \mathcal{K}(x,\boldsymbol{Y}) + \mathcal{R}(\boldsymbol{Y})$. Thus, for any $x^\star \in \mathbb{R}^{n_x}$, we have

$$[\mathcal{B}(R)]^\star(x^\star) = \sup_{x\in\mathbb{R}^{n_x}} \Big\{x^\top x^\star - \inf_{\boldsymbol{Y}} \big\{\mathcal{K}(x,\boldsymbol{Y}) + \mathcal{R}(\boldsymbol{Y})\big\}\Big\} = -\inf_{\boldsymbol{Y}} \Big\{\mathcal{R}(\boldsymbol{Y}) - \underbrace{\sup_{x\in\mathbb{R}^{n_x}} x^\top x^\star - \mathcal{K}(x,\boldsymbol{Y})}_{:=\Phi(\boldsymbol{Y})}\Big\}$$

As R is a proper polyhedral function, so is $\mathcal{R}$. By construction, $\mathcal{K}$ is polyhedral. Since $\mathcal{B}$ is a compact LBO, the minimization in $\boldsymbol{U}$ is over a compact, so that function $\mathcal{K}$ is never equal to $-\infty$. Furthermore, $\mathcal{K}$ is proper as $\mathrm{dom}(\mathcal{B}(R)) = \mathrm{dom}(\mathcal{G}) \neq \emptyset$. Note that for $x \notin \mathrm{dom}(\mathcal{G})$ we have $\mathcal{K}(x,\boldsymbol{Y}) = +\infty$, thus $\Phi(\boldsymbol{Y}) = \sup_{x\in\mathrm{dom}(\mathcal{G})} \big\{x^\top x^\star - \mathcal{K}(x,\boldsymbol{Y})\big\}$. Consequently, as $\mathrm{dom}(\mathcal{G})$ is a compact set, we deduce that $-\Phi$ is a proper polyhedral function.Finally, the RCR assumption ensures that $\mathrm{dom}(-\Phi)\cap\mathrm{dom}(\mathcal{R}) \neq \emptyset$. Now, using Fenchel duality (see Proposition 10.6.1) we have that

$$[\mathcal{B}(R)]^\star(x^\star) = -\sup_{\boldsymbol{Y}^\star} \Phi_\star(\boldsymbol{Y}^\star) - \mathcal{R}^\star(\boldsymbol{Y}^\star) = \inf_{\boldsymbol{Y}^\star} \mathcal{R}^\star(\boldsymbol{Y}^\star) - \Phi_\star(\boldsymbol{Y}^\star)$$

where $\mathcal{R}^\star(\boldsymbol{Y}^\star) = \mathbb{E}\big(R^\star(\boldsymbol{Y}^\star)\big)$ and

$$\begin{aligned}
\Phi_\star(\boldsymbol{Y}^\star) &= \inf_{\boldsymbol{Y}} \left\langle \boldsymbol{Y}^\star \,, \boldsymbol{Y} \right\rangle - \Phi(\boldsymbol{Y}) \\
&= \inf_{\boldsymbol{Y}} \left\langle \boldsymbol{Y}^\star \,, \boldsymbol{Y} \right\rangle - \sup_{x} \left\{ x^\top x^\star - \mathcal{K}(x, \boldsymbol{Y}) \right\} \\
&= \inf_{x, \boldsymbol{Y}, \boldsymbol{U}} \left\langle \boldsymbol{Y}^\star \,, \boldsymbol{Y} \right\rangle - x^\top x^\star + \left\langle \boldsymbol{C} \,, \boldsymbol{U} \right\rangle \\
&\quad + \sup_{\boldsymbol{\lambda} \geq 0} \Big\{ \left\langle \boldsymbol{\lambda} \,, Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) - \boldsymbol{H} \right\rangle \quad \Big| \quad Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) \leq \boldsymbol{H} \Big\}
\end{aligned}$$

As $\mathrm{dom}(\mathcal{G}) \neq \emptyset$, there exists a primal feasible solution to the above linear program, and by duality we have

$$\begin{aligned}
\Phi_\star(\boldsymbol{Y}^\star) = \sup_{\boldsymbol{\lambda} \geq 0} - \left\langle \boldsymbol{\lambda} \,, \boldsymbol{H} \right\rangle + \inf_{x} \left\{ - x^\top x^\star + \left\langle T^\top \boldsymbol{\lambda} \,, x \right\rangle \right\} + \inf_{\boldsymbol{Y}} \left\{ \left\langle \boldsymbol{Y}^\star \,, \boldsymbol{Y} \right\rangle + \left\langle \mathcal{W}_y^\dagger(\boldsymbol{\lambda}) \,, \boldsymbol{Y} \right\rangle \right\} \\
+ \inf_{\boldsymbol{U}} \left\{ \left\langle \boldsymbol{C} \,, \boldsymbol{U} \right\rangle + \left\langle \mathcal{W}_u^\dagger(\boldsymbol{\lambda}) \,, \boldsymbol{U} \right\rangle \right\} \\
= \sup_{\boldsymbol{\lambda} \geq 0} \Big\{ - \left\langle \boldsymbol{\lambda} \,, \boldsymbol{H} \right\rangle \quad | \quad \mathcal{W}_y^\dagger(\boldsymbol{\lambda}) = -\boldsymbol{Y}^\star, \quad \mathcal{W}_u^\dagger(\boldsymbol{\lambda}) = -\boldsymbol{C}, \quad T^\top \mathbb{E}(\boldsymbol{\lambda}) = x^\star \Big\}
\end{aligned}$$

Finally,

$$[\mathcal{B}(R)]^\star(x^\star) = \inf_{\boldsymbol{Y}^\star, \boldsymbol{\lambda} \geq 0} \Big\{ \mathbb{E}\big(\boldsymbol{H}^\top \boldsymbol{\lambda} + R^\star(\boldsymbol{Y}^\star)\big) \quad \Big| \quad T^\top \mathbb{E}\big(\boldsymbol{\lambda}\big) = x^\star, \ \mathcal{W}_y^\dagger(\boldsymbol{\lambda}) = -\boldsymbol{Y}^\star, \ \mathcal{W}_u^\dagger(\boldsymbol{\lambda}) = -\boldsymbol{C} \Big\}$$

which is equivalent to (10.7) with the correspondence $x^\star \to \lambda$, $\boldsymbol{\lambda} \to \boldsymbol{\mu}$ and $\boldsymbol{Y}^\star \to \boldsymbol{\nu}$. □

10.2.3. An abstract SDDP algorithm

We now consider a sequence of functions $\big\{R_t\big\}_{t \in \{0,..,T\}}$ that follows the Bellman recursion

$$\begin{cases} R_T = K \\ R_t = \mathcal{B}_t(R_{t+1}) \quad \forall t \in \{0, .., T-1\} \,, \end{cases} \tag{10.10}$$

where K is a proper polyhedral function, and a sequence of LBOs $\big\{\mathcal{B}_t\big\}_{t \in \{0,..,T-1\}}$ given by

$$\begin{aligned}
\mathcal{B}_t(R)(x) = \inf_{\boldsymbol{U}, \boldsymbol{Y}} \quad & \mathbb{E}\Big[\boldsymbol{C}_t^\top \boldsymbol{U} + R(\boldsymbol{Y})\Big] \\
\text{s.t.} \quad & T_t x + \mathcal{W}_t^u(\boldsymbol{U}) + \mathcal{W}_t^y(\boldsymbol{Y}) \leq \boldsymbol{H}_t \,,
\end{aligned}$$

with associated set valued mappings $\big\{\mathcal{G}_t\big\}_{t \in \{0,..,T-1\}}$:

$$\mathcal{G}_t(x) := \big\{ (\boldsymbol{U}, \boldsymbol{Y}) \mid T_t x + \mathcal{W}_t^u(\boldsymbol{U}) + \mathcal{W}_t^y(\boldsymbol{Y}) \leq \boldsymbol{H}_t \big\} \,,$$

and associated solution operators $\big\{\mathcal{S}_t\big\}_{t \in \{0,..,T-1\}}$:

$$\begin{aligned}
\mathcal{S}_t(R)(x) = \arg\min_{\boldsymbol{Y}} \ \inf_{\boldsymbol{U}} \quad & \mathbb{E}\Big[\boldsymbol{C}_t^\top \boldsymbol{U} + R(\boldsymbol{Y})\Big] \\
\text{s.t.} \quad & T_t x + \mathcal{W}_t^u(\boldsymbol{U}) + \mathcal{W}_t^y(\boldsymbol{Y}) \leq \boldsymbol{H}_t \,.
\end{aligned}$$

We now give an extension of the RCR Definition 10.2.3.

Definition 10.2.10. *Let* $\{\mathcal{B}_t\}_{t\in\{0,..,T-1\}}$ *be a sequence of LBOs, with* $\mathrm{dom}(\mathcal{G}_t) \neq \emptyset$*, and* $\{R_t\}_{t\in\{0,..,T\}}$ *be defined by the Bellman recursion* (10.10)*. We say that the sequence* $\{\mathcal{B}_t\}_{t\in\{0,..,T-1\}}$ *is* K-compatible *if,*

$$\forall t \in \{0,..,T-1\}, \quad \forall x \in \mathrm{dom}(\mathcal{G}_t), \quad \forall (\boldsymbol{U}_{t+1}, \boldsymbol{Y}_{t+1}) \in \mathcal{G}_t(x), \qquad \mathbb{P}\big(\boldsymbol{Y}_{t+1} \in \mathrm{dom}(\mathcal{G}_{t+1})\big) = 1 \;, \tag{10.11}$$

where by convention $\mathrm{dom}(\mathcal{G}_T) = \mathrm{dom}(K)$*.*

Remark 10.2.11. *The relatively complete assumption (see Definition 10.2.3) applied to a sequence of pairs* $\{(\mathcal{B}_t, R_{t+1})\}_{t\in\{0,..,T-1\}}$ *would be asking that, for all* $t \in \{0,..,T-1\}$*, the pair* $(\mathcal{B}_t, R_{t+1})$ *satisfies the RCR assumption, that is,*

$$\forall t \in \{0,..,T-1\}, \quad \forall x \in \mathrm{dom}(\mathcal{G}_t), \quad \exists (\boldsymbol{U}_{t+1}, \boldsymbol{Y}_{t+1}) \in \mathcal{G}_t(x), \qquad \mathbb{P}\big(\boldsymbol{Y}_{t+1} \in \mathrm{dom}(\mathcal{G}_{t+1})\big) = 1 \;. \tag{10.12}$$

At first glance, assuming K*-compatibility of LBOs seems to be stronger than this assumption. Indeed, we require that* every *admissible control leads to an admissible future state, instead of only assuming the existence of at least one such control.*

In fact, under the RCR assumption, the constraint $\boldsymbol{Y}_{t+1} \in \mathrm{dom}(\mathcal{G}_{t+1})$ $\mathbb{P}$*-a.s. is implicit in the definition of* $\mathcal{B}_t(R_{t+1})$*: a control* $(\boldsymbol{U}_{t+1}, \boldsymbol{Y}_{t+1})$ *such that* $\boldsymbol{Y}_{t+1} \notin \mathrm{dom}(\mathcal{G}_{t+1})$ *leads to an infinite value of* $\mathcal{B}_t(R_{t+1})(x)$ *since* $\boldsymbol{Y}_{t+1} \notin \mathrm{dom}(R_{t+1})$*.*

More precisely, consider a sequence of LBOs $\{\mathcal{B}^e_t\}_{t\in\{0,..,T-1\}}$*, with associated operators* $\{\mathcal{G}^e_t\}_{t\in\{0,..,T\}}$ *satisfying* (10.12)*, and define*

$$\begin{cases} R_T = K \\ R_t = \mathcal{B}^e_t(R_{t+1}) \quad \forall t \in \{0,..,T-1\} \;. \end{cases}$$

Then, we can define a new sequence of LBOs $\{\mathcal{B}_t\}_{t\in\{0,..,T-1\}}$*, where* $\mathcal{B}_t$ *is equal to* $\mathcal{B}^e_t$ *with the additional constraint that* $\boldsymbol{Y}_{t+1} \in \mathrm{dom}(\mathcal{G}^e_{t+1})$ $\mathbb{P}$*-a.s.. In this case we can easily see that the sequence* $\{\mathcal{B}_t\}_{t\in\{0,..,T-1\}}$ *is* K*-compatible, and that* $\{R_t\}_{t\in\{0,..,T\}}$ *follows the Bellman recursion* (10.10)*.*

Finally, note that constructing $\mathcal{B}_t$ *from* $\mathcal{B}^e_t$ *does not require multistage constraint propagation. Without loss of generality, we will assume* K*-compatibility instead of* (10.12)*. This is useful to ensure that all states generated in the forward pass of the SDDP algorithm are admissible states.* ♢

SDDP is an algorithm that iteratively constructs finite lower polyhedral approximations of the sequence of functions $\{R_t\}_{t\in\{0,..,T-1\}}$ given by Equation (10.10). Starting from an initial point $x_0 \in \mathbb{R}^{n_x}$, the algorithm determines in a forward pass a sequence of states $(x^k_t)_{t\in\{0,..,T\}}$ at which the approximations of the sequence $\{R_t\}_{t\in\{0,..,T-1\}}$ will be refined in the backward pass. More precisely, a pseudo-code describing the abstract SDDP algorithm is given in Algorithm 10.1.

Lemma 10.2.12. *Assume that* $R_0(x_0)$ *is finite and that* $(\mathcal{B}_t)_{t\in\{0,..,T-1\}}$ *is a* K*-compatible sequence of LBO. Then, the (abstract) SDDP Algorithm 10.1 is well defined and there exists a sequence* $(L_t)_{t\in\{0,..,T-1\}}$ *such that* R_t *is* L_t*-Lipschitz on its domain, and* $\|\lambda^{(k)}_t\|_\infty \leq L_t$*.*

Proof. We prove by induction that the points $x^{(k)}_t$ is well defined during the forward passes of SDDP. Let $t = 0$. By assumption, $x^k_0 \in \mathrm{dom}(\mathcal{G}_0)$. So $x^k_1 = \boldsymbol{X}^{(k)}_1(\omega^k)$ exists as a solution of a finite valued LP. Let $t \geq 1$. By the induction hypothesis, we suppose that x^k_t is well defined and belongs to $\mathrm{dom}(\mathcal{G}_t)$. We set $x^k_{t+1} = \boldsymbol{X}^{(k)}_{t+1}(\omega^k)$, which is well defined as a solution of a finite value LP. By assumption the sequence $\{\mathcal{B}_t\}_{t\in\{0,..,T-1\}}$ is K-compatible, hence $x^k_{t+1} \in \mathrm{dom}(\mathcal{G}_{t+1})$, thus proving the result at time $t+1$.

Algorithm 10.1: Abstract SDDP algorithm

Data: Initial point x_0
Set $\underline{R}_t^{(0)} \equiv -\infty$
for $k = 0, 1, \dots$ **do**
  // Forward Pass : compute a set of trial points $\left\{x_t^k\right\}_{t\in\{0,..,T\}}$
  Randomly select $\omega^k \in \Omega$;
  Set $x_0^k = x_0$;
  for $t : 0 \to T-1$ **do**
    select $\boldsymbol{X}_{t+1}^k \in \mathcal{S}_t(\underline{R}_{t+1}^k)\big(x_t^k\big)$; // see Definition 10.2.2
    set $x_{t+1}^k = \boldsymbol{X}_{t+1}^k(\omega^k)$;
  end
  // Backward Pass : refine the lower-approximations at the trial points
  Set $\underline{R}_T^{k+1} = K$;
  for $t : T-1 \to 0$ **do**
    $\theta_t^{k+1} = \mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k)$; // cut coefficients (see Remark 10.2.6)
    $\lambda_t^{k+1} \in \partial\mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k)$;
    $\beta_t^{k+1} := \theta_t^{k+1} - \left\langle \lambda_t^{k+1}, \overline{x}_t^k \right\rangle$;
    set $\mathcal{C}_t^{k+1} : x \mapsto \left\langle \lambda_t^{k+1}, x \right\rangle + \beta_t^{k+1}$; // new cut
    $\underline{R}_t^{k+1} := \max\left\{\underline{R}_t^k, \mathcal{C}_t^{k+1}\right\}$; // update lower approximation
  end
  If some stopping test is satisfied STOP ;
end

We now prove by backward induction that $\lambda_t^{(k)}$ is well defined during the backward passes of SDDP, and that there exists L_t such that $\|\lambda_t^{(k)}\|_\infty \leq L_t$. As K is a given L_T Lipschitz-continuous function, the property holds true for $t = T$. Let $t \leq T-1$. Assume that induction hypothesis holds for $t+1$. Then, by proposition 10.2.7, we know that $\mathcal{B}_t(\underline{R}_{t+1}^{k+1})$ is L_t-Lipschitz. We set $\lambda_t^{k+1} \in \partial\mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k)$, which is well defined as subgradient of a finite-valued polyhedral function. As $\mathcal{B}_t(\underline{R}_{t+1}^{k+1})$ is L_t-Lipschitz on its domain, we are able to choose λ_t^{k+1} such that $\|\lambda_t^{(k+1)}\|_\infty \leq L_t$. □

From Lemma 10.2.12, proven in Appendix 10.6.2, we have the boundness of $\lambda_t^{(k)}$, from which we can easily adapt the proof of Girardeau et al. (2014), to obtain the following convergence result.

Proposition 10.2.13. *Assume that $R_0(x_0)$ is finite and that $(\mathcal{B}_t)_{t\in\{0,..,T-1\}}$ is a K-compatible sequence of LBO. Further assume that, for all $t \in \{0, .., T\}$ there exists compact sets X_t such that, for all k, $x_t^k \in X_t$. In particular this is the case if $\mathcal{B}_t$ is compact for all t.*

Then, the abstract SDDP algorithm generates a non-decreasing sequence $(\underline{R}_t^{(k)})_{k\in\mathbb{N}}$ of lower approximation of R_t, and $\lim_k \underline{R}_0^{(k)}(x_0) = R_0(x_0)$.

This algorithm is abstract in the sense that it only requires a sequence of LBOs. In the following section we show how it can be applied to approximate the Bellman value functions $\left\{V_t\right\}_{t\in\{0,..,T\}}$, or to approximate the Fenchel transform of these functions.

10.3. Primal and dual SDDP

In this section we recall the usual SDDP algorithm applied to Problem (10.1). Next, leveraging the results of Section 10.2, we introduce a dual SDDP algorithm, which is the abstract SDDP

algorithm applied to the dual value functions. This eventually gives an exact upper bound over the value of Problem (10.1). In this section, we denote by V_t the primal value functions, and by $\mathcal{D}_t = [V_t]^\star$ the dual value functions.

10.3.1. Primal SDDP

10.3.1.1. Primal Dynamic Programming equations

Thanks to the discrete white noise Assumption 10.1.1, we can solve Problem (10.1) through Dynamic Programming, computing backward the value functions $\{V_t\}_{t\in\{0,..,T\}}$ given by

$$\begin{cases} V_T = K + \chi_{\underline{x}_T \leq \cdot \leq \overline{x}_T} \ , \\ V_t = \mathcal{T}_t^e\big(V_{t+1}\big) \ , \end{cases} \tag{10.13}$$

where the primal Bellman operator $\mathcal{T}_t^e : \mathcal{L}^0(\mathbb{R}^{n_x};\overline{\mathbb{R}}) \to \mathcal{L}^0(\mathbb{R}^{n_x};\overline{\mathbb{R}})$ is defined as follows:

$$\begin{aligned} \mathcal{T}_t^e(R) : x \mapsto \inf_{\boldsymbol{U}_{t+1},\boldsymbol{X}_{t+1}} \quad & \mathbb{E}\Big[a_t^\top x + b_{t+1}^\top \boldsymbol{U}_{t+1} + R(\boldsymbol{X}_{t+1})\Big] \ , && \text{(10.14a)} \\ \text{s.t.} \quad & \boldsymbol{X}_{t+1} = A_t x + B_{t+1}\boldsymbol{U}_{t+1} + C_{t+1}\boldsymbol{\xi}_{t+1} \ , && \text{(10.14b)} \\ & D_t x + E_{t+1}\boldsymbol{U}_{t+1} + G_{t+1}\boldsymbol{\xi}_{t+1} \leq 0 \ , && \text{(10.14c)} \\ & \underline{u}_{t+1} \leq \boldsymbol{U}_{t+1} \leq \overline{u}_{t+1} \ , && \text{(10.14d)} \\ & \underline{x}_t \leq x \leq \overline{x}_t \ . && \text{(10.14e)} \end{aligned}$$

Constraint (10.14e) ensures that if x does not satisfies $\underline{x}_t \leq x \leq \overline{x}_t$, then $\mathcal{T}_t^e(R)(x) = +\infty$. By Definition 10.2.2, the operator $\mathcal{T}_t^e$ is a LBO. The associated set valued mapping $\mathcal{G}_t^e$ is defined by

$$\mathcal{G}_t^e(x) := \big\{(\boldsymbol{U}_{t+1}, \boldsymbol{X}_{t+1}) \ , \ \text{s.t. constraints (10.14b)—(10.14e) are satisfied}\big\} \ .$$

We make the following assumptions.

Assumption 10.3.1.

1. *The function K is polyhedral.*
2. *For all $t \in \{0,..,T-1\}$, and all $x \in \mathrm{dom}(\mathcal{G}_t^e)$, there exists $(\boldsymbol{U}_{t+1}, \boldsymbol{X}_{t+1}) \in \mathcal{G}_t^e(x)$, such that $\boldsymbol{X}_{t+1} \in \mathrm{dom}(\mathcal{G}_{t+1}^e)$, where by convention $\mathrm{dom}(\mathcal{G}_T^e) = \mathrm{dom}(K)$.*
3. *Problem (10.1) is finite valued.*

Remark 10.3.2. *Point 2 of Assumption 10.3.1, is equivalent to asking that for all $t \in \{0,..,T-1\}$, the pair $(\mathcal{T}_t^e, V_{t+1})$ follows the RCR assumption as stated in Definition 10.2.3. Note that this assumption can be checked t by t, that is, without requiring backward constraint propagation.* ◊

Following Remark 10.2.11, we define a more constrained sequence of primal LBOs $\{\mathcal{T}_t\}_{t\in\{0,..,T-1\}}$ by adding in each Problem (10.14) the constraint

$$\boldsymbol{X}_{t+1} \in \mathrm{dom}(\mathcal{G}_{t+1}^e) \ , \tag{10.14f}$$

so that the following Bellman recursion holds true:

$$\begin{cases} V_T = K \ , \\ V_t = \mathcal{T}_t\big(V_{t+1}\big) \ . \end{cases} \tag{10.15}$$

The set valued mapping $\mathcal{G}_t$ associated to $\mathcal{T}_t$ is thus defined by

$$\mathcal{G}_t(x) := \left\{(\boldsymbol{U}_{t+1}, \boldsymbol{X}_{t+1}) \text{ , s.t. constraints (10.14b)—(10.14f) are satisfied}\right\} , \tag{10.16}$$

and is compact valued with compact domain.

Remark 10.3.3. *For the sake of notational simplicity, we assume from now on that constraints* (10.14d)-(10.14e)-(10.14f) *are embedded in Constraint* (10.14c) *(see an example of such an embedding in Appendix 10.6.5).* ♢

Lemma 10.3.4. *Under Assumption 10.3.1, the sequence* $\left\{\mathcal{T}_t\right\}_{t\in\{0,..,T-1\}}$ *is a K-compatible sequence of compact LBOs. Further, for any* $t \in \{0, .., T-1\}$*, we have*

$$\mathcal{T}_t(R) : x \mapsto \mathbb{E}\big(\widehat{\mathcal{T}}_t(R)(x, \boldsymbol{\xi}_{t+1})\big) , \tag{10.17}$$

where

$$\widehat{\mathcal{T}}_t(R) : (x, \xi) \mapsto \inf_{u_{t+1}, x_{t+1}} \quad a_t^\top x + b_{t+1}^\top u_{t+1} + R(x_{t+1}) , \tag{10.18a}$$

$$s.t. \quad x_{t+1} = A_t x + B_{t+1} u_{t+1} + C_{t+1}\xi , \tag{10.18b}$$

$$D_t x + E_{t+1} u_{t+1} + G_{t+1}\xi \leq 0 , \tag{10.18c}$$

where we used the change of notation given in Remark 10.3.3.

Proof. As Problem (10.1) is finite valued, the domain of each set valued mapping $\mathcal{G}_t$ is non empty. Further, as $\mathcal{G}_t$ is compact valued with compact domain, each LBO $\mathcal{T}_t$ is compact (see Definition 10.2.2). The K-compatibility of $\left\{\mathcal{T}_t\right\}_{t\in\{0,..,T-1\}}$ is a direct consequence of Remark 10.2.11.

Then, the reformulation as Equations (10.17) and (10.18) is the direct consequence of the measurability properties of the pair $(\boldsymbol{U}_{t+1}, \boldsymbol{X}_{t+1})$ allowing the interchange between minimization and expectation. □

To recover the optimal state and control trajectories from Bellman functions, we introduce the set valued mappings:

$$\widehat{\mathcal{S}}_t(R) : (x, \xi) \mapsto \quad \operatorname*{arg\,min}_{x_{t+1}} \quad \inf_{u_{t+1}} \quad a_t^\top x + b_{t+1}^\top u_{t+1} + R(x_{t+1}) , \tag{10.19a}$$

$$\text{s.t.} \quad x_{t+1} = A_t x + B_{t+1} u_{t+1} + C_{t+1}\xi \tag{10.19b}$$

$$D_t x + E_{t+1} u_{t+1} + G_{t+1}\xi \leq 0 . \tag{10.19c}$$

10.3.1.2. Primal SDDP

We now apply the abstract SDDP algorithm presented in §10.2.3 to the primal Bellman operator given by Equations (10.14). We denote, for all $t \in \{1, .., T\}$, and $\pi_t^\xi := \mathbb{P}(\boldsymbol{\xi}_t = \xi)$ for all $\xi \in \text{supp}(\boldsymbol{\xi})$. The pseudocode is given in Algorithm 4.

Remark 10.3.5. *Note that, the primal Bellman operator* (10.14) *is a specialized version of the abstract Bellman operator used in Equation* (10.10)*, which only involves pointwise operator in the constraints. Hence, in the forward pass we just have to compute* $\widehat{\mathcal{T}}_t(\underline{V}_{t+1}^k)(x_t^k, \xi_{t+1}^k)$ *and do not need to compute* $\mathcal{T}_t(\underline{V}_{t+1}^k)(x_t^k)$ *which would involve a larger linear problem. Similarly, in the backward pass at time t we solve* $|\text{supp}(\boldsymbol{\xi}_{t+1})|$ *linear problem of the form* $\widehat{\mathcal{T}}_t(\underline{V}_{t+1}^{k+1})(x_t^k, \xi_{t+1}^s)$ *(instead of the larger linear problem* $\mathcal{T}_t(\underline{V}_{t+1}^{k+1})(x_t^k)$*) and then perform an expectation. We will show in the sequel that this is no more the case in the dual SDDP algorithm.* ♢

Algorithm 10.2: Primal SDDP algorithm

Data: Initial point x_0, initial lower bounds $\underline{V}_t^0$ on V_t
for $k = 0, 1, \dots$ **do**
  Draw a noise scenario $\left\{\xi_t^k\right\}_{t\in\{0,..,T\}}$;
  `// Forward Pass : compute a set of trial points` $\left\{x_t^k\right\}_{t\in\{0,..,T\}}$
  Set $x_0^k = x_0$;
  for $t : 0 \to T-1$ **do**
    select $x_{t+1}^k \in \widehat{\mathcal{S}}_t(\underline{V}_{t+1}^k)(x_t^k, \xi_{t+1}^k)$; `// see` (10.19)
  end
  `// Backward Pass : refine the lower-approximations at the trial points`
  Set $\underline{V}_T^{k+1} = K$;
  for $t : T-1 \to 0$ **do**
    for $\xi \in \mathrm{supp}(\boldsymbol{\xi}_{t+1})$ **do**
      solve the linear program $\widehat{\mathcal{T}}_t(\underline{V}_{t+1}^{k+1})(x_t^k, \xi)$;
      yielding $\underline{\theta}_t^{\xi,k+1} := \widehat{\mathcal{T}}_t(\underline{V}_{t+1}^{k+1})(x_t^k, \xi)$ and $\underline{\lambda}_t^{\xi,k+1} \in \partial\widehat{\mathcal{T}}_t(\underline{V}_{t+1}^{k+1})(x_t^k, \xi)$;
    end
    $\underline{\lambda}_t^{k+1} := \sum_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \pi_{t+1}^{\xi}\underline{\lambda}_t^{\xi,k+1}$; `// taking expectation`
    $\underline{\beta}_t^{k+1} := \sum_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \pi_{t+1}^{\xi}(\underline{\theta}_t^{\xi,k+1} - \langle \underline{\lambda}_t^{\xi,k+1}, x_t^k\rangle)$;
    $\underline{V}_t^{k+1} := \max\left\{\underline{V}_t^k(\cdot),\ \langle \underline{\lambda}_t^k, \cdot\rangle + \underline{\beta}_t^{k+1}\right\}$; `// update lower approximation`
  end
  If some stopping test is satisfied STOP ;
end

Proposition 10.3.6. *Under Assumptions 10.1.1 and 10.3.1, the primal SDDP algorithm yields a converging lower bound for the value of Problem* (10.1)*. Further, the strategy induced by* $\underline{V}_t^{(k)}$ *is converging toward an optimal strategy.*

Proof. By Assumption 10.1.1 the sequence of value functions $\{V_t\}_{t\in\{0,..,T\}}$ can be obtained by Dynamic Programming and follows the recursion (10.15). By Lemma 10.3.4, we have the K-compatibility of the sequence of LBOs $\{\mathcal{T}_t\}_{t\in\{0,..,T-1\}}$. Further, as $\mathcal{T}_t$ is a compact LBO for any $t \in \{0,..,T-1\}$, the sequence $\{x_t^k\}_{k\in\mathbb{N}}$ generated by the algorithm remains in a compact set. Hence, we can apply Proposition 10.2.13.

Proof of the convergence of the strategy obtained can be found in Girardeau et al. (2014). □

10.3.2. Dual SDDP

We present here a dual SDDP algorithm, which leverages the conjugacy results of §10.2.2. We show that the Fenchel conjugates of the primal value functions $(V_t)_{t\in\{0,..,T\}}$ follow a recursive equation on which we apply the abstract SDDP algorithm 10.1.

10.3.2.1. Dual Dynamic Programming equations

By Definition 10.2.8, a straightforward computation shows that the dual LBO of $\mathcal{T}_t$, is given by

$$\mathcal{T}_t^\ddagger(Q) : \lambda_t \mapsto \inf_{\boldsymbol{\Lambda}_{t+1}, \boldsymbol{N}_{t+1} \geq 0} \mathbb{E}\big[-(C_{t+1}^\top \boldsymbol{\Lambda}_{t+1} + G_{t+1}^\top \boldsymbol{N}_{t+1})^\top \boldsymbol{\xi}_{t+1} + Q(\boldsymbol{\Lambda}_{t+1})\big] \tag{10.20a}$$

$$\text{s.t.} \quad a_t + A_t^\top \mathbb{E}\big[\boldsymbol{\Lambda}_{t+1}\big] + D_t^\top \mathbb{E}\big[\boldsymbol{N}_{t+1}\big] - \lambda_t = 0 \tag{10.20b}$$

$$b_{t+1} + B_{t+1}^\top \boldsymbol{\Lambda}_{t+1} + E_{t+1}^\top \boldsymbol{N}_{t+1} = 0 \,, \tag{10.20c}$$

where $\boldsymbol{\Lambda}_{t+1} : \Omega \to \mathbb{R}^{n_x}$ and $\boldsymbol{N}_{t+1} : \Omega \to \mathbb{R}^{n_c}$ are two $\boldsymbol{\xi}_{t+1}$-measurable random variables.

In Equation (10.20), the function Q is a cost-to-go at time $t+1$ for the dual linear problem, λ_t is a state variable, and $(\boldsymbol{\Lambda}_{t+1}, \boldsymbol{N}_{t+1})$ are control variables. Equations (10.20b) and (10.20c) define the admissible control set of the problem.

Theorem 10.3.7. *We assume that K is a polyhedral function and that Assumption 10.3.1 holds true. For any $t \in \{0, .., T\}$, we denote $\mathcal{D}_t := V_t^\star$, where V_t are the Bellman value functions obtained by* (10.15)*. Let, for all $t \in \{0, .., T\}$, $L_t > 0$ be such that V_t is L_t-Lipschitz (for the L^1-norm) on its domain. Then the sequence of dual value functions $\big\{\mathcal{D}_t\big\}_{t \in \{0,..,T\}}$ satisfies the following backward recursion:*

$$\mathcal{D}_T = K^\star \,, \tag{10.21a}$$

$$\mathcal{D}_t = \mathcal{T}_{t, L_{t+1}}^\ddagger(\mathcal{D}_{t+1}) \qquad \forall t \in \{0, .., T-1\} \,, \tag{10.21b}$$

where $\mathcal{T}_{t, L_{t+1}}^\ddagger$ is defined by Equation (10.20)*, with the additional constraint $\|\boldsymbol{\Lambda}_{t+1}(\omega)\|_\infty \leq L_{t+1}$.*

Proof. From Lemma 10.3.4, we have that $\big\{\mathcal{T}_t\big\}_{t=\{0,..,T-1\}}$ is a K-compatible sequence of compact LBOs, with the associated sequence $\big\{V_t\big\}_{t=\{0,..,T\}}$ of Bellman functions defined by Equation (10.15). Let $t \in \{0, .., T-1\}$. Consider the L_t-Lipschitz regularization of V_t defined by $V_t^{L_t} := V_t \square (L_t \|\cdot\|_1)$ (see section 10.6.1 for details). The Bellman function V_t is L_t-Lipschitz continuous on its domain (see proposition 10.2.7 for the existence of L_t), so that $V_t^{L_t}$ coincides with V_t on its domain, and is L_t-Lipschitz continuous everywhere. The K-compatibility property implies that what only matters is the restriction of V_{t+1} to $\mathrm{dom}(\mathcal{G}_{t+1})$, and thus $V_t = \mathcal{T}_t\big(V_{t+1}^{L_{t+1}}\big)$. Theorem 10.2.9 applies, so that

$$[V_t]^\star = \mathcal{T}_t^\ddagger\Big(\Big[V_{t+1}^{L_{t+1}}\Big]^\star \Big) \,.$$

As V_{t+1} and $L_{t+1}\|\cdot\|_1$ takes values in $(-\infty, +\infty]$, we have ((Bauschke et al., 2017, Corollary 13.24))

$$\Big[V_{t+1}^{L_{t+1}}\Big]^\star = [V_{t+1}]^\star + \chi_{B_\infty(0, L_{t+1})} \,,$$

where $B_\infty(0, L_{t+1})$ is the L_∞-ball of radius L_{t+1} centered in 0. Thus we have

$$\mathcal{D}_t = \mathcal{T}_t^\ddagger\Big(\mathcal{D}_{t+1} + \chi_{B_\infty(0, L_{t+1})}\Big) \,,$$

which precisely matches the backward recursion (10.21). □

Remark 10.3.8. *Lemma 10.2.12 shows how to find such a sequence of Lipschitz constants $\big\{L_t\big\}_{t \in \{0,..,T\}}$. But in some cases we can directly derive Lipschitz constant on the value functions, and plug it into Equation* (10.21)*.* ♢

10.3.2.2. Dual SDDP

From now on we assume that

Assumption 10.3.9. $\left\{\mathcal{T}^{\ddagger}_{t,L_{t+1}}\right\}_{t\in\{0,..,T-1\}}$ *is* $K^\star$*-compatible.*

This is ensured for example if all A_t in Problem (10.1) are square invertible matrices (see Appendix 10.6.5).

The dual value functions $\left\{\mathcal{D}_t\right\}_{t\in\{0,..,T\}}$ are solutions of a Bellman recursion (Theorem 10.3.7) involving linear Bellman operators $\left\{\mathcal{T}^{\ddagger}_{t,L_{t+1}}\right\}_{t\in\{0,..,T-1\}}$, thus opening the door to the computation of outer approximations $\left\{\underline{\mathcal{D}}^k_t\right\}_{t\in\{0,..,T\}}$ by SDDP, as shown in Algorithm 5.

Algorithm 10.3: Dual SDDP algorithm

Data: Initial primal point x_0, Lipschitz bounds $\left\{L_t\right\}_{t\in\{0,..,T\}}$

for $k = 0, 1, \dots$ **do**

`// Forward Pass : compute a set of trial points` $\left\{\lambda_t^{(k)}\right\}_{t\in\{0,..,T\}}$

Compute

$\lambda_0^k \in \arg\max_{\|\lambda_0\|_\infty \leq L_0} \left\{x_0^\top \lambda_0 - \underline{\mathcal{D}}^k_0(\lambda_0)\right\}$; `// Fenchel transform`

for $t : 0 \to T-1$ **do**

select $\boldsymbol{\lambda}^k_{t+1} \in \arg\min \mathcal{T}^{\ddagger}_{t,L_{t+1}}(\underline{\mathcal{D}}^k_{t+1})(\lambda^k_t)$;

and draw a realization λ^k_{t+1} of $\boldsymbol{\lambda}^k_{t+1}$;

end

`// Backward Pass : refine the lower-approximations at the trial points`

Set $\underline{\mathcal{D}}^k_T = K^\star$. ;

for $t : T-1 \to 0$ **do**

$\overline{\theta}^{k+1}_t := \mathcal{T}^{\ddagger}_{t,L_{t+1}}(\underline{\mathcal{D}}^{k+1}_{t+1})(\lambda^k_t)$; `// computing cut coefficients`

$\overline{x}^{k+1}_t \in \partial\mathcal{T}^{\ddagger}_{t,L_{t+1}}(\underline{\mathcal{D}}^{k+1}_{t+1})(\lambda^k_t)$;

$\overline{\beta}^{k+1}_t := \overline{\theta}^{k+1}_t - \left\langle \lambda^k_t \, , \overline{x}^{k+1}_t \right\rangle$;

$\mathcal{C}^{k+1}_t : \lambda \mapsto \left\langle \overline{x}^{k+1}_t \, , \lambda \right\rangle + \overline{\beta}^{k+1}_t$;

$\underline{\mathcal{D}}^{k+1}_t = \max\left\{\mathcal{D}^k_t, \mathcal{C}^{k+1}_t\right\}$; `// update lower approximation`

end

If some stopping test is satisfied STOP ;

end

Lemma 10.3.10. *For all* $t \in \{0, .., T-1\}$, $(\underline{\mathcal{D}}^k_t)^\star$ *is a decreasing sequence of upper bounds of the primal value function* V_t*:* $(\underline{\mathcal{D}}^k_t)^\star \geq V_t$.

Proof. The sequence of functions $\underline{\mathcal{D}}^k_t$ is obtained by applying SDDP to the dual recursion (10.21), which is an increasing sequence of lower-bounds of the function $\mathcal{D}_t$ by Proposition 10.2.13. By conjugacy property, we obtain a decreasing sequence of functions $(\underline{\mathcal{D}}^k_t)^*$ that are upper bounds of the function $\mathcal{D}^*_t = V_t$. □

We have the following convergence theorem.

Theorem 10.3.11. *Under assumptions 10.1.1, 10.3.1 and 10.3.9,* $(\underline{\mathcal{D}}^k_0)^\star(x_0)$ *is a converging upper bound to the value* $V(x_0)$ *of Problem* (10.1)*, that is* $\lim_k (\underline{\mathcal{D}}^k_0)^\star(x_0) = V_0(x_0)$.

Proof. We add a fictive time step $t = -1$, in order to approximate the Fenchel transform of $\underline{\mathcal{D}}^k_0$ at x_0. More precisely, we define $\mathcal{T}^{\ddagger}_{-1,L_0}$ as follows

$$\mathcal{T}^{\ddagger}_{-1,L_0}(R) := \min_{\lambda_0 : \|\lambda_0\|_\infty \leq L_0} -x_0^\top \lambda_0 + R(\lambda_0) \; .$$

Then Algorithm 5 is the abstract SDDP Algorithm 10.1 applied to the Bellman recursion $\mathcal{D}_t = \mathcal{T}^{\ddagger}_{t,L_{t+1}}(\mathcal{D}_{t+1})$ for $t \in \{-1,..,T-1\}$ and $\mathcal{D}_T = K^\star$. The initial point λ_{-1} is arbitrarily set to the value 0 as $\mathcal{D}_{-1}$ is the constant function.

We check that, by definition of $\mathcal{T}^{\ddagger}_{t,L_t}$, $\|\lambda^k_t\|_\infty \leq L_t$. Further, as V_0 is L_0 Lipschitz for the L_1-norm, the supremum of $x_0^T\lambda - V_0^\star(\lambda)$ is attained for some λ_0 such that $\|\lambda_0\|_\infty \leq L_0$, thus we deduce that $\mathcal{D}_{-1}(0) = -[V]^{\star\star}(x_0) = -V(x_0) \in \mathbb{R}$. Finally, note that $\left\{\mathcal{T}^{\ddagger}_{t,L_{t+1}}\right\}_{t\in\{-1,..,T\}}$ is a $K^\star$-compatible sequence of LBOs.

Lemma 10.3.10 shows that, for any $k \in \mathbb{N}$, $-\underline{\mathcal{D}}^k_{-1}(0) = [\underline{\mathcal{D}}_0(x_0)]^\star$ is an upper bound of $V_0(x_0)$. Finally, the convergence of the abstract SDDP algorithm and the lower semi-continuity of V_0 at x_0 yields the convergence of the upper bound. □

Remark 10.3.12. *Recall that, in order to obtain an upper bound of the optimal value of Problem* (10.1)*, the seminal method consists in computing the expected cost of SDDP's strategy with a Monte-Carlo approach (see discussion in §10.1.2). This approach has two weaknesses: it requires a large number M of forward pass (simulation), and the obtained bound is only an upper bound with (asymptotic) probability α, where the bound increases with α as well. Furthermore, the statistical upper-bound is not converging toward the actual problem value, unless we also increase the number of Monte Carlo simulations along the iterations.*

In contrast to the Monte Carlo method, Theorem 10.3.11 shows that we obtain a converging sequence of exact upper bounds for Problem (10.1)*.* ♢

Remark 10.3.13. *In the forward pass of Algorithm 5, we need to solve $\mathcal{T}^{\ddagger}_{t,L_{t+1}}(\underline{\mathcal{D}}^k_{t+1})(\lambda^k_t)$, which in extended form reads*

$$\left\{\lambda^{k,\xi}_{t+1}\right\}_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \in \underset{\left\{\lambda^{\xi}_{t+1}\right\}_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})}}{\arg\min} \inf_{\nu_{t+1}\geq 0} \sum_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \pi^{\xi}_{t+1}\Big[-\big(C^\top_{t+1}\lambda^{\xi}_{t+1} + G^\top_{t+1}\nu^{\xi}_{t+1}\big)^\top \xi^{\xi}_{t+1} + \underline{\mathcal{D}}^k_{t+1}(\lambda^{\xi}_{t+1})\Big] \tag{10.22a}$$

$$s.t. \quad \sum_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \pi^{\xi}_{t+1}(A_t^\top\lambda^{\xi}_{t+1} + D^\top_{t+1}\nu^{\xi}_{t+1}) = \lambda^k_t \tag{10.22b}$$

$$c_{t+1} + B^\top_{t+1}\lambda^{\xi}_{t+1} + E^\top_{t+1}\nu^{\xi}_{t+1} = 0 \quad \forall\xi\,, \tag{10.22c}$$

$$\|\lambda^{\xi}_{t+1}\|_\infty \leq L_{t+1}\,. \tag{10.22d}$$

Then drawing a random realization of $\boldsymbol{\lambda}^k_{t+1}$ consists in drawing ξ with respect to the law of $\boldsymbol{\xi}_{t+1}$ and selecting $\lambda^{k,\xi}_{t+1}$.

In contrast with primal SDDP algorithm (see Remark 10.3.5), here we need to solve a linear problem coupling all possible outcomes of the random variable $\boldsymbol{\xi}_{t+1}$, both in the forward and in the backward pass. In particular it means that we can also compute cuts during the forward pass, thus rendering the backward pass optional. ♢

10.4. Inner-approximation strategy

In Section 10.3, we detailed how to use the SDDP algorithm to get an outer approximation $\left\{\underline{\mathcal{D}}_t\right\}_{t\in\{0,..,T\}}$ of the dual value functions $\left\{\mathcal{D}_t\right\}_{t\in\{0,..,T\}}$. We now explain how to build an inner approximation of the primal value functions $\left\{V_t\right\}_{t\in\{0,..,T\}}$ using this dual outer approximation. Assume that, $\left\{L_t\right\}_{t\in\{0,..,T-1\}}$ is given by Lemma 10.2.12.

10.4.1. Inner approximation of value functions

Let $\left\{\underline{\mathcal{D}}_t^k\right\}_{t\in\{0,..,T\}}$ be the outer approximation of the dual value functions $\left\{\mathcal{D}_t\right\}_{t\in\{0,..,T\}}$ obtained at iteration k of the dual SDDP algorithm. We denote by $\left\{(\overline{x}_t^\kappa, \overline{\beta}_t^\kappa)\right\}_{\kappa\in\{1,..,k\}}$ the cuts coefficients computed by the dual SDDP algorithm:

$$\underline{\mathcal{D}}_t^k(\lambda) = \min_{\theta} \quad \theta \,, \tag{10.23a}$$

$$\text{s.t.} \quad \theta \geq \left\langle \overline{x}_t^\kappa \,, \lambda \right\rangle + \overline{\beta}_t^\kappa \qquad \forall \kappa \in \{1,..,k\} \,. \tag{10.23b}$$

We define the linear inner approximation $\overline{V}_t^k$ of the primal value functions $\left\{V_t\right\}_{t\in\{0,..,T\}}$ as the Lipschitz regularization of the Fenchel conjugate of the dual outer approximation.

Definition 10.4.1. *We define $\overline{V}_t^k$ by*

$$\overline{V}_t^k = \left[\underline{\mathcal{D}}_t^k\right]^\star \square(L_t\|\cdot\|_1) \qquad \forall t \in \{0,..,T\} \,. \tag{10.24}$$

From proposition 10.6.2 in appendix we have the following properties of $\overline{V}_t^k$.

Proposition 10.4.2. *For all $t \in \{0,..,T-1\}$ we have*

i) *$\overline{V}_t^k \geq V_t$ on X_t.*

ii) *We have*

$$\overline{V}_t^k(x) = \min_{y\in\mathbb{R}^{n_x}, \sigma\in\Delta} L_t\|x-y\|_1 - \sum_{\kappa=1}^k \sigma_\kappa \overline{\beta}_t^\kappa \tag{10.25a}$$

$$s.t. \sum_{\kappa=1}^k \sigma_\kappa \overline{x}_t^\kappa = y \,, \tag{10.25b}$$

where $\Delta = \left\{\sigma \in \mathbb{R}^k \mid \sigma \geq 0 \,,\ \sum_{\kappa=1}^k \sigma_\kappa = 1\right\}$ is the simplex of $\mathbb{R}^k$.

iii) *The inner approximation can be computed by solving*

$$\overline{V}_t^k(x) = \sup_{\lambda,\theta} \quad x^\top\lambda - \theta \tag{10.26a}$$

$$s.t. \quad \theta \geq \left\langle \overline{x}_t^i \,, \lambda \right\rangle + \overline{\beta}_t^\kappa \qquad \forall \kappa \in \{1,..,k\} \tag{10.26b}$$

$$\|\lambda\|_\infty \leq L_t \,. \tag{10.26c}$$

iv) *The Fenchel transform of the inner approximation is given by $\left[\overline{V}_t^k\right]^\star = \underline{\mathcal{D}}_t^k + \chi_{B_\infty(0,L_t)}$.*

Proof. Let X_t be $\mathrm{dom}(\mathcal{G}_t)$.

i) lemma 10.3.10 proves that $\left[\underline{\mathcal{D}}_t^k\right]^\star \geq V_t$ for all $t \in \{0,..,T\}$. Thus $\overline{V}_t^k \geq V_t\square(L_t\|\cdot\|_1)$, which is equal to V_t on X_t as V_t is L_t-Lipschitz on its domain.

ii) Furthermore, the Fenchel conjugate $\left[\underline{\mathcal{D}}_t^k\right]^\star$ reads

$$\left[\underline{\mathcal{D}}_t^k\right]^\star(x) = \sup_{\lambda,\theta} \left\{ x^\top\lambda - \theta \quad \middle| \quad \theta \geq \left\langle \overline{x}_t^i \,, \lambda \right\rangle + \overline{\beta}_t^\kappa \qquad \forall \kappa \in \{1,..,k\} \right\} ,$$

$$\left[\underline{\mathcal{D}}_t^k\right]^\star(x) = \sup_{\lambda,\theta} x^\top\lambda - \theta$$
$$\text{s.t. } \theta \geq \langle \overline{x}_t^i, \lambda\rangle + \overline{\beta}_t^\kappa \qquad \forall \kappa \in \{1,..,k\},$$

which is a linear program admitting an admissible solution, hence by strong duality

$$\left[\underline{\mathcal{D}}_t^k\right]^\star(x) = \min_{\sigma\in\Delta}\left\{ -\sum_{\kappa=1}^k \sigma_\kappa \overline{\beta}_t^\kappa \quad | \quad \sum_{\kappa=1}^k \sigma_\kappa \overline{x}_t^\kappa = x \right\}.$$

Taking the inf-convolution with $L_t\|\cdot\|$ yields Problem (10.25).

iii) The right hand side of Equation (10.26) is simply $\left[\underline{\mathcal{D}}_t^k + \chi_{B_\infty(0,L_t)}\right]^\star(x)$, which is equal to $\left[\underline{\mathcal{D}}_t^k\right]^\star \square \left[\chi_{B_\infty(0,L_t)}\right]^\star(x)$ by finite polyhedrality, hence the result.

iv) Finally, $\left[\overline{V}_t^k\right]^\star = \left[\underline{\mathcal{D}}_t^k\right]^{\star\star} + \chi_{B_\infty(0,L_t)}$.

□

Figure 10.4.1 illustrates how to interpret the dual outer approximation as a primal inner approximation of the original value function (in black). The slopes x_1, x_2, x_3 computed for the dual outer approximation (blue curve, right) are breakpoints for the primal problem and we can consider the value of the dual value function at these points to build a primal inner approximation (blue curve, left).

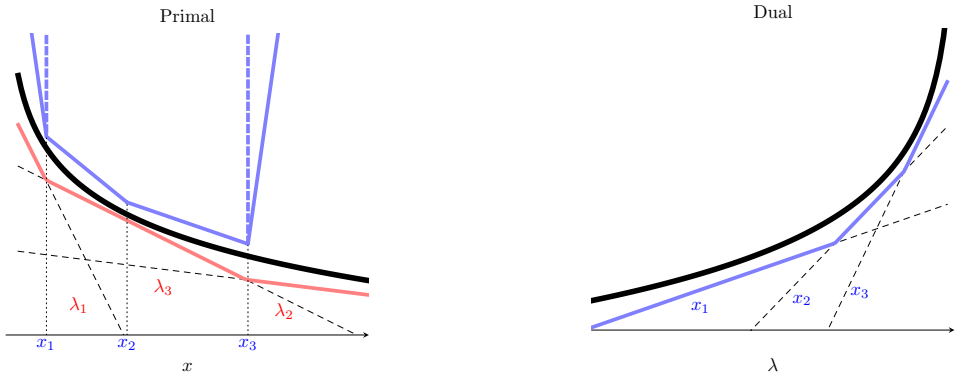


Figure 10.1.: Primal SDDP computes an outer approximation (in red) of the original value function (in black). Dual SDDP computes an outer approximation in the dual, whose (regularized) Fenchel-transform (in blue) yields an inner approximation of the primal problem.

10.4.2. A bound on the inner approximation strategy value

Hence, we have obtained inner approximations of the primal value functions. Such approximations can be used to define an admissible strategy for the initial problem. We now study the properties of such a strategy, proven in Appendix 10.6.2.

Theorem 10.4.3. *Let* $\left\{\boldsymbol{X}_t^{IA}, \boldsymbol{U}_t^{IA}\right\}_{t\in\{0,..,T-1\}}$ *be the state and control processes obtained by applying the strategy induced by the inner approximation* $\left\{\overline{V}_t^k\right\}_{t\in\{0,..,T\}}$*, that is,* $(\boldsymbol{X}_{t+1}^{IA}, \boldsymbol{U}_{t+1}^{IA}) \in$

$\mathcal{S}_t\big(\overline{V}_{t+1}^k\big)(\boldsymbol{X}_t^{IA})$. *Consider the expected cost of this strategy when starting from state x at time t:*

$$C_t^{IA}(x) = \mathbb{E}\Big(\sum_{\tau=t}^{T-1} a_\tau^\top \boldsymbol{X}_\tau^{IA} + b_{\tau+1}^\top \boldsymbol{U}_{\tau+1}^{IA} + K(\boldsymbol{X}_T^{IA}) \;\Big|\; \boldsymbol{X}_t^{IA} = x \Big) .$$

Then,

$$C_t^{IA}(x) \leq \overline{V}_t^k(x) . \tag{10.27}$$

Further, the inner approximation strategy is converging in the sense that $\lim_{k\to+\infty} C_0^{IA,k}(x_0) = V_0(x_0)$.

Proof. We proceed by backward induction on time t. The property holds for $t = T$.

Assume that $C_{t+1}^{IA} \leq \overline{V}_{t+1}^k$. We have

$$\begin{aligned}
C_t^{IA}(x) &= \mathbb{E}\big[a_t^\top x + b_{t+1}^\top \boldsymbol{U}_{t+1}^{IA} + C_{t+1}^{IA}(\boldsymbol{X}_{t+1}^{IA})\big] \\
&\leq \mathbb{E}\big[a_t^\top x + b_{t+1}^\top \boldsymbol{U}_{t+1}^{IA} + \overline{V}_{t+1}^k(\boldsymbol{X}_{t+1}^{IA})\big] && \text{by induction} \\
&= \mathcal{T}_t\big(\overline{V}_{t+1}^k\big)(x) && \text{by definition of } \boldsymbol{U}_{t+1}^{IA} \\
&\leq \overline{V}_t^k(x) && \text{by Lemma 10.6.4}
\end{aligned}$$

hence the result.

Finally, the convergence of the strategy is easily obtained. By definition of $V_0(x_0)$, we have $C_0^{IA,k}(x_0) \geq V_0(x_0)$. Furthermore, $V_0(x_0) \leq C_0^{IA,k}(x_0) \leq \overline{V}_0^k(x_0)$. By Theorem 10.3.11, we know that $\lim_k(\overline{V}_0^k)(x_0) = V_0(x_0)$. Hence the result. □

Remark 10.4.4. *A similar result on the performance of an inner approximation is given in Philpott et al. (2013). As already explained in §10.1.2, the authors construct polyhedral inner approximations $\overline{V}_t$ of the Bellman functions V_t. They then prove that the expected cost of the policy based on the functions $\overline{V}_t$ is always less than or equal to the deterministic upper bound given by the inner approximation algorithm.* ♢

We sum up the available inequalities for the values obtained when implementing the primal and dual SDDP algorithms.

$$\underline{V}_0(x_0) \;\leq\; V_0(x_0) \;\leq\; \overline{V}_0(x_0) , \tag{10.28a}$$

$$\underline{V}_0(x_0) \;\leq\; C_0^{IA}(x_0) \;\leq\; \overline{V}_0(x_0) , \tag{10.28b}$$

$$\underline{V}_0(x_0) \;\leq\; C_0^{OA}(x_0) . \tag{10.28c}$$

Equation (10.28a) corresponds to the deterministic bounds of the optimal value of Problem (10.1), whereas Equations (10.28b) and (10.28c) are of statistical nature.

10.5. Numerical results

In this section, we present some numerical results applying dual SDDP and inner strategy evaluation to a stochastic operation planning problem inspired by Électricité de France (EDF, main European electricity producer). The problem is about the energy production planning on a multi-period horizon including a network of production zones, like in the European Market for electricity. It results in a large-scale stochastic multi-stage optimization problem, for which we need to determine strategies for the management of the European water dams. Such strategies cannot be

computed via Dynamic Programming because of the state variable size, so that SDDP is the reference method to compute the optimal Bellman functions.

10.5.1. Description of the problem

We consider an operation planning problem at the European scale. Different countries are connected together via a network, and exchange energy with their neighbors. We formulate the problem on a graph, where each country is modeled as a node and each interconnection line between two countries as an edge (see Figure 10.2). Every country uses a reservoir to store energy,

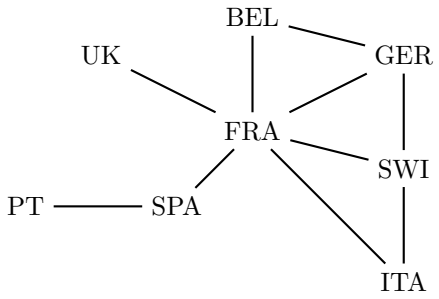


Figure 10.2.: Schematic description of the European network

and must fulfill its own energy demand. To do so, it can produce energy from its reservoir, with its local thermal power plant, or it can import energy from the other countries. A very similar problem has already been studied by Mahey et al. (2017). Its formulation is close to the one given in Shapiro et al. (2012) concerning the Brazilian interconnected power system.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ the graph modeling the European network. The number of nodes in $\mathcal{N}$ is denoted by n and the number of edges in $\mathcal{E}$ by ℓ. For each node $i \in \{1, .., n\}$, we denote by v_t^i the energy stored in the reservoir at time t. The reservoir's dynamics is given by

$$v_{t+1}^i = v_t^i + a_{t+1}^i - q_{t+1}^i - s_{t+1}^i \ , \tag{10.29}$$

where a_{t+1}^i is the (random) water inflow in the reservoir and q_{t+1}^i is the water turbinated between time t and $t+1$ in order to produce electricity. We add a spillage s_{t+1}^i as recourse variable to avoid to overthrow the reservoir. Still at node i, the *load balance* equation at stage t writes

$$q_t^i + g_t^i + \sum_{j \in N_i} f_t^{ji} + r_t^i = d_t^i \ , \tag{10.30}$$

where g_t^i is the thermal production, $N_i \subset \mathcal{N}$ is the set of nodes connected to node i and f_t^{ji} is the energy exchanged between nodes $j \in N_i$ and node i, d_t^i is the (random) demand of the node and $r_t^i \geq 0$ is a recourse variable added to ensure that the load balance is always satisfied. The thermal production g_t^i and the exchanges f_t^{ji} between node i and nodes $j \in N_i$ induce linear costs, and the cost of the recourse variable r_t^i is taken into account through a linear penalization. Hence the total cost attached to node i at time t writes

$$c_t^i g_t^i + \delta_t^i r_t^i + \sum_{j \in N_i} p_t^{ji} f_t^{ji} \ , \tag{10.31}$$

where c_t^i is the thermal price, δ_t^i is the recourse price and p_t^{ji} is the transportation price between nodes j and i. To avoid empty stocks at the end of the time horizon, we penalize the final stock

at each node i if it is beyond a threshold v_0^i using a piecewise linear function:

$$K^i(v_T^i) = \kappa_T^i \max(0, v_0^i - v_T^i) \ . \tag{10.32}$$

Stocks and controls are bounded:

- $0 \leq v_t^i \leq \overline{v}^i$, reservoir volume upper bound,
- $0 \leq q_t^i \leq \overline{q}^i$, reservoir generation upper bound,
- $0 \leq g_t^i \leq \overline{g}^i$, thermal generation upper bound,
- $0 \leq r_t^i$, recourse control lower bound,
- $\underline{f}^{ji} \leq f_t^{ji} \leq \overline{f}^{ji}$, energy flow lower and upper bound.

This problem is formulated as a stochastic optimal control problem, where for all t,

- the state is $v_t = (v_t^1, \cdots, v_t^n)$ (denoted x_t in §10.3),
- the control is $u_t = (q_t, s_t, g_t, r_t, f_t)$, with $q_t = (q_t^i)_{i\in\{1,..,n\}}$ and likewise for s_t, g_t, r_t and f_t,
- the uncertainty is $\xi_t = (a_t^i, d_t^i)_{i\in\{1,..,n\}}$.

The state has dimension n, the control u_t dimension $4n + \ell$ and the uncertainty ξ_t dimension $2n$. We assume that the random variables $\boldsymbol{\xi}_t$ have a discrete finite support. For a given realization (a_{t+1}, d_{t+1}) of the uncertainty, the primal Bellman operator $\widehat{\mathcal{T}}_t$, defined by (10.18), writes

$$\widehat{\mathcal{T}}_t(V_{t+1})\big(v_t, (a_{t+1}, d_{t+1})\big) = \min_{v_{t+1}, q_{t+1}, s_{t+1}, g_{t+1}, r_{t+1}, f_{t+1}} \sum_{i=1}^{n} (c_{t+1}^i g_{t+1}^i + \delta_{t+1}^i r_{t+1}^i + \sum_{j\in N_i} p_{t+1}^{ji} f_{t+1}^{ji}) + V_{t+1}(v_{t+1}) \ , \tag{10.33a}$$

$$\text{s.t.} \quad v_{t+1}^i = v_t^i - q_{t+1}^i - s_{t+1}^i + a_{t+1}^i \ , \tag{10.33b}$$

$$q_{t+1}^i + g_{t+1}^i + \sum_{j\in N_i} f_{t+1}^{ji} + r_{t+1}^i = d_{t+1}^i \ , \tag{10.33c}$$

$$0 \leq v_{t+1}^i \leq \overline{v}^i \ , \ \ 0 \leq q_{t+1}^i \leq \overline{q}^i \ , \ \ 0 \leq r_{t+1}^i \ , \tag{10.33d}$$

$$0 \leq g_{t+1}^i \leq \overline{g}^i \ , \ \ \underline{f}^{ji} \leq f_{t+1}^{ji} \leq \overline{f}^{ji} \ . \tag{10.33e}$$

We rewrite Problem (10.33) in the standard form (10.1) with matrices (A, B, C) for the dynamics and (D, E, G) for the constraints. We note that A is the identity matrix I_n. The expressions of these matrices are given in §10.6.3. Then the expression (10.20) of the dual Bellman operator $\mathcal{T}_t^\ddagger$ is obtained in a straightforward manner, namely

$$\mathcal{T}_{t,L_{t+1}}^\ddagger(\mathcal{D}_{t+1})(\lambda_t) = \inf_{\lambda_{t+1}^\xi, \nu_{t+1}^\xi \geq 0} \sum_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \pi_\xi \big[-(\xi_{t+1}^\xi)^\top C^\top \lambda_{t+1}^\xi - g_{t+1}^\top \nu_{t+1}^\xi + \mathcal{D}_{t+1}(\lambda_{t+1}^\xi) \big] \tag{10.34a}$$

$$\text{s.t.} \quad \sum_{\xi\in\mathrm{supp}(\boldsymbol{\xi}_{t+1})} \pi_\xi (\lambda_{t+1}^\xi + D^\top \nu_{t+1}^\xi) = \lambda_t \tag{10.34b}$$

$$\bar{c}_{t+1} + B^\top \lambda_{t+1}^\xi + E^\top \nu_{t+1}^\xi = 0 \quad \forall \xi \ , \tag{10.34c}$$

$$\left\| \lambda_{t+1}^\xi \right\|_\infty \leq L_{t+1} \ , \tag{10.34d}$$

where L_t a Lipschitz constant of V_{t+1}.

10.5.2. Numerical implementation

The forward and backward passes of dual SDDP are *independent* from the forward and backward passes of primal SDDP. Accordingly, a first "natural" implementation of the whole algorithm runs primal and dual SDDP in two independent processes, and thus enables to compute primal and dual value functions in parallel.

However, each backward pass of the primal SDDP algorithm computes a set of cuts whose slopes are $\left\{\lambda_t\right\}_{t\in\{0,..,T\}}$. As explained in Figure 10.4.1, these slopes can be considered as trajectories for the dual problem. If primal SDDP has converged, they are even the optimal co-state of the problem, because of the Fenchel-Young equality. Thereby, it may prove useful to view these sequences of slopes as trajectories for the dual problem, along which we run afterward a backward pass producing cuts for the dual problem. In this implementation, each iteration of the algorithm consists of four steps (the complete algorithm is given in §10.6.4):

1. Run a forward pass of primal SDDP Algorithm 4 and get trajectories $\left\{x_t\right\}_{t\in\{0,..,T\}}$.
2. Run a backward pass of primal SDDP Algorithm 4 along $\left\{x_t\right\}_{t\in\{0,..,T\}}$ and obtain new slopes $\left\{\lambda_t\right\}_{t\in\{0,..,T\}}$.
3. Run a backward pass of dual SDDP Algorithm 5 along $\left\{\lambda_t\right\}_{t\in\{0,..,T\}}$, thus updating the sets of cuts for the dual problem.
4. Run a forward pass of dual SDDP Algorithm 5 and update the cuts along the obtained trajectories.

The last step of this iteration ensures that we recover the convergence hypotheses of SDDP, as given in Girardeau et al. (2014), by having one set of cuts computed at point sampled along uncertainty drawn independently from the past. This algorithm has the same number of forward and backward passes as the original one (one forward pass and one backward pass in both the primal and the dual space). However, this scheme proves to be numerically more efficient, both in term of convergence and computation time. That is why we use this implementation in all the numerical experiments.

From the computational point of view, we implement primal and dual SDDP in Julia 0.6, with the `StochDynamicProgramming.jl` package built on top of the JuMP modeler of Dunning et al. (2017). We use Gurobi 7.02 to solve the LP subproblems. All experiments are run on a Intel Core i7-5500 CPU @2.4GHz, 64bit computer.

10.5.3. Results

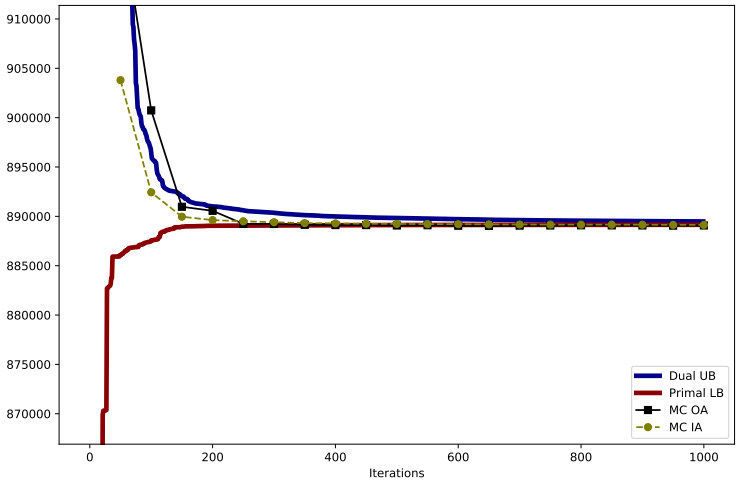
We consider the problem described at §10.5.1, which exhibits a 8-dimensional state. We aim to compute the value functions $\left\{V_t\right\}_{t\in\{0,..,T\}}$ with monthly time steps, and we consider different time horizons T, depending on the desired goal (illustration of convergence, comparison of bounds...). The uncertainties in the model are the inflows a_t in the reservoir and the demands d_t in every considered countries. Inflows and demands trajectories are simulated using a software provided by EDF, so that these data are realistic enough. From these simulated samples, we use quantization methods to obtain the marginal laws of the uncertainty $\boldsymbol{\xi}_t$ at each $t \in \{0, .., T\}$. The support of the quantized probability laws is limited to 10 possible values for $\boldsymbol{\xi}_t$ at each timestep t.

To solve the problem, we run primal and dual SDDP on 1,000 iterations, with a single forward pass in the primal and in the dual.

10.5.3.1. Assessing convergence

To ease the description of the results, we denote by `Primal LB` the primal lower bound $\underline{V}_0(x_0)$ obtained by primal SDDP, and by `Dual UB` the upper bound $[\underline{\mathcal{D}}_0]^\star(x_0)$ given by dual SDDP. The Monte Carlo cost evaluation obtained by simulating the outer (resp. inner) strategy uses the procedure described in §10.3.1.2, and is denoted by `MC OA` (resp. `MC IA`). Confidence intervals (with a confidence level $\alpha = 97.5\%$) are associated to these Monte Carlo approximations, and we denote by `MC OA UB` and `MC IA UB` the associated upper bounds of these intervals. Whereas `Primal LB` and `Dual UB` are deterministic bounds, `MC OA`, `MC IA` `MC OA UB` and `MC IA UB` are statistical quantities.

Solving the problem over a one-year time horizon. First, we run dual and primal SDDP on a twelve months problem, that is, with $T = 12$. Convergence of the optimal costs given by dual and primal SDDP is detailed in Figure 10.3. The two last columns in the table give the cumulative



Iter.	LB	UB	Gap	Time LB	Time UB
Unit	$\times 10^5$	$\times 10^5$	%	s	s
50	8.861	9.577	8.08	2.	8.
100	8.874	8.969	1.06	3.	22.
200	8.890	8.910	0.21	8.	72.
300	8.891	8.904	0.14	13.	153.
400	8.891	8.900	0.09	20.	275.
500	8.891	8.898	0.08	29.	443.
600	8.891	8.897	0.06	38.	651.
700	8.891	8.896	0.05	49.	888.
800	8.891	8.896	0.04	61.	1191.
900	8.891	8.895	0.04	74.	1534.
1000	8.891	8.895	0.03	89.	1928.

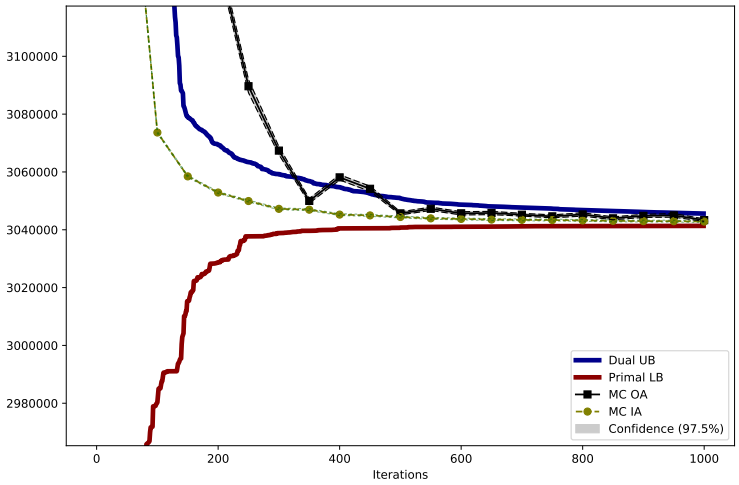
Figure 10.3.: Convergence of primal and dual SDDP for $T = 12$. Time corresponds to cumulated time along iterations.

computation times needed to run both primal and dual SDDP algorithms. We observe that the upper bound `Dual UB` $[\underline{\mathcal{D}}_0]^\star(x_0)$ given by dual SDDP converges towards the primal lower bound `Primal LB` $\underline{V}_0(x_0)$ given by primal SDDP, with a relative gap close to 0.03% after 1,000 iterations. For this specific (with few time steps) example, the convergence of dual SDDP proves to be effective. As noticed at Remark 10.3.13, running dual SDDP is much more time consuming than running

primal SDDP.

The outer and inner strategies are evaluated by Monte Carlo. An evaluation is performed every 50 iterations with an once for all given set of 10,000 scenarios, that is, a "large" sample. Both evaluations `MC OA` and `MC IA` converge to the optimal value. We notice that `MC IA` is below `Dual UB`, thus illustrating the result stated by Theorem 10.4.3.

Solving the problem over a three years time horizon. We now consider the same problem, but over a three years horizon, that is, with $T = 36$. The convergence of primal and dual SDDP is shown in Figure 10.4. Compared to Figure 10.3, we have materialized the confidence intervals



Iter.	LB	UB	Gap	Time LB	Time UB
Unit	$\times 10^6$	$\times 10^6$	%	s	s
50	2.837	3.917	38.1	5.	20.
100	2.980	3.151	5.7	11.	74.
200	3.029	3.070	1.4	27.	267.
300	3.039	3.059	0.67	46.	592.
400	3.040	3.055	0.46	75.	1113.
500	3.041	3.051	0.34	108.	1783.
600	3.041	3.049	0.25	144.	2601.
700	3.041	3.048	0.21	187.	3585.
800	3.041	3.047	0.18	235.	4751.
900	3.041	3.046	0.15	296.	6140.
1000	3.041	3.046	0.13	360.	7545.

Figure 10.4.: Convergence of primal and dual SDDP for $T = 36$. Time corresponds to cumulated time along iterations.

(here very thin) of the inner and outer strategies Monte Carlo simulations, both estimated every 50 iterations with an once for all given set of 10,000 scenarios. A first observation is that both dual and primal SDDP exhibit a slower convergence than in the first example: after 1,000 iterations, the gap between the primal lower bound `Primal LB` and the dual upper bound `Dual UB` is equal to 0.13%. This well-known behavior of SDDP arises from the increasing number of time-steps (36 instead of 12). Moreover, `Dual UB` is still significantly decreasing after iteration 500, and it seems that it converges more slowly than `Primal LB`.

A second observation is that the dual upper bound `Dual UB` is better than the statistical cost value `MC OA` up to iteration 500. After the first 500 iterations, `MC OA` is better that `Dual UB` and

slightly fluctuates above the primal lower bound `Primal LB` (the remaining gap being around 0.1% after 1,000 iterations).

Finally, on this example, the value of `MC OA` is greater than the value of `MC IA` at every iteration. Surprisingly enough, the value of `MC IA` exhibits a more stable behavior than the one given by `MC OA`. It would be interesting to be able to assess such behaviors.

10.5.3.2. Using the dual upper bound in a stopping criteria

Consider the problem over a three years time horizon. The gap between the two deterministic bounds (primal lower bound `Primal LB` and dual upper bound `Dual UB`) against the number of iterations is given in Figure 10.4. To complete these results, we give the evolution of the statistical upper bound `MC OA UB` obtained by the outer strategy in Table 10.1. We aim at comparing two stopping tests.

Iter.	`Primal LB` ($\times 10^6$)	Gap `Dual UB` (%)	`MC OA UB` ($\times 10^6$)	Gap `MC OA UB` (%)
50	2.837	38.1	3.392	19.6
100	2.980	5.7	3.310	11.1
200	3.029	1.4	3.137	3.6
300	3.039	0.67	3.069	1.0
400	3.040	0.46	3.059	0.62
500	3.041	0.34	3.046	0.18
600	3.041	0.25	3.046	0.18
700	3.041	0.21	3.046	0.15
800	3.041	0.18	3.046	0.16
900	3.041	0.16	3.045	0.14
1000	3.041	0.13	3.044	0.08

Table 10.1.: Statistical upper bound for $T = 36$.

Statistical stopping test: it is the stopping test proposed in Shapiro (2011) and which has been detailed at §10.1.2. We choose a confidence level $\alpha = .975$, and we estimate the statistical upper bound `MC OA UB` every 50 iterations with a given set of 10,000 scenarios.

Dual stopping test: this stopping test is just based on the gap between the available deterministic upper and lower bounds, namely `Dual UB` and `Primal LB`.

For different accuracy levels ε, as described by Shapiro (2011) we compare the CPU times taken by these two tests in order to stop the SDDP algorithm. Results are given in Table 10.2. The

	Dual stopping test		Statistical stopping test	
ε (%)	n it.	CPU time	n it.	CPU time
2.0	156	183s	250	618s
1.0	236	400s	300	787s
0.5	388	1116s	450	1429s
0.1	> 1000	.	1000	5519s

Table 10.2.: Comparing dual and statistical stopping criteria for different accuracy levels ε.

given times correspond to the total time required to run SDDP (including both the computation of cuts and the computation of the stopping test). We notice that the dual stopping test gives better results than the statistical stopping test: for $\varepsilon \geq 0.45\%$, it stops SDDP earlier and require less computation time. Compared with the statistical test, the speed-up is between 3.3 for $\varepsilon = 2\%$

and 1.3 for $\varepsilon = 0.5\%$. However, the dual stopping test is penalized by the slow convergence of dual SDDP. Indeed it cannot achieve a gap lower than 0.1 %, thus penalizing the performance of the dual stopping test for high accuracy levels ε.

As a conclusion of these numerical experiments, the deterministic dual stopping test seems to be better that the statistical stopping test, especially if restrictions on the CPU time impose to perform a limited number of SDDP iterations (less than 500 in our case). Such a situation exists in the energy field, as shown by the description of the Brazilian interconnected power system in Shapiro et al. (2012).

Remark 10.5.1. *We can also use the statistical upper bound* `MC IA UB` *obtained by evaluating the inner strategy for the statistical stopping test designed by Shapiro. Indeed, in our numerical experiments, this upper bound is always lower than the one given by the outer strategy. However, this would require much longer computational time, as this approach combines the computation of the dual cuts together with a Monte-Carlo estimation.* ♢

10.5.3.3. Strengths and weaknesses of dual SDDP

Dual SDDP allows us to obtain a deterministic stopping criterion, which proves to be effective compared to the standard statistical stopping test. Furthermore, dual SDDP computes cuts that can be used to design an inner approximation strategy, which appears to be better than the outer strategy whenever primal SDDP has not exactly converged.

However, we observe that the convergence of dual SDDP is penalized by different considerations.

- It is well-known from Shapiro (2011) that the convergence of SDDP is impacted by the number of stages in the problem. This issue impacts both primal and dual SDDP.
- Furthermore, we notice that dual SDDP exhibits a slower convergence than primal SDDP. In fact, primal SDDP computes its trajectories from a fixed initial point x_0, whereas, as explained at §10.3.2.2, dual SDDP updates its initial point λ_0 at each iteration, with

$$\lambda_0^k \in \underset{\|\lambda_0\| \leq L_0}{\arg\max} \left\{ x_0^\top \lambda_0 - \underline{\mathcal{D}}_0^k(\lambda_0) \right\}. \tag{10.35}$$

- One iteration of dual SDDP takes longer than one iteration of primal SDDP. Indeed, dual SDDP solves bigger LP problems than primal SDDP, as it has to consider explicitly a coupling constraint (10.34b) between all samples. Dual SDDP would greatly benefit from a cuts selection algorithm, which would limit the number of constraints added in the problem.

10.6. Conclusion

In this paper, we have described a new method to obtain an exact upper bound when using the SDDP algorithm, the computation of which relies on applying SDDP to the Fenchel transform of Bellman's value functions. We thus have built a sequence of inner approximations $\left\{\overline{V}_t\right\}_{t\in\{0,..,T\}}$ of the primal value functions $\left\{V_t\right\}_{t\in\{0,..,T\}}$, and proved that the policy induced is converging to an optimal policy, with guaranteed performance of the associated expected cost.

We have shown that under classical assumptions the sequence of upper bounds generated by the algorithm converges to the problem optimal value.

We have taken advantage of the dual value functions to build a sequence of inner approximations of the primal value functions. We proved that the policy induced by these inner approximations is converging to an optimal policy, with guaranteed performance of the associated expected cost. We tested dual SDDP and presented numerical results on a realistic stochastic production planning problem, proving the effectiveness of dual SDDP and the underlying inner strategy. Furthermore,

we showed on this particular problem that using a dual stopping test outperforms the classical statistical stopping test of SDDP, both in term of number of iterations and in term of computational burden.

We plan to extend this study in several directions. First, an extension of dual SDDP to risk averse or distributionally robust problems remains to be investigated. Second, the dual SDDP algorithm does not decompose the computation of the dual LBOs uncertainty by uncertainty A proper way to effectively decompose the dual subproblems is under study. Finally we want to explore the interactions between primal and dual SDDP. For example, we think that the upper bounds given by dual SDDP might be effective to regularize SDDP, for instance with the method introduced by Van Ackooij et al. (2017).

Appendix

10.6.1. Recalls on convex analysis

We quickly recall some results that can be found in any convex analysis book. We follow the definitions of Rockafellar (1970).

A convex extended real-valued function is *proper* if it never takes the value $-\infty$ and is not identically equal to $+\infty$. A *polyhedral subset* of $\mathbb{R}^n$ is the finite intersection of closed half spaces, and a *polyhedral function* is a function whose epigraph is a polyhedral set. In particular a polyhedral function is convex l.s.c., but not necessarily proper. A non-proper polyhedral function takes value $-\infty$ on a polyhedral set, and $+\infty$ elsewhere.

Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be an extended real-valued function. By definition its Fenchel conjugate $f^\star$ and concave conjugate $f_\star$ are defined as

$$f^\star(x^\star) := \sup_{x\in\mathbb{R}^n} \langle x^\star, x\rangle - f(x) \qquad f_\star(x^\star) := \inf_{x\in\mathbb{R}^n} \langle x^\star, x\rangle - f(x) \; . \tag{10.36}$$

Let f and $-g$ be proper polyhedral functions. Then we have the following duality result (see (Rockafellar, 1970, § 31))

Proposition 10.6.1 (Fenchel-Duality)**.** *If* $\mathrm{dom}(f) \cap \mathrm{dom}(-g) \neq \emptyset$*, we have*

$$\inf_x f(x) - g(x) = \sup_{x^\star} g_\star(x^\star) - f^\star(x^\star) \; .$$

We end these recalls by results on Lipschitz regularization. For any proper functions f and g of $\mathbb{R}^n$ we define the infimal convolution as

$$f \square g : x \mapsto \inf_{y\in\mathbb{R}^n} f(y) + g(x-y) \; . \tag{10.37}$$

Let f be a proper function of $\mathbb{R}^n$, we denote f^L its L-Lipschitz regularization – or Pasch Hausdorff regularization – with respect to the L^1-norm defined has

$$f^L := f \square (L\|\cdot\|_1) \; . \tag{10.38}$$

We have the following properties (see (Bauschke et al., 2017, Corollary 12.19))

Proposition 10.6.2. *Let f be a proper function of $\mathbb{R}^n$, and L be a positive number. Then, f^L is the largest L-Lipschitz function that is lower than f. In particular we have*

- *f^L is a lower approximation of f : $f^L \leq f$,*
- *f^L is L-Lipschitz with respect to the L_1 norm,*
- *if f is L-Lipschitz on its domain, then for all $x \in \mathrm{dom}(f)$, $f^L(x) = f(x)$.*

10.6.2. Omitted proofs

Lemma 10.6.3. *We have,*

$$\mathcal{T}^\ddagger_{t,L}\big(\underline{\mathcal{D}}^k_{t+1}\big) \geq \underline{\mathcal{D}}^k_t \qquad \forall t \in \{0,..,T\} \; . \tag{10.39}$$

Proof. Equation (10.39) is satisfied for $k = 0$. Assume that Equation (10.39) holds at iteration k. By definition of $\mathcal{C}^{k+1}$ in Algorithm 5, we have $\mathcal{C}^{k+1} \leq \mathcal{T}^\ddagger_{t,L_{t+1}}(\underline{\mathcal{D}}^{k+1}_{t+1})$. On the other hand, by monotonicity of $\mathcal{T}^\ddagger_{t,L_{t+1}}$, since $\underline{\mathcal{D}}^{k+1}_{t+1} \geq \underline{\mathcal{D}}^k_{t+1}$, we have $\mathcal{T}^\ddagger_{t,L_{t+1}}\big(\underline{\mathcal{D}}^{k+1}_{t+1}\big) \geq \mathcal{T}^\ddagger_{t,L_{t+1}}\big(\underline{\mathcal{D}}^k_{t+1}\big)$ which is greater than $\underline{\mathcal{D}}^k_t$ by induction hypothesis. Thus, $\mathcal{T}^\ddagger_{t,L_{t+1}}\big(\underline{\mathcal{D}}^{k+1}_{t+1}\big) \geq \max\big\{\underline{\mathcal{D}}^k_t, \mathcal{C}^{k+1}\big\} = \underline{\mathcal{D}}^{k+1}_t$.

□

Lemma 10.6.4. *Let $\overline{V}_t^k$ be the inner approximation of the value function V_t generated at iteration k of the dual SDDP algorithm. Then,*

$$\mathcal{T}_t(\overline{V}_{t+1}^k)(x) \leq \overline{V}_t^k(x) \qquad \forall t \in \{0, .., T\} \; . \tag{10.40}$$

Proof. We have

$$\begin{aligned}
\Big[\mathcal{T}_t(\overline{V}_{t+1}^k)\Big]^\star &= \mathcal{T}_t^\ddagger\Big(\Big[\overline{V}_{t+1}^k\Big]^\star\Big) && \text{by Theorem 10.2.9} \\
&= \mathcal{T}_t^\ddagger\big(\underline{\mathcal{D}}_{t+1}^k + \chi_{B_\infty(0,L_{t+1})}\big) && \text{by Proposition 10.4.2} \\
&= \mathcal{T}_{t,L}^\ddagger\big(\underline{\mathcal{D}}_{t+1}^k\big) \\
&\geq \underline{\mathcal{D}}_t^k && \text{by Lemma 10.6.3}
\end{aligned}$$

Furthermore, as $\mathcal{T}_t(\overline{V}_{t+1}^k)$ is polyhedral, we have

$$\mathcal{T}_t(\overline{V}_{t+1}^k) = \Big[\mathcal{T}_t(\overline{V}_{t+1}^k)\Big]^{\star\star} \leq \Big[\underline{\mathcal{D}}_t^k\Big]^\star \; ,$$

and as $\overline{V}_{t+1}^k$ is L_{t+1}-Lipschtiz, then $\mathcal{T}_t(\overline{V}_{t+1}^k)$ is L_t-Lipschtiz, thus $\mathcal{T}_t(\overline{V}_{t+1}^k) \leq \Big[\underline{\mathcal{D}}_t^k\Big]^\star \square(L_t\|\cdot\|)$ which ends the proof. □

10.6.3. Numerical settings

We describe the problem in §10.5 with dynamics and constraint matrix.

$$A = I_n \; , \;\; B = -(I_n \; I_n \; 0_n \; 0_n \; 0_{qn}) \; , \;\; C = (I_n \; 0_n) \; , \tag{10.41}$$

where I_n is the identity matrix and 0_n the square null matrix with size n. The costs vector becomes $a_t = 0$ and $b_t = (0 \; 0 \; c_t \; t_t \; p_t)^\top$. The constraints matrix write

$$D = \begin{pmatrix} 0_n & 0_n & 0_n & 0_n & 0_n & 0_n & 0_{qn} & 0_{qn} & A & -A \end{pmatrix}^\top \; , \tag{10.42}$$

and

$$E = \begin{pmatrix}
I_n & 0_n & 0_n & 0_n & 0_{qn} \\
-I_n & 0_n & 0_n & 0_n & 0_{qn} \\
0_n & -I_n & 0_n & 0_n & 0_{qn} \\
0_n & 0_n & I_n & 0_n & 0_{qn} \\
0_n & 0_n & -I_n & 0_n & 0_{qn} \\
0_n & 0_n & 0_n & -I_n & 0_{qn} \\
0_{qn} & 0_{qn} & 0_{qn} & 0_{qn} & I_q \\
0_{qn} & 0_{qn} & 0_{qn} & 0_{qn} & -I_q \\
I_n & 0_n & I_n & I_n & R \\
 & B & & & \\
 & -B & & &
\end{pmatrix} \; , \tag{10.43}$$

and

$$g_{t+1} = \begin{pmatrix} \overline{q} & 0 & 0 & \overline{g} & 0 & 0 & \overline{f} & \underline{f} & \boldsymbol{d}_{t+1} & \boldsymbol{d}_{t+1} & (\overline{v} - \boldsymbol{a}_{t+1}) & \boldsymbol{a}_{t+1} \end{pmatrix}^\top \tag{10.44}$$

10.6.4. Exhaustive primal-dual SDDP algorithm

Algorithm 10.4: Primal-Dual SDDP algorithm

Data: Lipschitz bounds $\{L_t\}_{t\in\{0,..,T\}}$
for $k = 0, 1, \dots$ **do**
  Draw a noise scenario $\left\{\xi_t^k\right\}_{t\in\{0,..,T\}}$;
  begin
    Primal Forward Pass : compute a set of trial points $\left\{x_t^k\right\}_{t\in\{0,..,T\}}$;
    Primal Backward Pass: refine primal value functions $\underline{V}_t^k$ along $\left\{x_t^k\right\}_{t\in\{0,..,T\}}$;
    Fetch computed cuts $\left\{\underline{\lambda}_t^{k+1}\right\}_{t\in\{0,..,T\}}$;
    Dual Backward Pass: refine dual value functions $\left\{\underline{\mathcal{D}}_{t+1}^{k+\frac{1}{2}}\right\}_{t\in\{0,..,T\}}$ along $\left\{\underline{\lambda}_t^{k+1}\right\}_{t\in\{0,..,T\}}$;
  end
  Draw a new noise scenario $\left\{\overline{\xi}_t^k\right\}_{t\in\{0,..,T\}}$;
  begin
    Compute $\lambda_0^k \in \arg\max_{\|\lambda_0\|\leq L_0} \left\{x_0^\top \lambda_0 - \underline{\mathcal{D}}_0^k(\lambda_0)\right\}$;
    Dual Forward Pass: compute a set of trial points $\left\{\overline{\lambda}_t^k\right\}_{t\in\{0,..,T\}}$ and update directly dual value functions $\left\{\underline{\mathcal{D}}_{t+1}^{k+1}\right\}_{t\in\{0,..,T\}}$;
  end
end

10.6.5. Compatibility of the primal and dual Bellman operators

We consider Problem (10.1) and its associated recursive Bellman equation. Ignoring the constant term $a_t^\top x$, this equation rewrites as a linear Bellman operator

$$\begin{aligned}\mathcal{B}_t(V_{t+1})(x) = \inf_{\boldsymbol{U},\boldsymbol{Y}} \quad & \mathbb{E}\Big[b_{t+1}^\top \boldsymbol{U} + V_{t+1}(\boldsymbol{Y})\Big] \,, \\ \text{s.t.} \quad & Tx + \mathcal{W}^u(\boldsymbol{U}) + \mathcal{W}^y(\boldsymbol{Y}) \leq \boldsymbol{H} \,,\end{aligned}$$

with the notation $T = [A_t \ -A_t \ 0 \ 0 \ 0 \ 0 \ D_t]^\top$, $\mathcal{W}^u = [B_{t+1} \ -B_{t+1} \ I \ -I \ 0 \ 0 \ E_{t+1}]^\top$, $\mathcal{W}_y = [-I \ I \ 0 \ 0 \ I \ -I \ 0]^\top$ and $\boldsymbol{H} = [-C_{t+1}\boldsymbol{\xi}_{t+1} \ C_{t+1}\boldsymbol{\xi}_{t+1} \ \overline{u}_{t+1} \ -\underline{u}_{t+1} \ \overline{x}_{t+1} \ -\underline{x}_{t+1} \ -G_{t+1}\boldsymbol{\xi}_{t+1}]^\top$.

Denoting by $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_7)$ the multiplier associated to the constraint of this problem, the Fenchel conjugate of $\mathcal{B}_t(V_{t+1})$ writes

$$\begin{aligned}\mathcal{B}_t^\ddagger(V_{t+1}^\star)(\lambda) = \inf_{\boldsymbol{\mu},\boldsymbol{\nu}} \quad & \mathbb{E}\Big[(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top C_{t+1}\boldsymbol{\xi}_{t+1} - \boldsymbol{\mu}_3^\top \overline{u}_{t+1} + \boldsymbol{\mu}_4^\top \underline{u}_{t+1} - \boldsymbol{\mu}_5^\top \overline{x}_{t+1} \\ & \qquad + \boldsymbol{\mu}_6^\top \underline{x}_{t+1} + \boldsymbol{\mu}_7^\top G_{t+1}\boldsymbol{\xi}_{t+1} + V_{t+1}^\star(\boldsymbol{\nu})\Big] \,, \\ \text{s.t.} \quad & A_t^\top \mathbb{E}\big[\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\big] + D_t^\top \mathbb{E}\big[\boldsymbol{\mu}_7\big] = -\lambda \,, \\ & B_{t+1}^\top(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + (\boldsymbol{\mu}_3 - \boldsymbol{\mu}_4) + E_{t+1}^\top \boldsymbol{\mu}_7 = b_{t+1} \,, \\ & -(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + (\boldsymbol{\mu}_5 - \boldsymbol{\mu}_6) = \boldsymbol{\nu} \,, \\ & \boldsymbol{\mu}_1 \leq 0 \,, \ \boldsymbol{\mu}_2 \leq 0 \,, \ \boldsymbol{\mu}_3 \leq 0 \,, \ \boldsymbol{\mu}_4 \leq 0 \,, \ \boldsymbol{\mu}_5 \leq 0 \,, \ \boldsymbol{\mu}_6 \leq 0 \,, \ \boldsymbol{\mu}_7 \leq 0 \,.\end{aligned}$$

We make the following assumption:

$$\forall \lambda \in \mathbb{R}^n,\ \exists \mu_a \in \mathbb{R}^n,\ \exists \mu_b \in \mathbb{R}^p,\ \mu_b \leq 0 \ \text{ such that } \ A_t^\top \mu_a + D_t^\top \mu_b + \lambda = 0\ .$$

Note that this assumption is fulfilled if A_t is a (square) full rank matrix. Then, with any arbitrary non positive random variables $\boldsymbol{\mu}_5$ and $\boldsymbol{\mu}_6$, the pair of random vectors $(\boldsymbol{\mu}, \boldsymbol{\nu})$ defined by

- $\boldsymbol{\mu} = \Big((\mu_a)^-, -(\mu_a)^+, (b_{t+1} - B_{t+1}^\top \mu_a - E_{t+1}^\top \mu_b)^-, -(b_{t+1} - B_{t+1}^\top \mu_a - E_{t+1}^\top \mu_b)^+, \boldsymbol{\mu}_5, \boldsymbol{\mu}_6, \mu_b\Big)$,
- $\boldsymbol{\nu} = -\mu_a + (\boldsymbol{\mu}_5 - \boldsymbol{\mu}_6)$,

satisfies the constraints of the optimization problem associated to the computation of $\mathcal{B}_t^\ddagger(V_{t+1}^\star)(\lambda)$. Such a pair $(\boldsymbol{\mu}, \boldsymbol{\nu})$ exists for all possible values of λ. Moreover, $\boldsymbol{\nu}$ linearly depends on the difference $\boldsymbol{\mu}_5 - \boldsymbol{\mu}_6$, $\boldsymbol{\mu}_5$ and $\boldsymbol{\mu}_6$ being any arbitrary negative random variables, so that $\boldsymbol{\nu}$ can take any possible value in $\mathbb{R}^n$. We thus deduce that the domain of the dual constraint set valued mapping $\mathcal{G}_t^\ddagger$, defined by Equation (10.8), is equal to the whole space $\mathbb{R}^n$, so that the sequence of dual linear Bellman operators $\big(\mathcal{B}_t^\ddagger\big)_{t\in\{0,..,T-1\}}$ is compatible.

Chapter 11.

A complement on Fenchel duality and Dynamic Programming

This chapter is a joint work with Maël Forcier. The author thanks him warmly for his precious help.

11.1. Introduction

This chapter is a complement of Chapter 10. It gives a proof of convergence for the abstract SDDP algorithm and some interpretation concerning the dual version of the algorithm. We refer to Philpott and Guan (2008) for the first proof of SDDP, in the linear case, and to Girardeau et al. (2014) for an extension of the proof in the convex case.

11.2. Alternating forward and backward passes

We reconsider the notation introduced in Chapter 10. SDDP approximates the original value functions $\{R_t\}$ as a supremum of affine cuts $\{\underline{R}_t^k\}$

$$\underline{R}_t^k(x) = \max_{\kappa=1,\cdots,k} \{\langle \lambda_t^\kappa \, , x - x_t^\kappa \rangle + \theta_t^\kappa\} \, , \quad \forall x \in \mathcal{X}_t \, , \tag{11.1}$$

where the parameters $\{\theta_t^\kappa, x_t^\kappa, \lambda_t^\kappa\}_{\kappa=1,\cdots,k}$ are computed iteratively. We view $x \to \langle \lambda_t^\kappa \, , x - x_t^\kappa \rangle + \theta_t^\kappa$ as an affine cut.

At iteration k, SDDP updates the set of value functions $\{\underline{R}_t^k\}$ by sampling a new set of parameters $(\theta_t^k, x_t^k, \lambda_t^k)$ to refine the piecewise approximation.

Forward pass. SDDP samples iteratively primal points by considering the value functions $\{\underline{R}_t^k\}$

$$x_0^k \rightsquigarrow x_1^k \rightsquigarrow \cdots \rightsquigarrow x_T^k \, ,$$

such that

$$\begin{cases} \omega_t^k \in \Omega_t \\ \boldsymbol{X}_{t+1}^k = \mathcal{S}_t(\underline{R}_{t+1}^k)(x_t^k) \\ x_{t+1}^k = \boldsymbol{X}_{t+1}^k(\omega_t^k) \, . \end{cases} \tag{11.2}$$

Backward pass. SDDP updates the cost-to-go by considering new cuts computed backward:

$$\lambda_T^{k+1} \rightsquigarrow \lambda_{T-1}^{k+1} \rightsquigarrow \cdots \rightsquigarrow \lambda_0^{k+1} \, ,$$

and new values θ_t^{k+1} satisfying

$$\begin{cases} \theta_t^{k+1} = \mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k) \\ \lambda_t^{k+1} = \partial\mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k) \ . \end{cases} \tag{11.3}$$

Then, it updates the value function $\underline{R}_t^k$ by

$$\underline{R}_t^{k+1}(\cdot) = \max\{\underline{R}_t^k(\cdot), \langle \lambda_t^{k+1} \ , \cdot - x_t^k \rangle + \theta_t^{k+1}\} \ . \tag{11.4}$$

We note that for all iteration k, the approximated value functions $\{\underline{R}_t^k\}$ depends on parameters $\{\theta_t^\kappa, x_t^\kappa, \lambda_t^\kappa\}_{\kappa=1,\cdots,k}$, computed during previous iterations. To ease the notation, we define the cuts operator $\mathcal{C}_t$:

$$\mathcal{C}_t(x_t^{1:k}, \theta_t^{1:k}, \lambda_t^{1:k})(x) = \max_{\kappa=1,\cdots,k} \{\langle \lambda_t^\kappa \ , x - x_t^\kappa \rangle + \theta_t^\kappa\} \ . \tag{11.5}$$

11.3. A proof of convergence of abstract SDDP

We adapt the proof of SDDP presented in Girardeau et al. (2014).

$\mathcal{Q}$-value function. We define the intermediate $\mathcal{Q}$ value function by, for all functions R

$$\mathcal{Q}_t(R)(\boldsymbol{U}, \boldsymbol{Y}) = \mathbb{E}\big[\boldsymbol{C}_t^\top \boldsymbol{U} + R(\boldsymbol{Y})\big] = \sum_{i=1}^{|\Omega|} \pi_i \big[\boldsymbol{C}_t(\omega_i)^\top \boldsymbol{U}_t(\omega_i) + R\big(\boldsymbol{Y}(\omega_i)\big)\big] \ . \tag{11.6}$$

By definition the Bellman operator $\mathcal{B}_t$ satisfies

$$\mathcal{B}_t(R)(x_t) = \inf_{(\boldsymbol{U},\boldsymbol{Y}) \in \mathcal{G}_t(x_t)} \mathcal{Q}_t(R)(\boldsymbol{U}, \boldsymbol{Y}) \ , \ \ \forall x_t \in \mathcal{X}_t \ . \tag{11.7}$$

We recall basic facts about the SDDP algorithm.

Fact 11.3.1. *Let $t \in \{0, .., T-1\}$ and $\{\underline{R}_t^k\}_{k\in\mathbb{N}}$ value functions generated by the above procedure. Then, the value functions are lower-bounds of the original value function R_t*

$$\underline{R}_t^k \leq R_t \ , \ \ \forall k \in \mathbb{N} \ . \tag{11.8}$$

Fact 11.3.2. *Let $t \in \{0, .., T-1\}$, $\underline{R}_t^k$ and $\underline{R}_{t+1}^k$ the approximated value function at time t and $t+1$, and $\mathcal{B}_t$ the Bellman operator. Then, we have, at iteration $k \in \mathbb{N}$ and x_t^k sampled during the forward pass:*

$$\begin{aligned} &\underline{R}_t^k(x_t^k) = \mathcal{B}_t(\underline{R}_{t+1}^k)(x_t^k) \\ &\underline{R}_t^k(x) \leq \mathcal{B}_t(\underline{R}_{t+1}^k)(x) \ , \ \ \forall x \in \mathcal{X}_t \ . \end{aligned} \tag{11.9}$$

Sketching the proof in three steps. The proof of abstract SDDP is divided in three steps:

1. First, we prove that the value functions converge along the sampled trajectories;
2. Then, we prove the convergence along all state process generated by the noise process;
3. Finally, we state the final theorem.

11.3.1. Convergence along sampled trajectories

We first prove that SDDP converges along the state trajectories corresponding to the noise scenarios $\omega^k = (\omega_0^k, \cdots, \omega_T^k)$ sampled during the forward passes of SDDP.

We begin to prove the convergence by shifting the iteration and prove that the value function $\underline{R}_t^{k+1}$ converges to the original value function R_t along the points x_t^k sampled during SDDP forward passes.

Lemma 11.3.3. *Consider the approximated value functions $\{\underline{R}_t^k\}_{k\in\mathbb{N}}$ and sequence of state values $\{x_t^k\}_{k\in\mathbb{N}}$ sampled by the above procedure. Let $t \in \{0, .., T-1\}$. We suppose that*

$$\lim_{k\to\infty} R_{t+1}(\boldsymbol{X}_{t+1}^k(\omega_t)) - \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k(\omega_t)) = 0\ ,\ \ \forall \omega_t \in \Omega_t\ . \tag{11.10}$$

Then,

$$\lim_{k\to\infty} R_t(x_t^k) - \underline{R}_t^{k+1}(x_t^k) = 0\ , \tag{11.11a}$$

and

$$\lim_{k\to\infty} \mathcal{B}_t(R_{t+1})(x_t^k) - \mathcal{Q}_t(R_{t+1})(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) = 0\ . \tag{11.11b}$$

Proof. We have, for all $x_t^k \in \mathcal{X}_t$,

$$\begin{aligned}
\underline{R}_t^{k+1}(x_t^k) &= \max_{\kappa\leq k+1} \theta_t^\kappa + \left\langle \lambda_t^\kappa\ , x_t^k - x_t^{\kappa-1}\right\rangle \\
&\geq \theta_t^{k+1} && \text{by taking } \kappa = k+1 \\
&= \mathcal{B}_t(\underline{R}_{t+1}^{k+1})(x_t^k) \\
&\geq \mathcal{B}_t(\underline{R}_{t+1}^{k})(x_t^k) && \text{as } \underline{R}_{t+1}^k \leq \underline{R}_{t+1}^{k+1} \text{ and } \mathcal{B}_t \text{ is monotone} \\
&= \inf_{(\boldsymbol{U},\boldsymbol{Y})\in\mathcal{G}_t(x_t^k)} \mathcal{Q}_t(\underline{R}_{t+1}^k)(\boldsymbol{U},\boldsymbol{Y}) \\
&= \mathcal{Q}_t(\underline{R}_{t+1}^k)(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) && \text{as } \boldsymbol{X}_{t+1}^k \in \mathcal{S}_{t+1}(\underline{R}_{t+1}^k)(x_t^k)
\end{aligned}$$

Thus, we have :

$$\begin{aligned}
0 \leq R_t(x_t^k) - \underline{R}_t^{k+1}(x_t^k) &\leq R_t(x_t^k) - \mathcal{Q}_t(\underline{R}_{t+1}^k)(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) \\
&= R_t(x_t^k) - \mathbb{E}\Big[\boldsymbol{C}_t^\top \boldsymbol{U}_t^k + \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k)\Big] \\
&= R_t(x_t^k) - \mathbb{E}\Big[\boldsymbol{C}_t^\top \boldsymbol{U}_t^k + R_{t+1}(\boldsymbol{X}_{t+1}^k) - R_{t+1}(\boldsymbol{X}_{t+1}^k) + \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k)\Big] \\
&= R_t(x_t^k) - \mathbb{E}\Big[\boldsymbol{C}_t^\top \boldsymbol{U}_t^k + R_{t+1}(\boldsymbol{X}_{t+1}^k)\Big] + \mathbb{E}\Big[R_{t+1}(\boldsymbol{X}_{t+1}^k) - \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k)\Big] \\
&= \mathcal{B}_t(R_{t+1})(x_t^k) - \mathcal{Q}_t(R_{t+1})(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) + \mathbb{E}\Big[R_{t+1}(\boldsymbol{X}_{t+1}^k) - \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k)\Big] \\
&\leq \mathbb{E}\Big[R_{t+1}(\boldsymbol{X}_{t+1}^k) - \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k)\Big]\ ,
\end{aligned}$$

Indeed by definition of $\mathcal{B}_t(R_{t+1})(x_t^k)$, and as $(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) \in \mathcal{G}_t(x_t^k)$ is admissible, we have

$$\mathcal{B}_t(R_{t+1})(x_t^k) - \mathcal{Q}_t(R_{t+1})(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) \leq 0\ .$$

So, noting $A_k := \mathbb{E}\Big[R_{t+1}(\boldsymbol{X}_{t+1}^k) - \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k)\Big]$ we get the following inequalities

$$0 \leq R_t(x_t^k) - \underline{R}_t^{k+1}(x_t^k) \leq A_k\ ,$$

$$-A_k \leq \mathcal{B}_t(R_{t+1})(x_t^k) - \mathcal{Q}_t(R_{t+1})(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) \leq 0\ .$$

But by induction hypothesis :

$$\begin{aligned}\lim_{k\to\infty} A_k &= \lim_{k\to\infty} \mathbb{E}\big[R_{t+1}(\boldsymbol{X}_{t+1}^k) - \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k)\big] \\ &= \sum_{i=1}^{|\Omega|} \pi_i \lim_{k\to\infty} \Big(R_{t+1}(\boldsymbol{X}_{t+1}^k(\omega_i)) - \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k(\omega_i))\Big) \\ &= 0 \ .\end{aligned}$$

As a consequence, we have

$$\lim_{k\to\infty} \mathcal{B}_t(R_{t+1})(x_t^k) - \mathcal{Q}_t(R_{t+1})(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) = 0 \ ,$$

$$\lim_{k\to\infty} R_t(x_t^k) - \underline{R}_t^{k+1}(x_t^k) = 0 \ .$$

Hence the results. □

Now, we extend the result of Lemma 11.3.3 by proving the convergence for $\underline{R}^k$, without index shifting. To do so, we need the following lemma.

Lemma 11.3.4. *Let g a convex function and $\mathcal{X}$ a compact. Suppose there exists a sequence of α-Lipschitz convex functions g^k satisfying*

$$g^k(x) \leq g^{k+1}(x) \leq g(x) \ , \ \ \forall k \in \mathbb{N}, \forall x \in \mathcal{X} \ . \tag{11.14}$$

Then for any infinite sequence $(x^k)_k \in \mathcal{X}^{\mathbb{N}}$

$$\lim_{k\to\infty} g(x^k) - g^{k+1}(x^k) = 0 \Longleftrightarrow \lim_{k\to\infty} g(x^k) - g^k(x^k) = 0 \ . \tag{11.15}$$

Proof. This is a direct application of (Girardeau et al., 2014, Lemma A1) with $\kappa = 1$ and $g^k = f^{k-1}$. □

Lemma 11.3.5. *Consider the approximated value functions $\{\underline{R}_t^k\}_{k\in\mathbb{N}}$ and sequence of state values $\{x_t^k\}_{k\in\mathbb{N}}$ sampled by the above procedure. Let $t \in \{0, .., T-1\}$. We suppose that*

$$\lim_{k\to\infty} R_{t+1}(\boldsymbol{X}_{t+1}^k(\omega_t)) - \underline{R}_{t+1}^k(\boldsymbol{X}_{t+1}^k(\omega_t)) = 0 \ , \ \ \forall \omega_t \in \Omega_t \ . \tag{11.16}$$

Then,

$$\lim_{k\to\infty} R_t(x_t^k) - \underline{R}_t^k(x_t^k) = 0 \ . \tag{11.17}$$

Proof. By Lemma 11.3.3 we have,

$$\lim_{k\to\infty} R_t(x_t^k) - \underline{R}_t^{k+1}(x_t^k) = 0 \ .$$

We have also proven for all k that $||\lambda_t^k||_\infty \leq L_t$, so $\underline{R}_t^k(.) = \max_{\kappa\leq k} \theta_t^\kappa + \left\langle \lambda_t^\kappa \ , \cdot - x_t^{\kappa-1} \right\rangle$ is Lispchitz continuous with constant L_t. By applying Lemma 11.3.4 to $\{\underline{R}_t^k\}_k$, R_t and $\{x_t^k\}_k$, we get

$$\lim_{k\to\infty} R_t(x_t^k) - \underline{R}_t^k(x_t^k) = 0 \ .$$

Hence the results. □

11.3.2. Proving the convergence for all atoms

We extend the convergence property for all realizations of the discrete noise process. This result relies on the law of large numbers: as for all i, $\pi_i > 0$ the set $S_{t,i} := \{k \in \mathbb{N} \mid \omega_t^k = \omega_i\}$ is infinite.

Let $t \in \{0, .., T-1\}$ and $\varepsilon > 0$. We define the set corresponding to the iterations $k \in \mathbb{N}$ where the discrepancy between the original value function R_t and its approximation $\underline{R}_t^k$ is above ε along the state $\boldsymbol{X}_t^k$ sampled during the forward passes:

$$\mathcal{K}_{t,i,\varepsilon} = \{k \in \mathbb{N} \mid R_t(\boldsymbol{X}_t^k(\omega_i)) - \underline{R}_t^k(\boldsymbol{X}_t^k(\omega_i)) \geq \varepsilon\} \ . \tag{11.18}$$

We define accordingly two Boolean stochastic processes $\{v_i^k\}_k$, $\{y_i^k\}_k$ as

$$\begin{aligned} y_{t,i}^k &= \mathbb{I}_{\{\omega_t^k = \omega_i\}} = \mathbb{I}_{\{k \in S_i\}} \\ v_{t,i}^k &= \mathbb{I}_{\{k \in \mathcal{K}_{t,i,\varepsilon}\}} \ . \end{aligned}$$

At iteration k, $v_{t,i}^k$ states whether the approximation $\underline{R}_t^k$ as converged for the realization ω^i, and $y_{t,i}^k$ states whether the noise ω^i has been selected during the forward pass.

We define the *selection stochastic process* as

$$y^k = \{y_i^k\}_{t \in \{0,..,T\}, i \in \Omega_t} \ , \tag{11.19}$$

that corresponds to the nodes selected at iterations k. We prove that if we know the realization of the process $\{y^k\}_k$ we are able to deduce the realization of the process $v_{t,i}^k$.

As we deal with information, we introduce the σ-field corresponding to information brought by the process y^k up to iteration k:

$$\mathcal{F}^k = \sigma(\{y^\kappa\}_{\kappa \in \{1,..,k\}}) \ , \tag{11.20}$$

and the σ-field corresponding to the available information during pass k:

$$\mathcal{F}_{\to t}^k = \sigma(\mathcal{F}^{k-1}, \{y_{s,i}^k\}_{1 \leq s \leq t}) \ . \tag{11.21}$$

Lemma 11.3.6. *The random variable $v_{t,i}^k$ is measurable w.r.t. $\mathcal{F}_{\to t-1}^k$:*

$$\sigma(v_i^k) \subset \mathcal{F}_k^{\to t-1} \ .$$

Proof. Let $k \in \mathbb{N}$. Using the cut operator defined in Equation (11.5), we have that

$$\underline{R}_t^k = \mathcal{C}_t(\{x_t^\kappa, \theta_t^{\kappa+1}, \lambda_t^{\kappa+1}\}_{\kappa=1,\cdots,k}) \ , \tag{11.22}$$

thus, the approximated value function $\underline{R}_t^k$ depends on trajectories previously sampled. We deduce that the function $\underline{R}_t^k$ is a random variable. We have:

- The previously selected noises $\{\omega_s^k\}_{1 \leq s \leq t}$ depend on $\{y_{s,i}^k\}_{1 \leq s \leq t}$.
- By Equation (11.2), $\boldsymbol{X}_t^k$ depends on previous selected noises $\{\omega_s^k\}_{1 \leq s \leq t}$ and on current value functions $\underline{R}_t^k$.
- By Equation (11.3), θ_t^{k+1} and λ_t^{k+1} depend on x_t^k and on R_t^{k+1}.

We deduce that $\boldsymbol{X}_t^k$ is measurable w.r.t. $\sigma(\mathcal{F}^k, \mathcal{F}_{\to t}^k)$, and we prove by backward induction that θ_t^{k+1} and λ_t^{k+1} (and thus $\underline{R}_t^{k+1}$ by Equation (11.22)) are measurable w.r.t. $\mathcal{F}^k$.

Furthermore, we have that

$$v_i^k = \mathbb{I}_{\{k \in \mathcal{K}_{i,\varepsilon}\}} = \mathbb{I}_{\{R_t(\boldsymbol{X}_t^k(\omega_i)) - \underline{R}_t^k(\boldsymbol{X}_t^k(\omega_i)) \geq \varepsilon\}} \ , \tag{11.23}$$

thus v_i^k depends on $\boldsymbol{X}_t^k$ and $\underline{R}_t^k$. We deduce that

$$\sigma(v_i^k) \subset \sigma(\mathcal{F}_{\to t}^k, \mathcal{F}^{k-1}) = \mathcal{F}_{\to t}^k \ . \tag{11.24}$$

Hence the results. □

We recall Lemma A3 of Girardeau et al. (2014) :

Lemma 11.3.7. *Let $\{v\}_{k\in\mathbb{N}}$ be a Boolean stochastic process in $\{0,1\}$ adapted to a filtration $(\mathcal{F}^k)_{k\in\mathbb{N}}$ such that the number of terms that are non zero are almost surely infinite. Let $\{y^k\}_{k\in\mathbb{N}}$ be a sequence of i.i.d. discrete random variables. Define the filtration $\mathcal{G}^k := (\sigma(\mathcal{F}^k \cup \sigma(y^1, ..., y^{k-1}))$ and assume that for all $k \in \mathbb{N}, y^k$ is independent of $\mathcal{G}^k$. Let χ an extraction such that $\{k \mid v^k = 1\} = \{\chi(k) \mid k \in \mathbb{N}^\star\}$. Finally we define*

$$z^k := y^{\chi(k)} \ .$$

Then $(z^k)_{k\in\mathbb{N}}$ is a sequence of i.i.d. random variables equal in law to y^0.

Proof. See the technical proof in (Girardeau et al., 2014, Lemma A3). □

We deduce from the two previous lemma the following results.

Lemma 11.3.8. *Consider the approximated value functions $\{\underline{R}_t^k\}_{k\in\mathbb{N}}$, sequence of state process $\{\boldsymbol{X}_t^k\}_{k\in\mathbb{N}}$ and noise trajectories $\{\omega_t^k\}_{k\in\mathbb{N}}$ sampled by the above procedure. We suppose that*

$$\lim_{k\to\infty} R_t(\boldsymbol{X}_t^k(\omega_t^k)) - \underline{R}_t^k(\boldsymbol{X}_t^k(\omega_t^k)) = 0 \ , \tag{11.25}$$

then we obtain the almost-sure convergence, for all $\omega_i \in \Omega_t$

$$R_t(\boldsymbol{X}_t^k(\omega_i)) - \underline{R}_t^k(\boldsymbol{X}_t^k(\omega_i)) = 0 \ . \tag{11.26}$$

Proof. By contradiction, we suppose there exists $\omega_i \in \Omega_t$ such that

$$R_t(\boldsymbol{X}_t^k(\omega_i)) - \underline{R}_t^k(\boldsymbol{X}_t^k(\omega_i)) \underset{k\to\infty}{\not\to} 0 \ .$$

There exists an $\varepsilon > 0$ such that the set $\mathcal{K}_{i,\varepsilon}$ (defined in Equation (11.18)) is infinite.

We can then define an extraction χ such that

$$\{\chi(k), k \in \mathbb{N}\} = \mathcal{K}_{i,\varepsilon} \ . \tag{11.27}$$

We define the Boolean stochastic processes $\{z^k\}_k$ as

$$z^k = y_{t,i}^{\chi(k)} \ . \tag{11.28}$$

By Lemma 11.3.7, we know that the process $\{z^k\}_k$ is a sequence of i.i.d random variables equal in law to $y_{t,i}^1$.

According to the law of large number, we should have almost surely:

$$\frac{1}{N}\sum_{k=1}^{N} z^k \underset{N\to\infty}{\longrightarrow} \mathbb{E}(z^1) = \mathbb{E}(y_{t,i}^1) = \mathbb{P}(y_{t,i}^1 = 1) = \mathbb{P}(\omega^1 = \omega_i) = \pi_i > 0.$$

But by the convergence hypothesis (11.25), we have that $\mathcal{K}_{i,\varepsilon} \cap S_i = \{k \in \mathbb{N} \mid R_t(\boldsymbol{X}_t^k(\omega^k)) -$

$\underline{R}_t^k(\boldsymbol{X}_t^k(\omega^k)) \geq \varepsilon \ , \ \omega^k = \omega_i\}$ is finite a.s. Then,

$$0 \leq \frac{1}{N}\sum_{k=1}^{N} z^k \leq \frac{1}{N}\sum_{k=1}^{\infty} z^k = \frac{1}{N}\sum_{k\in\mathcal{K}_{i,\varepsilon}} y_i^k = \frac{|\mathcal{K}_{i,\varepsilon} \cap S_i|}{N} \underset{N\to\infty}{\longrightarrow} 0. \quad a.s.$$

We obtain a contradiction. Hence the conclusion. □

11.3.3. Convergence of abstract SDDP

We are now able to prove the convergence of the overall abstract SDDP algorithm.

Theorem 11.3.9. *Let $t \in \{0, .., T-1\}$, $\{\underline{R}_t^k\}$, $\{x_t^k\}$ and $\{\boldsymbol{X}_t^k\}$ generated by the above procedure. We have the almost-sure convergences*

$$\lim_{k\to\infty} R_t(\boldsymbol{X}_t^k(\omega_i)) - \underline{R}_t^k(\boldsymbol{X}_t^k(\omega_i)) = 0 \ , \ \ \forall \omega_i \in \Omega_t \ , \tag{11.29a}$$

$$\lim_{k\to\infty} \mathcal{B}_t(R_{t+1})(x_t^k) - \mathcal{Q}_t(R_{t+1})(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) = 0 \ . \tag{11.29b}$$

Proof. We prove (11.29a) by backward induction on t. Since for all k, $R_T = \underline{R}_T^k = K$, the induction hypothesis is true at time T. We now suppose that the induction hypothesis is true at time $t+1$ where $t \leq T-1$.

By applying Lemma 11.3.5 and Lemma 11.3.8 we obtain the first inequality (11.29a):

$$\lim_{k\to\infty} R_t(\boldsymbol{X}_t^k(\omega_i)) - \underline{R}_t^k(\boldsymbol{X}_t^k(\omega_i)) = 0 \ , \ \ \forall \omega_i \in \Omega_t \ . \tag{11.30}$$

Thanks to Lemma 11.3.3 we have (11.29b):

$$\lim_{k\to\infty} \mathcal{B}_t(R_{t+1})(x_t^k) - \mathcal{Q}_t(R_{t+1})(\boldsymbol{U}_t^k, \boldsymbol{X}_{t+1}^k) = 0 \ . \tag{11.31}$$

Hence the convergence of the abstract SDDP algorithm. □

11.4. Conclusion

In this chapter, we extend Chapter 10 by adapting the proof of convergence of Girardeau et al. (2014) to the abstract SDDP algorithm introduced in §10.2.3. We then detail some numerical schemes to combine primal SDDP with dual SDDP. We refer to the conclusion of Chapter 10 for an overview of the possible extensions.

Conclusion

It is good to have an end to journey towards; but it is the journey that matters, in the end.

Ursula K. Le Guin

This last chapter concludes three years of works between the Optimization and Systems group — at CERMICS — and Efficacity — a French institute for energy transition. This work is in continuation of previous scientific works done at the Optimization and Systems group, and develops some ideas introduced in several PhD theses (Girardeau, 2010)-(Alais, 2013)-Leclère (2014).

We sum up hereafter the different contributions and perspectives for each chapter of this manuscript.

Contributions

The main contributions of this thesis are the following.

- In Chapter 3, we have framed different existing stochastic algorithms in a common framework that allows to classify online policies generation methods.
- In Chapter 4, we have introduced a proper modelling for urban microgrids. This modelling proves to be tractable for the use of stochastic optimization algorithms.
- In Chapter 5, we have compared the performance of three online policies — a heuristic, a lookahead policy based on the Model Predictive Control (MPC) algorithm and a cost-to-go policy based on the Stochastic Dual Dynamic Programming (SDDP) algorithm — to the control of a domestic microgrid equipped with a micro-Combined Heat and Power generator and a battery. Numerical results have illustrated that whereas optimization based policies outperformed the heuristic, the performance of MPC and SDDP were too close to conclude on the predominance of one algorithm over the other.
- In Chapter 6, we have extended the comparison between the SDDP based cost-to-go policy and the MPC based lookahead policy, by dealing more carefully with the online information brought by the past realizations of uncertainties. We have applied these policies to another domestic microgrid example, this time equipped with solar panels and battery. Numerical results have depicted the effectiveness of the SDDP based policy, which has yielded better results in assessment than the MPC based policy.
- In Chapter 7, we have broadened the formalism lying behind multistage stochastic decomposition algorithms by considering generic coupling constraints between spatial units. We have presented in this formalism two decomposition algorithms, based respectively on price decomposition and resource decomposition. We have displayed conditions under which spatial and temporal decompositions are compatible.
- In Chapter 8, we have applied the price and resource decomposition algorithms to coupling constraints formulated on a graph. We have compared these two decomposition algorithms with a state-of-the-art SDDP solver, and showed the effectiveness of the decomposition algorithms to solve large-scale problems (w.r.t. the number of state variables).

- In Chapter 9, we have presented the results of a common work with Pierre Carpentier, Jean-Philippe Chancelier and Vincent Leclère, aiming to apply decomposition algorithms on dams-valley problems. This chapter originated from an article published in 2018 (Carpentier et al., 2018b).
- In Chapter 10, we have inserted the result of another common work — this time with Pierre Carpentier, Jean-Philippe Chancelier, Vincent Leclère and Arnaud Lenoir — which was submitted in 2018 (Leclère et al., 2018).
- In Chapter 11, we have provided a short complement to the previous chapter, with a proof of convergence of the abstract SDDP algorithm introduced in Chapter 10. This chapter was a joint work with Maël Forcier.

Perspectives

The different contributions open several doors, leading to several possible extensions.

- We believe that the comparison between lookahead and cost-to-go policies in Chapter 3 can be refined, for instance by considering more complicated lookahead policies (such as stochastic MPC methods) or by updating the value functions of the cost-to-go policies online. Another approach would consider a mix of these two classes of policy, where the final value functions of (stochastic) MPC would be computed offline by Dynamic Programming.
- The proper choice of the coordination design in Chapter 7 remains a challenge. We have chosen to focus only on deterministic designs, but we obtained poor results for resource decomposition in Chapter 8.
- Concerning decomposition algorithms, we have tested mainly price and resource decomposition algorithms. On the problems studied in Chapter 8, other decomposition algorithms gave poor results (such as ADMM, but the implementation used was not so elaborated) or did not converge at all (like interaction-prediction based decomposition).
- In Chapter 8, we have decomposed the problems node by node, and solved each nodal problem with the Stochastic Dual Dynamic Programming algorithm. An extension would be to decompose the problems zone by zone rather than node by node. By doing so, we would obtain larger suproblems, but whose sizes remain tractable to apply SDDP effectively.
- In Chapter 10, we have obtained a new manner to get a deterministic upper bound for SDDP. However, the resolution of the dual problem remains slower than the resolution of the primal problem, as all dual one-step problems deal with larger LP than primal one-step problems. We believe that we could improve the performance of the dual algorithm by taking advantage of the couplings existing between the primal and the dual problems.

Bibliography

ADEME (2017). Chiffres clés - Climat, Air et Énergie. Technical report, ADEME.

Alais, J.-C. (2013). *Risque et optimisation pour le management d'énergies*. Thèse de doctorat, Université Paris-Est.

Arrow, K. J. and Hurwicz, L. (1958). *Decentralization and computation in resource allocation*. Stanford University, Department of Economics.

Bacaud, L., Lemaréchal, C., Renaud, A., and Sagastizábal, C. A. (2001). Bundle methods in stochastic optimal power management: A disaggregated approach using preconditioner. *Computational Optimization and Applications*, 20(3):227–244.

Bacher, P., Madsen, H., and Nielsen, H. A. (2009). Online short-term solar power forecasting. *Solar Energy*, 83(10):1772–1783.

Baetens, R. and Saelens, D. (2016). Modelling uncertainty in district energy simulations by stochastic residential occupant behaviour. *Journal of Building Performance Simulation*, 9(4):431–447.

Bally, V., Pages, G., et al. (2003). A quantization algorithm for solving multidimensional discrete-time optimal stopping problems. *Bernoulli*, 9(6):1003–1049.

Barty, K., Carpentier, P., Cohen, G., and Girardeau, P. (2010a). Price decomposition in large-scale stochastic optimal control. arXiv:1012.2092.

Barty, K., Carpentier, P., and Girardeau, P. (2010b). Decomposition of large-scale stochastic optimal control problems. *RAIRO Operations Research*, 44(3):167–183.

Barty, K., Roy, J.-S., and Strugarek, C. (2009). A stochastic gradient type algorithm for closed-loop problems. *Mathematical Programming, Series A*, 119(1):51–78.

Baucke, R., Downward, A., and Zakeri, G. (2017). A deterministic algorithm for solving multistage stochastic programming problems. *Optimization Online*.

Bauschke, H. H., Combettes, P. L., et al. (2017). *Convex analysis and monotone operator theory in Hilbert spaces*, volume 2011. Springer.

Beeker, N., Malisani, P., and Petit, N. (2016). Discrete-time optimal control of electric hot water tank. *Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS)*.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, New Jersey.

Berthou, T. (2013). *Development of building models for load curve forecast and design of energy optimization and load shedding strategies*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris.

Bertsekas, D. P. (2005a). *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, third edition.

Bertsekas, D. P. (2005b). Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334.

Bertsekas, D. P. (2008). Extended monotropic programming and duality. *Journal of optimization theory and applications*, 139(2):209–225.

Bertsekas, D. P. (2012). *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*. Athena Scientific, 4 edition.

Bertsekas, D. P. and Shreve, S. E. (1996). *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, Belmont, Massachusets.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bonabe de Rougé, R. (2018). *Modélisation des solutions de micro-cogénération en vue de leur*

intégration optimale au sein des bâtiments. PhD thesis, Ecole Nationale Supérieure des Mines de Paris.

Borwein, J. and Lewis, A. S. (2010). *Convex analysis and nonlinear optimization: theory and examples*. Springer.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Carpentier, P., Chancelier, J.-P., De Lara, M., and Rigaut, T. (2018a). Time blocks decomposition of multistage stochastic optimization problem. *arXiv:1804.01711*.

Carpentier, P., Chancelier, J.-P., Leclère, V., and Pacaud, F. (2018b). Stochastic decomposition applied to large-scale hydro valleys management. *European Journal of Operational Research*, 270(3):1086–1098.

Carpentier, P., Cohen, C., Culioli, J.-C., and Renaud, A. (1996). Stochastic optimization of unit commitment: a new decomposition framework. *IEEE Transactions on Power Systems*, 11(2):1067–1073.

Carpentier, P. and Cohen, G. (2017). *Décomposition-coordination en optimisation déterministe et stochastique*, volume 81. Springer.

Carpentier, P., Cohen, G., Chancelier, J.-P., and De Lara, M. (2015). *Stochastic Multi-Stage Optimization*, volume 75. Springer.

Cohen, G. (1978). Optimization by Decomposition and Coordination: A Unified Approach. *IEEE Transactions on Automatic Control*, 23:222–232.

Cohen, G. (1980). Auxiliary Problem Principle and decomposition of optimization problems. *Journal of Optimization Theory and Applications*, 32(3):277–305.

Cohen, G. (2004). *Optimisation des Grands Systèmes*. Cours du DEA MMME, Université de Paris I.

Cohen, G. and Culioli, J.-C. (1990). Decomposition Coordination Algorithms for Stochastic Optimization. *SIAM Journal on Control and Optimization*, 28(6):1372–1403.

Cohen, G. and Miara, B. (1990). Optimization with an auxiliary constraint and decomposition. *SIAM Journal on control and optimization*, pages 137–157.

Dallagi, A. (2007). *Méthodes particulaires en commande optimale stochastique*. PhD thesis, Université Panthéon-Sorbonne-Paris I.

De Lara, M., Carpentier, P., Chancelier, J.-P., and Leclère, V. (2014). *Optimization Methods for the Smart Grid*. Conseil Francais de l'Energie.

de Matos, V. L., Philpott, A. B., and Finardi, E. C. (2015a). Improving the performance of stochastic dual dynamic programming. *Journal of Computational and Applied Mathematics*, 290:196–208.

de Matos, V. L., Philpott, A. B., and Finardi, E. C. (2015b). Improving the performance of stochastic dual dynamic programming. *Journal of Computational and Applied Mathematics*, 290:196–208.

Dellacherie, C. and Meyer, P. A. (1975). *Probabilités et potentiel*. Hermann.

Dunning, I., Huchette, J., and Lubin, M. (2017). JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320.

Dupačová, J., Gröwe-Kuska, N., and Römisch, W. (2003). Scenario reduction in stochastic programming. *Mathematical programming*, 95(3):493–511.

Ekeland, I. and Temam, R. (1999). *Convex Analysis and Variational Problems*, volume 28 of *Classics in Applied Mathematics*. SIAM.

Ernst, D., Glavic, M., Capitanescu, F., and Wehenkel, L. (2009). Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):517–529.

Garcia, C. E., Prett, D. M., and Morari, M. (1989). Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348.

Gilbert, J. C. and Jonsson, X. (2007). LIBOPT – An environment for testing solvers on heterogeneous collections of problems. arXiv preprint cs/0703025.

Girardeau, P. (2010). *Résolution de grands problèmes en optimisation stochastique dynamique*. Thèse de doctorat, Université Paris-Est.

Girardeau, P., Leclere, V., and Philpott, A. B. (2014). On the convergence of decomposition methods for multistage stochastic convex programs. *Mathematics of Operations Research*, 40(1):130–145.

Guigues, V. (2016). Convergence analysis of sampling-based decomposition methods for risk-averse multistage stochastic convex programs. *SIAM Journal on Optimization*, 26(4):2468–2494.

Guigues, V. (2017). Dual dynamic programing with cut selection: Convergence proof and numerical experiments. *European Journal of Operational Research*, 258(1):47–57.

Gurobi Optimization Inc (2014). Gurobi Optimizer Reference Manual.

Haessig, P. (2014). *Dimensionnement et gestion d'un stockage d'énergie pour l'atténuation des incertitudes de production éolienne*. PhD thesis, Ecole normale supérieure de Cachan.

Haessig, P., Ahmed, H. B., and Multon, B. (2015). Energy storage control with aging limitation. In *PowerTech, 2015 IEEE Eindhoven*, pages 1–6. IEEE.

Heitsch, H. and Römisch, W. (2003). Scenario reduction algorithms in stochastic programming. *Computational optimization and applications*, 24(2-3):187–206.

Heitsch, H. and Römisch, W. (2009). Scenario tree modelling for multistage stochastic programs. *Mathematical Programming*, 118:371–406.

Heitsch, H., Römisch, W., and Strugarek, C. (2006). Stability of multistage stochastic programs. *SIAM Journal on Optimization*, 17:511–525.

Heymann, B., Bonnans, J. F., Martinon, P., Silva, F. J., Lanas, F., and Jiménez-Estévez, G. (2015). Continuous optimal control approaches to microgrid energy management. *Energy Systems*, pages 1–19.

Heymann, B., Bonnans, J. F., Silva, F., and Jimenez, G. (2016). A stochastic continuous time model for microgrid energy management. In *European Control Conference (ECC)*, pages 2084–2089. IEEE.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (2012). *Fundamentals of convex analysis*. Springer Science & Business Media.

Homem-de Mello, T., De Matos, V. L., and Finardi, E. C. (2011). Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling. *Energy Systems*, 2(1):1–31.

Houwing, M., Negenborn, R. R., and De Schutter, B. (2011). Demand response with micro-CHP systems. *Proceedings of the IEEE*, 99(1):200–213.

Journal Officiel (2013). Arrêté du 28 décembre 2012 relatif aux caractéristiques thermiques et aux exigences de performance énergétique des bâtiments nouveaux.

Kall, P., Wallace, S. W., and Kall, P. (1994). *Stochastic programming*. Springer.

Kargarian, A., Mohammadi, J., Guo, J., Chakrabarti, S., Barati, M., Hug, G., Kar, S., and Baldick, R. (2016). Toward distributed/decentralized DC optimal power flow implementation in future electric power systems. *IEEE Transactions on Smart Grid*.

Kasten, F. and Young, A. T. (1989). Revised optical air mass tables and approximation formula. *Applied optics*, 28(22):4735–4738.

Kuhn, D., Wiesemann, W., and Georghiou, A. (2011). Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, 130(1):177–209.

Lamoudi, M. Y. (2012). *Distributed Model Predictive Control for energy management in buildings*. PhD thesis, Université de Grenoble.

Leclère, V. (2014). *Contributions to decomposition methods in stochastic optimization*. PhD thesis, Université Paris Est.

Leclère, V., Carpentier, P., Chancelier, J.-P., Lenoir, A., and Pacaud, F. (2018). Exact converging bounds for stochastic dual dynamic programming via fenchel duality. *Optimization Online*.

Lenoir, A. and Mahey, P. (2017). A survey on operator splitting and decomposition of convex programs. *RAIRO Operations Research*, 51(1):17–41.

Liu, P., Fu, Y., and Kargarian marvasti, A. (2014). Multi-stage stochastic optimal operation of energy-efficient building with combined heat and power system. *Electric Power Components and Systems*, 42(3-4):327–338.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137.

Loève, M. (1977). *Probability Theory I*. Springer Science & Business Media, New York, fourth edition.

Löhndorf, N. and Shapiro, A. (2017). Modeling time-dependent randomness in Stochastic Dual Dynamic Programming. *Optimization Online*.

Maceira, M. E. P. and Damazio, J. M. (2006). Use of PAR(p) model in the stochastic dual dynamic programming optimization scheme used used in the operation planning of the Brazilian hydropower system. *Probability in the Engineering and Informational Sciences*, 20:143–156.

Mahey, P., Koko, J., and Lenoir, A. (2017). Decomposition methods for a spatial model for long-term energy pricing problem. *Mathematical Methods of Operations Research*, 85(1):137–153.

Malisani, P. (2012). *Pilotage dynamique de l'énergie du bâtiment par commande optimale sous contraintes utilisant la pénalisation intérieure*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris.

Mataoui, M. (1990). *Contributions à la décomposition et à l'agrégation des problèmes variationnels*. Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris.

Mohammadi, S., Soleymani, S., and Mozafari, B. (2014). Scenario-based stochastic operation management of microgrid including wind, photovoltaic, micro-turbine, fuel cell and energy storage devices. *International Journal of Electrical Power & Energy Systems*, 54:525–535.

Morales, J. M., Conejo, A. J., Madsen, H., Pinson, P., and Zugno, M. (2013). *Integrating renewables in electricity markets: operational problems*, volume 205. Springer Science & Business Media.

Noorian, A. M., Moradi, I., and Kamali, G. A. (2008). Evaluation of 12 models to estimate hourly diffuse irradiation on inclined surfaces. *Renewable energy*, 33(6):1406–1412.

Oldewurtel, F. (2011). *Stochastic model predictive control for energy efficient building climate control*. PhD thesis, Eidgenössische Technische Hochschule ETH Zürich.

Oldewurtel, F., Parisio, A., Jones, C. N., Gyalistras, D., Gwerder, M., Stauch, V., Lehmann, B., and Morari, M. (2012). Use of model predictive control and weather forecasts for energy efficient building climate control. *Energy and Buildings*, 45:15–27.

Olivares, D. E., Cañizares, C. A., and Kazerani, M. (2011). A centralized optimal energy management system for microgrids. In *Power and Energy Society General Meeting, 2011 IEEE*, pages 1–6. IEEE.

Olivares, D. E., Lara, J. D., Cañizares, C. A., and Kazerani, M. (2015). Stochastic-predictive energy management system for isolated microgrids. *IEEE Transactions on Smart Grid*, 6(6):2681–2693.

Olivares, D. E., Mehrizi-Sani, A., Etemadi, A. H., Canizares, C., Iravani, R., Kazerani, M., Hajimiragha, A. H., Gomis-Bellmunt, O., Saeedifard, M., Palma-Behnke, R., et al. (2014). Trends in microgrid control. *IEEE Transactions on Smart Grid*, 5(4):1905–1919.

Page, J., Robinson, D., Morel, N., and Scartezzini, J.-L. (2008). A generalised stochastic model for the simulation of occupant presence. *Energy and buildings*, 40(2):83–98.

Paridari, K., Parisio, A., Sandberg, H., and Johansson, K. H. (2016). Robust scheduling of smart appliances in active apartments with user behavior uncertainty. *IEEE Transactions on Automation Science and Engineering*, 13(1):247–259.

Parisio, A., Wiezorek, C., Kyntaja, T., Elo, J., and H., J. K. (2015). An MPC-based energy management system for multiple residential microgrids. In *IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE.

Pereira, M. V. and Pinto, L. M. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375.

Pflaum, P., Alamir, M., and Yacine Lamoudi, M. (2014). Comparison of a primal and a dual decomposition for distributed MPC in smart districts. *5th IEEE International Conference on Smart Grid Communications*.

Pflug, G. C. and Pichler, A. (2014). *Multistage Stochastic Optimization*. Springer.

Philpott, A., de Matos, V., and Finardi, E. (2013). On solving multistage stochastic programs with coherent risk measures. *Operations Research*, 61(4):957–970.

Philpott, A. B. and De Matos, V. L. (2012). Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research*, 218(2):470–483.

Philpott, A. B. and Guan, Z. (2008). On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4):450–455.

Pinson, P., Baroche, T., Moret, F., Sousa, T., Sorin, E., and You, S. (2018). The emergence of consumer-centric electricity markets. http://pierrepinson.com/docs/pinsonetal17consumercentric.pdf.

Pontryagin, L. S. (1962). *Mathematical theory of optimal processes*. Routledge.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

Powell, W. B. (2014). Clearing the jungle of stochastic optimization. *Informs*, pages 109–137.

Puterman, M. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York.

Rockafellar, R. T. (1968). Integrals which are convex functionals. *Pacific Journal of Mathematics*, 24(3):525–539.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press.

Rockafellar, R. T. (1971). Integrals which are convex functionals, II. *Pacific Journal of Mathematics*, 39(2):439–469.

Rockafellar, R. T. (1974). *Conjugate duality and optimization*, volume 16. Siam.

Rockafellar, R. T. and Wets, R. J. (1976). Stochastic convex programming: relatively complete recourse and induced feasibility. *SIAM Journal on Control and Optimization*, 14(3):574–589.

Rockafellar, R. T. and Wets, R. J.-B. (1991a). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147.

Rockafellar, R. T. and Wets, R. J.-B. (1991b). Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.*, 16(1):119–147.

Rockafellar, R. T. and Wets, R. J.-B. (1998). *Variational Analysis*. Springer Science & Business Media, Berlin.

Rujeerapaiboon, N., Schindler, K., Kuhn, D., and Wiesemann, W. (2017). Scenario reduction revisited: Fundamental limits and guarantees. *arXiv preprint arXiv:1701.04072*.

Ruszczyński, A. (1997). Decomposition methods in stochastic programming. *Mathematical programming*, 79(1):333–353.

Ruszczyński, A. and Shapiro, A., editors (2003). *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier.

Schütz, T., Streblow, R., and Müller, D. (2015). A comparison of thermal energy storage models for building energy system optimization. *Energy and Buildings*, 93:23–31.

Shapiro, A. (2006). On complexity of multistage stochastic programs. *Operations Research Letters*, 34:1–8.

Shapiro, A. (2011). Analysis of Stochastic Dual Dynamic Programming Method. *European Journal of Operational Research*, 209:63–72.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2009). *Lectures on stochastic programming: modeling and theory*. SIAM.

Shapiro, A., Tekaya, W., da Costa, J. P., and Soares, M. P. (2012). Final report for technical cooperation between georgia institute of technology and ons–operador nacional do sistema elétrico. *Georgia Tech ISyE Report*.

Strugarek, C. (2006). *Approches variationnelles et autres contributions en optimisation stochas-*

tique. Thèse de doctorat, École Nationale des Ponts et Chaussées.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Takriti, S., Birge, J. R., and Long, E. (1996). A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems*, 11(3):1497–1508.

Thomas, B. (2008). Benchmark testing of Micro-CHP units. *Applied Thermal Engineering*, 28(16):2049–2054.

Tsitsiklis, J. N. and Van Roy, B. (1996). Feature-based methods for large-scale dynamic programming. *Machine Learning*, 22:59–94.

Van Ackooij, W., de Oliveira, W., and Song, Y. (2017). On regularization with normal solutions in decomposition methods for multistage stochastic programming. *Optimization Online*.

Van Slyke, R. M. and Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.

Vezolle, P., Vialle, S., and Warin, X. (2009). Large Scale Experiment and Optimization of a Distributed Stochastic Control Algorithm. Application to Energy Management Problems. In *International Workshop on Large-Scale Parallel Processing (LSPP 2009)*, Rome, Italy.

Wallace, S. W. and Fleten, S.-E. (2003). Stochastic programming models in energy. *Handbooks in operations research and management science*, 10:637–677.

Wets, R. J. (2002). Stochastic programming models: wait-and-see versus here-and-now. In *Decision Making Under Uncertainty*, pages 1–15. Springer.

Widén, J. and Wäckelgård, E. (2010). A high-resolution stochastic model of domestic activity patterns and electricity demand. *Applied Energy*, 87(6):1880–1892.

Wytock, M., Moehle, N., and Boyd, S. (2017). Dynamic energy management with scenario-based robust MPC. In *American Control Conference (ACC), 2017*, pages 2042–2047. IEEE.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

Zou, J., Ahmed, S., and Sun, X. A. (2017). Stochastic dual dynamic integer programming. *Mathematical Programming*, pages 1–42.