Computational Complexity of Multi-way, Dataflow Constraint Problems

Gilles Trombettoni, Bertrand Neveu

Rapport de Recherche CERMICS 97.86

Janvier 97

Computational Complexity of Multi-way, Dataflow Constraint Problems

Gilles Trombettoni, Bertrand Neveu

Abstract

Although it is widely acknowledged that multi-way dataflow constraints make it easier to specify certain relationships in interactive applications, concerns about their efficiency have impeded their acceptance. Some of the local propagation algorithms that solve these constraints are polynomial, others (such as SkyBlue) are exponential. In fact, every system handles a specific problem and the influence of any particular restriction on the computational complexity is not yet precisely determined. In this report, we present theoretical results that allow us to classify existing multi-way constraint problems. We especially prove that the problem handled by SkyBlue is NP-hard.

Keywords: Constraints, Computational Complexity, Local Propagation, User Interfaces.

Complexité théorique de problèmes de contraintes fonctionnelles multi-directionnelles

Résumé

Les contraintes fonctionnelles multi-directionnelles permettent de spécifier facilement certaines relations dans les applications graphiques interactives. Elles sont pourtant encore peu utilisées à cause de doutes concernant l'efficacité des techniques de résolution par propagation locale : certains algorithmes de résolution par propagation locale sont polynomiaux, d'autres (comme SkyBlue) sont exponentiels. En fait, chaque système existant traite un problème spécifique et les conséquences des diverses restrictions dans le formalisme des contraintes traitées sur la complexité théorique ne sont pas déterminées précisément. Ce rapport présente des résultats de complexité théorique qui permettent de classifier les différents problèmes traités par les systèmes existants. Nous montrons en particulier que le problème traité par SkyBlue est NP-difficile.

Mots-clés: Contraintes, Complexité théorique, Propagation locale, Interfaces graphiques.

1 Introduction

Dataflow constraints are rapidly gaining popularity in interactive applications because they simplify the programming task, are conceptually simple and easy to understand, and are capable of expressing relationships over multiple data types, including numbers, strings, booleans, bitmaps, fonts and colors. The dataflow constraint solvers are used in numerous interactive systems, such as graphical interface toolkits, spreadsheets, graphical layout systems and animations.

Dataflow constraints are divided into two main categories. A *one-way* dataflow constraint is associated to a function that recovers the consistency of the constraint by calculating output variables based on the current value of the input variables. The *spreadsheet model*, more formally known as the *dependency graph* model [Hoover 87], only takes into account one-way constraints. This model is widely used in interactive systems, mainly because the solving process is based on an efficient *incremental evaluation* phase that topologically sorts the functions to execute.

A *multi-way* dataflow constraint has *several* functions (called *methods*) that may be used to satisfy it. The solving process of problems that contain multi-way constraints needs an additional *planning phase* that occurs before the evaluation phase. The planning phase assigns one method to each constraint.

Although multi-way constraints are more expressive than one-way constraints, concerns about their efficiency have made them less popular.

Every solving algorithm handles a specific constraint planning problem and the conditions that allow us to decide whether it is computationally difficult are not clear by now.

This report aims at giving a classification for the computational complexity of the main existing multi-way constraint problems.

2 Background

A *multi-way dataflow constraint* system is denoted as (V, C, M). V is a set of variables that all have a current value. C is a set of dataflow constraints and M is a set of *methods* that can satisfy the constraints.

Definition 1 *A* **multi-way dataflow constraint** *is an equation that has one or more methods associated with it that may be used to satisfy the equation.*

A **method** *consists of zero or more inputs, one or more outputs, and an arbitrary piece of code that computes the output variables based on the current value of the input variables. A* **single-output** *method determines only one variable.*

A (dataflow) constraint system is commonly represented by a *constraint graph* G_c as shown in figure 1 (a).

Local propagation of multi-way constraint systems works in two phases:

- The *planning phase* directs the edges in G_c by assigning one method to each constraint. The result of this phase (*i.e.*, the solution of the corresponding problem) is a *valid* graph G_m called *method graph* (see figure 1 (b) and (c)).

  Definition 2 *A method graph G_m is* **valid** *if (1) every constraint has one method associated with it in G_m, and (2) G_m contains no* **variable conflict***, that is, each variable is the output of at most one method* (i.e., *has at most one incoming edge).*

- When the method graph G_m contains no directed cycles, the *evaluation phase* executes the methods in some topological order. When a method is executed, it sets the output variables to values such that the constraint is satisfied. When G_m is cyclic, strongly connected components are collected and generally passed to external solvers to be satisfied as a whole.

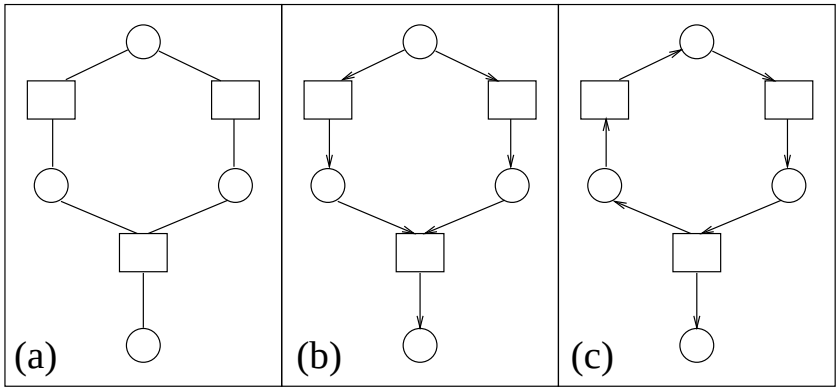

Figure 1: A *constraint graph* is a bipartite graph which nodes are constraints and variables, respectively represented by rectangles and circles. Each constraint is connected to its variables. Figure (a) shows an example. Figures (b) and (c) show two possible *method graphs*. The method selected for each constraint is symbolized by directed edges from the constraint to the output variables, and from the input variables to the constraint. The method graph in (b) contains no directed cycles, contrarily to the method graph in (c).

Ignoring the operations involved in method execution and cycle solving, the evaluation problem is in the class P of polynomial problems. Indeed, topological sort is $O(d \times |C|)$ where d is the maximum number of methods associated to one constraint. This report deals with the computational complexity of the problem solved by the planning phase that will be called *constraint planning problem* in the following.

Constraint planning algorithms can be divided into three main categories:
(1) DeltaBlue [Freeman-Benson 90] and SkyBlue [Sannella 94] work by propagating the conflicts from the perturbations to the leaves of the constraint graph. (2) The propagation of degrees of freedom scheme (in short PDOF) selects the methods in the reverse order (selecting first the methods to be executed last). It has been used in SketchPad [Sutherland 63] and QuickPlan [Vander Zanden 96]. (3) A third approach is related to the classical problem of graph matching. It gives the *Maximum-matching* algorithm [Gangnet 92].

3 Types of constraint planning problems

Existing local propagation algorithms solve different planning problems that imply various tradeoffs between simplicity and power:

- *required* constraints only, or both required and *preferential* constraints that are satisfied if possible

- single-output constraints only, or both single-output and multi-output ones
- acyclic constraint graphs only, or cycles allowed
- *method restriction* imposed or relaxed (see definition 3 below)

problem	method restriction	single-output	complexity	proof	algorithms
acp_1	yes	yes	P	[Sutherland 63]	PDOF, DeltaBlue
acp_2	yes	no	P	[Vander Zanden 96]	QuickPlan
acp_3	no	yes	NPC	[Maloney 91] 3	–
acp_4	no	no	NPC	[Maloney 91] 1 and 2	–
cp_1	yes	yes	P	[Gangnet 92]	Maximum-matching
cp_2	**yes**	**no**	**??**	–	**SkyBlue**
cp_3	**no**	**yes**	**??**	–	–
cp_4	**no**	**no**	**??**	–	–

Table 1: Computational complexity of constraint planning problems. Cycles in the constraint graph are allowed. Constraints are *required* (not *preferential*). Every problem depends on three characteristics: (1) a problem that only accepts *a*cyclic method graphs is designed by acp_i ($i \in \{1..4\}$), whereas a problem that accepts both acyclic and cyclic solutions is designed by cp_i; (2) the method restriction; (3) the presence of single-output constraints only. A *yes* in a cell indicates that the corresponding restriction is imposed. NPC stands for NP-complete. The computational complexity of problems cp_2, cp_3 and cp_4 is not known by now, especially cp_2 that is handled by SkyBlue.

Moreover, some systems forbid directed cycles in the method graph, whereas others do not, which leads in fact to two different (and incomparable) problems. Indeed, a general computational result states that a restriction imposed on a solution does not necessarily make the corresponding problem easier [Papadimitriou 94].

Table 1 shows the computational complexity of some existing constraint planning problems.

4 An NP-complete planning problem with method restriction

Definition 3 *The* **method restriction** *imposes that every constraint method of a given problem must use all of the variables in the constraint either as an input or an output variable.*

Disjunctive constraints do not respect the method restriction (*e.g.*, constraint $a \vee b$ has two methods that output to either variable a or b with no input). However, these constraints are not usually needed in interactive systems. Therefore, all of the propagation algorithms impose the method restriction. The problems handled by these algorithms are in P, except cp_2 (see table 1) that has not been yet analyzed.

The only known NP-completeness results aim at problems acp_3 and acp_4 [Maloney 91] that are not very interesting in practice because they relax the method restriction.

The following theorem states an NP-completeness result about the problem cp_2 handled by SkyBlue, and for which the method restriction holds.

Theorem 1 *Let G be a dataflow constraint system for which the method restriction holds.*

Then proving the existence of a valid method graph (cyclic or not) corresponding to G is NP-complete.

The proof and the polynomial reduction involved in it are described in the two following paragraphs.

4.1 Polynomial reduction

We will prove that the known NP-complete problem "*Exact Cover by 3-Sets*" [Papadimitriou 94] can be reduced to "*constraint-planning*" (*i.e.*, cp_2). This reduction will be called *planning reduction* in the following.

Definition 4 (Exact Cover by 3-Sets) *Let X be a finite set, such that $|X| = 3q$ for some integer q. Let E be a family of sets that contain 3 elements of X each. Every element of X belongs to at least one 3-set of E.*[1]

Does E contain an exact cover for X, that is, a subset S of E such that every element of X belongs to exactly one 3-set of S?

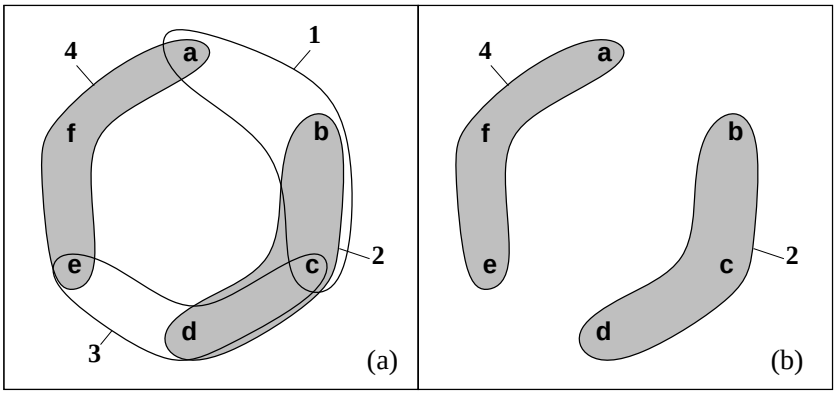

Figure 2: An instance of the "*Exact Cover by 3-Sets*" problem. $X = \{a...f\}$. The 3-sets in $E = \{1...4\}$ are represented by hyper-arcs, as shown in (a). The unique solution is shown in (b).

Let $G = (X, E)$ be an instance of "*Exact Cover by 3-Sets*". Let $G' = (V, C, M)$ be an instance of "*constraint-planning*" obtained by a planning reduction applied to G as follows:

- Variables of V are divided into two sets *VarX* and *VarE*. Constraints of C are divided into two sets *ConstX* and *ConstE*.
- Each element i of X corresponds to one variable *i-at-most* of *VarX*.
- A 3-set $p = \{a, b, c\}$ in E corresponds to a constraint set_p of *ConstE* connecting six variables. set_p has two triple output methods that indicate whether the 3-set p is either *present* or *absent* in the solution. The present method outputs to the three variables *a-at-most*, *b-at-most*, *c-at-most* of *VarX* (It ensures that no other 3-set will cover the corresponding elements.) The absent method outputs to the three variables *a-p*, *b-p* and *c-p* of *VarE*.

[1] This additional hypothesis discards trivial instances while keeping the problem NP-complete.

- When element i of X can be covered by n different 3-sets $\{p_1...p_n\}$ of E, one constructs n variables $\{i\text{-}p_1...i\text{-}p_n\}$ of *VarE*. One also builds one constraint *i-at-least* of *ConstX* that connects these variables. *i-at-least* has n single-output methods, one for each variable in the constraint. The method that outputs to variable $i\text{-}p_k$ ensures that element i of X is covered by (at least) the 3-set p_k.

Figure 2 shows an instance of "*Exact Cover by 3-Sets*". It is reduced to the "*constraint-planning*" instance of figure 3.

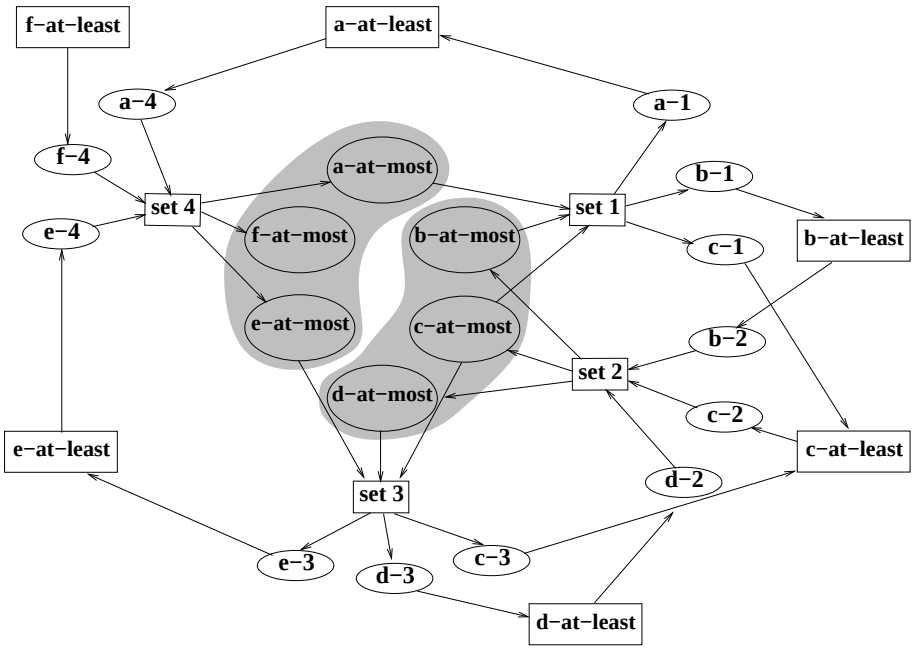

Figure 3: Valid method graph after transforming the instance of "*Exact Cover by 3-Sets*" given in figure 2.

The planning reduction is based on the following intuition. One element i of X appears in exactly one 3-set of solution S. Thus, i appears *at most once* and *at least once* in a 3-set of S. This is translated into the planning problem as follows:

- (at most once) If the 3-set p belongs to solution S, then the *present* method of set_p is selected in the corresponding constraint planning problem. Thus, the variables determined by set_p ensure that no other 3-set than set_p will cover them, otherwise it would lead to variable conflicts.
- (at least once) As said above, a constraint *i-at-least* directed onto variable $i\text{-}p$ ensures that element i of X is covered by the 3-set p in the solution. In fact, no method can be selected for constraint *i-at-least* if *every* connected variable is the output of an *absent* method.

4.2 Proof of theorem 1

First, the planning reduction is $O(|X|+|E|)$. Indeed, one element of E corresponds to one constraint, five methods, nine edges and three variables in the planning problem; one element of X corresponds

to one constraint and one variable.

Second, "*constraint-planning*" is in *NP* since verifying that a method graph issued from a planning reduction is valid is $O(|V| + |C|)$. Indeed, one just has to verify that every variable is determined by at most one constraint, and that every constraint has one method selected for it.

Finally, the two following paragraphs prove the equivalence between (i) a solution S for the problem G of "*Exact Cover by 3-Sets*" and (ii) a solution S' for the planning problem G' obtained by the planning reduction applied to G.

$(i) \rightarrow (ii)$ S' is obtained by selecting present methods for the constraints set_p in *ConstE* when the 3-set p is in S. The absent method is selected for the 3-sets that are not in S. Every constraint *i-at-least* in *ConstX* is directed onto variable *i-p* of *VarE* when the 3-set p is in S. *i-p* is an input of the present method of constraint set_p in *ConstE*. By construction, every constraint has one method selected for it.

By hypothesis, every element of X belongs to exactly one 3-set of S. Since there is no intersection between two any 3-sets in S, this construction does not generate any conflict on variables *i-at-most* of *VarX*.

Every element i of X is covered by (at least) one 3-set p of S. By construction, set_p is activated with the present method that outputs to *i-at-most*. By construction, variable *i-p* is determined by constraint *i-at-least*, thus generating no variable conflict. Since the other variables of constraint *i-at-least* are not determined by it and are linked to exactly two constraints, they cannot provide variable conflicts. Thus, for every element i of X, no corresponding variable in the constraint planning problem can cause a variable conflict.

$(ii) \rightarrow (i)$ Based on S', S is built by collecting a 3-set $\{a, b, c\}$ when the present method that outputs to variables *a-at-most*, *b-at-most*, and *c-at-most* is selected. Since the method graph is valid, the intersection of any two 3-sets in S is empty.

Let us consider every constraint *i-at-least* of *ConstX* in S'. Let *i-p* be the variable determined by *i-at-least*. *i-p* is necessarily an input variable of constraint set_p that determines variable *i-at-most*, otherwise a variable conflict would occur on *i-p*. By construction, i necessarily belongs to a 3-set in S. □

4.3 Complexity of 2-output constraint planning problems

We know that when "*constraint-planning*" is restricted to single-output constraints, the problem complexity comes down to P (cp_1). Our previous planning reduction shows that "*constraint-planning*" is NP-complete with 3-output constraints. A natural question is therefore whether the 2-output constraint restriction would yield a polynomial problem or not.

Theorem 2 *Let G be a dataflow constraint system for which the method restriction holds. G contains methods that have at most two outputs.*

Then proving the existence of a valid method graph (cyclic or not) corresponding to G is NP-complete.

Proof Every 3-output constraint set_p can easily be transformed into two 2-output constraints and a "dummy" variable, as shown in figure 4. The global behavior is exactly the same. □

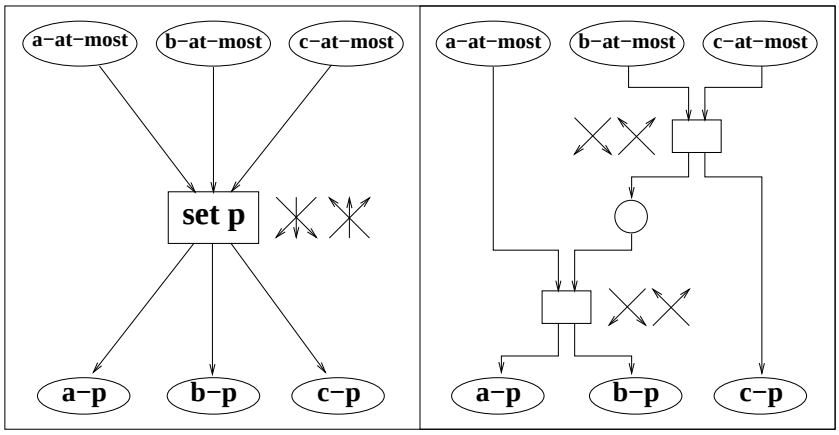

Figure 4: A 3-output constraint set_p transformed into two 2-output constraints. The small diagrams close to a constraint indicate the possible methods.

5 Influence of the method restriction on problems with cyclic solutions

Theorem 3 *Let C be a class of dataflow constraint systems and let P_C be the problem of existence of a valid method graph (cyclic or not) for any instance in the class C. Let P'_C be the restriction of P_C to constraint systems that satisfy the method restriction.*

Then P_C and P'_C are polynomially (actually LOG-space) equivalent.

In other words, when cyclic solutions are allowed, the method restriction has no influence on the computational complexity of the constraint planning problem.

The proof is based on the *method transformation* defined as follows:

Definition 5 *Let $G_1 = (V_1, C_1, M_1)$ be a constraint system. Based on G_1, the* **method transformation** *provides a constraint system $G_2 = (V_2, C_2, M_2)$ such that: (1) $V_1 = V_2$, (2) $C_1 = C_2$, (3) methods in M_1 for which the method restriction holds, occur unchanged in M_2, and (4) every method m_1 in M_1 for which the method restriction does not hold is replaced by a method m_2 in M_2 for which the method restriction holds: m_2 has the same output variables as m_1 and has* all *of the other variables of the associated constraint as input.*

Note that this trivial transformation is LOG-space. Thus, theorem 3 can be applied to problems that are in P or are NP-complete.

Proof of theorem 3 First, P'_C can be reduced to P_C since P'_C is a restriction of P_C. Second, P_C can be reduced to P'_C thanks to the method transformation that reduces a constraint system G_1 into a constraint system G_2 for which the method restriction holds. We prove the equivalence between a solution of G_1 (i) and a solution of G_2 (ii).

$(ii) \rightarrow (i)$ A valid method graph of G_2 can be transformed into a valid method graph of G_1 since G_1 is G_2, except that some edges are omitted, which cannot induce variable conflicts.

$(i) \rightarrow (ii)$ A valid method graph of G_1 can be transformed into a valid method graph of G_2 since every edge added by the method transformation connects a constraint and one of its *input* variable, which cannot generate variable conflicts. □

Note that if directed cycles were forbidden in the solution, this latest implication would be false because adding an input edge could introduce a directed cycle.

6 Synthesis

problem	method restriction	single-output	complexity	proof	algorithms
acp_1	yes	yes	P	[Sutherland 63]	PDOF, DeltaBlue
acp_2	yes	no	P	[Vander Zanden 96]	QuickPlan
acp_3	no	yes	NPC	[Maloney 91] 3	–
acp_4	no	no	NPC	[Maloney 91] 1 and 2	–
cp_1	yes	yes	P	[Gangnet 92]	Maximum-matching
cp_2	**yes**	**no**	***NPC***	**theorems 1 and 2**	**SkyBlue**
cp_3	**no**	**yes**	***P***	**theorem 3 and cp_1**	**Maximum-matching**
cp_4	**no**	**no**	***NPC***	**cp_2**	–

Table 2: Computational complexity of constraint planning problems. The contributions of this work are bold-faced.

These three theorems allow us to deduce the three missing computational complexity results, as shown in table 2. Since cp_2 (that is NP-complete) is a restriction of cp_4 and cp_4 is in NP, cp_4 is also NP-complete[2]. Theorem 3 proves that cp_1 and cp_3 have the same computational complexity.

Since the table is now complete, we can highlight interesting points about the constraint planning problems handled by existing algorithms.

SkyBlue Problem cp_2 assumes that the constraints must be *required*, whereas SkyBlue [Sannella 94] can handle a kind of *preferential* constraints. Therefore, the SkyBlue problem is NP-hard. The NP-completeness result given by theorem 1 makes the exponential worst case time complexity of SkyBlue less surprising. However, [Sannella 94] has proved that SkyBlue could reach this worst case complexity even on problems acp_2 and cp_1 that are in P.

QuickPlan QuickPlan [Vander Zanden 96] cannot be extended without making the corresponding problem NP-complete. Indeed, the gap between acp_2 (in P) and acp_4 (NP-complete) is due to the method restriction, as mentioned in [Vander Zanden 96]. Moreover, the gap between acp_2 (in P) and cp_2 (NP-complete) also lies in the fact that cyclic solutions are forbidden or not.

Note that if the following proposition was false, we would have proved that P would be equal to NP.

Proposition 1 *There exist instances of* "constraint-planning" *issued from the planning reduction which only have cyclic solutions.*

[2]Theorem 3 applied to cp_2 and cp_4 also proves this result.

Proposition 1 can be proved as follows. Let us consider an instance $G = (X, E)$ of "*Exact Cover by 3-Set*" for which at least two elements of E intersect. Let G' be the instance of "*constraint-planning*" issued from the planning reduction applied to G. Let GM be a valid method graph that corresponds to G'. We prove that GM contains at least one directed cycle.

Indeed, let us consider an element i that belongs to two 3-sets of E. Since GM is valid, a method is selected for contraint *i-at-least* that outputs to a variable *i-p*. Since there is no variable conflict, variable *i-p* is an input variable of a present method of the constraint set_p that outputs to variable *i-at-most*. For the same reason, variable *i-at-most* is an input variable of a constraint set_q (set_q exists by construction) that outputs to a variable *i-q*, which leads to a directed cycle on contraint *i-at-least*. □

Maximum-matching The Maximum-matching problem is in P [Gangnet 92]. Since the gap between cp_1 and cp_2 lies in the *single-output constraint* restriction, Maximum-matching cannot be extended to multi-output constraints.

Note that Maximum-matching can also solve problem cp_3. Indeed, theorem 3 can easily be extended to the problem of *finding* a solution, thanks to a reverse transformation. So one needs to (1) transform an instance of cp_3 into one of cp_1 with the *method transformation*, (2) call Maximum-matching on the cp_1 instance and (3) retrieve the solution (if any) with the reverse method transformation.

7 Complexity of other constraint planning problems

Table 2 has given the computational complexity of constraints problems for which:

- the constraints are *required* (not *preferential*), and
- the constraint graph may be cyclic or not.

Based on the previous results, paragraphs 7.2 and 7.3 analyse the influence of these two characteristics on the computational complexity. The restriction to a widely known type of single-output constraints is also discussed in the following paragraph.

7.1 Regular constraints

Interactive applications usually enclose a large part of regular constraints, such as linear or quadratic equations.

Definition 6 *A* **regular constraint** *connecting n variables is a multi-way single-output constraint that contains n single-ouput methods that output to every variable.*

We can observe that problems cp_1 and acp_1 are both in P. Thus, the restrictions of cp_1 and acp_1 to problems that only contain regular constraints are also in P.

Remark Maximum-matching is restricted to regular constraints. However, a minor improvement could allow it to handle any type of single-output constraints by ensuring that the latest chosen method in the alterning path can actually be "inverted"[3]. Thus, the computational complexity of problem cp_1 has in fact been deduced thanks to this improvement.

7.2 Complexity of constraint hierarchies

Given a problem with both preferential and required constraints, if not all of the preferential constraints can be satisfied, algorithms need a way to select which solutions are desired.

In existing local propagation systems that handle preferential constraints, required and preferential constraints are partitioned in an arbitrary number of levels of preference called *constraint hierarchies*, each successive level being less preferred than the previous one. Moreover, the best solution is always determined with respect to the same *generic-LGB* comparator.[4] In short, a valid method graph is *generic-LGB* iff:

- all of the required constraints are satisfied, and
- one cannot satisfy a desactived preferential constraint in the method graph without desactivating a stronger constraint or a constraint with the same strength[5].

See [Borning 92] for more details.

Let $hacp_i$ and hcp_i ($i \in \{1..4\}$) be the respective generalizations of problems acp_i and cp_i for which a solution is expected that satisfies the generic-LGB criterion in a constraint hierarchy.

Since table 2 is now complete, we can observe that all of the constraint planning algorithms able to handle constraint planning problems in P (*i.e.*, acp_1, acp_2, cp_1, and cp_3) can also support constraint hierarchies with the generic-LGB criterion, while remaining polynomial.

Then, we can conclude that the problems $hacp_1$, $hacp_2$, hcp_1 and hcp_3 are in P.

We can also remark that, as the problems acp_3, acp_4, cp_2, cp_4, are NP-complete with required constraints, their respective generalizations $hacp_3$, $hacp_4$, hcp_2, hcp_4 are NP-hard.

7.3 Complexity of problems with acyclic constraint graphs

We know that an acyclic constraint graph cannot yield a method graph with directed cycles. The restriction of the two problems acp_i and cp_i ($i \in \{1..4\}$) to acyclic constraints graphs is then a unique problem p'_i.

- Since p'_1 and p'_2 are restrictions of problems acp_1 and acp_2, they are also in P.
- In the same way, p'_3 is in P since it is a restriction of cp_3.
- p'_2 and p'_4 can be seen as problems where cyclic solutions are allowed (in fact, all solutions are acyclic and one does not need to forbid cyclic solutions). They satisfy the conditions of theorem 3. Thus, p'_4 is in P since it has the same complexity as p'_2.

We can then conclude that all of the restrictions to acyclic constraint graphs are in P.

[3] This improvement is very close to the way of calculating the *walkabout strengths* in DeltaBlue.

[4] This criterion is called *generic* since it can be applied to any constraint planning problem (with the single-output constraint restriction or not, accepting cyclic solutions or not...)

[5] It is the well-known LGB comparator in DeltaBlue and QuickPlan, the l.p.b.-predicate one in Maximum-matching and the MGB one in SkyBlue.

Constraint hierarchies

Let us call hp'_i the restriction to acyclic constraint graphs of the two problems hcp_i and $hacp_i$ (see paragraph 7.2).

The problems hp'_1 and hp'_2 are in P because they are restrictions of the problems $hacp_1$ and $hacp_2$. Moreover, these problems can be solved by QuickPlan, so that the problems hp'_3 and hp'_4 can also be solved by Quickplan thanks to the theorem 3 extended to the problem of finding a solution (see paragraph 6). Thus, the problems hp'_3 and hp'_4 are also in P.

8 Conclusion

This report has proved two computational complexity results. First, the constraint planning problem handled by SkyBlue is NP-hard. We do not know yet whether it is in NP, when handling constraint hierarchies with the generic-LGB criterion. Second, when cyclic solutions are allowed, the computational complexity is not sensitive to the method restriction. Based on the theoretical results presented in this report, the following simple rule gives sufficient conditions to determine if a given constraint planning problem is in P.

---

if the constraint graph contains no cycle **then**
  the problem is in P
else if an acyclic solution is expected **then**
  the problem is in P if the method restriction is imposed
else
  the problem is in P if it only contains single-output constraints

---

This rule highlights the importance of the "cyclic/acyclic solution" condition. When directed cycles are not allowed in the solution, the gap between problems in P and NP-complete ones comes from the method restriction, whereas it is not sensitive to the single-output constraint restriction. It is exactly the opposite behavior when cyclic solutions are allowed. Finally, the polynomial complexity of problem acp_1, acp_2, cp_1, or cp_3 is not lost when handling constraint hierarchies.

We believe that these results will help designers to conceive multi-way constraint systems that provide a good balance between expressiveness and efficiency.

Acknowledgements

We especially want to thank Nicolas Chleq and Thomas Schiex whose comments were very helpful. Also thanks to Christian Bliek and Nicolas Prcovic.

References

[Borning 92] Alan Borning, Bjorn Freeman-Benson, Molly Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5(3):223–270, Sept. 1992.

[Freeman-Benson 90] Bjorn Freeman-Benson, John Maloney, Alan Borning. An incremental constraint solver. *Communications of the ACM*, 33(1):54–63, Jan. 1990.

[Gangnet 92] Michel Gangnet, Burton Rosenberg. Constraint programming and graph algorithms. In *Second International Symposium on Artificial Intelligence and Mathematics*, Jan. 1992.

[Hoover 87] Roger Hoover. *Incremental Graph Evaluation.* PhD thesis, Cornell University, Ithaca, 1987.

[Maloney 91] John Maloney. *Using Constraints for User Interface Construction.* PhD thesis, Department of Computer Science and Engineering, University of Washington, Seattle, 1991. Published as Technical Report 91-08-12.

[Papadimitriou 94] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[Sannella 94] Michael Sannella. *Constraint Satisfaction and Debugging for Interactive User Interfaces.* PhD thesis, Department of Computer Science and Engineering, University of Washington, Seattle, 1994. Also available as Technical Report 94-09-10.

[Sutherland 63] Ivan Sutherland. *Sketchpad: A Man-Machine Graphical Communication System.* PhD thesis, Department of Electrical Engineering, MIT, 1963.

[Vander Zanden 96] Bradley Vander Zanden. An incremental algorithm for satisfying hierarchies of multi-way, dataflow constraints. *ACM Transactions on Programming Languages and Systems*, 18(1):30–72, Jan. 1996.

Liste des derniers rapports de recherche du CERMICS

List of previous CERMICS research reports

96-73	Y. J. Xiao	*Variation of product function and numerical of some partial differential equations by low-discrepancy sequences* 7 pages, août 1996, Noisy-Le-Grand
96-74	B. Lapeyre Y. J. Xiao	*Volume-discrepancy of sequences and numerical tests* 16 pages, septembre 1996, Noisy-Le-Grand
96-75	A. Toubol	*High temperature regime for a multidimensional Sherrigton-Kirkpatrick model of spin glass* 34 pages, septembre 1996, Noisy-Le-Grand
96-76	C. Buisson, J. P.Lebacque, J. B.Lesort	*Travel Times Computation for Dynamic Assignment Modelling* 15 pages, novembre 1996, Noisy-Le-Grand
96-77	E. Cancès, C.Le Bris	*On the perturbation methods for some nonlinear quantum chemistry models* 45 pages, novembre 1996, Noisy-Le-Grand
96-78	S. Depeyre	*Une méthode couplée pour la simulation d'écoulements disphasiques dispersés* 57 pages, octobre 1996, Sophia-Antipolis
96-79	J.F.Gerbeau N.Glinsky-Olivier B.Larrouturou	*Semi-implicit Roe-type fluxes for low-mach number flows* octobre 1996, Sophia-Antipolis
96-80	*B. Jourdain*	*Propagation du chaos trajectorielle pour les lois de conservation scalaire* 14 pages, décembre 1996, Noisy-Le-Grand

96-81	*B. Jourdain*	*Diffusions with a nonlinear drift coefficient and probabilistic interpretation of generalized Burger's equations* 16 pages, décembre 1996, Noisy-Le-Grand
96-82	*D. Hirschkoff*	*Up-to context proofs for the π-calculus in the Coq system* 18 pages, janvier 1997, Noisy-Le-Grand
96-83	*E. V Abrarova,* *A. V. Karapetyan*	*Sur la stabilité et les bifurcations des mouvements stationnaires d'un corps solide dans un champ de gravitation central* 22 pages, janvier 1997, Noisy-Le-Grand
96-84	*E. V Abrarova,*	*Sur les mouvements stationnaires en orbite d'un système de deux corps avec liaison élastique* 20 pages, janvier 1997, Noisy-Le-Grand
97-85	J.P. Cioni	*Parallelisation of Maxwell and 3D simulations in electromagnetism using clusters of workstations* 18 pages, janvier 1997, Sophia-Antipolis
97-86	B. Neveu G. Trombettoni	*Computational complexity of Multi-way, Dataflow constraint problems* 13 pages, janvier 1997, Sophia-Antipolis

Ces rapports peuvent être obtenus en s'adressant au secrétariat du CERMICS :
The reports can be asked from:

Imane HAMADE
ENPC-CERMICS
6 et 8 Avenue Blaise Pascal
Cité Descartes Champs-sur-Marne
77455-Marne-La-Vallée CEDEX 2
Tél : (33) 01 - 64-15-35-71
email: `hamade@newaphro.enpc.fr`