Learning a classification of Mixed-Integer Quadratic Programming problems

CERMICS 2018 – June 29, 2018, Fréjus

Pierre Bonami[1], **Andrea Lodi**[2], Giulia Zarpellon[2]

[1] CPLEX Optimization, IBM Spain
[2] Polytechnique Montréal, CERC Data Science for real-time Decision Making

Table of contents

About Mixed-Integer Quadratic Programming problems

Mixed-Integer Quadratic Programming problems

We consider *Mixed-Integer Quadratic Programming* (MIQP) pbs.

$$\begin{aligned} \min \quad & \frac{1}{2}x^T Qx + c^T x \\ & Ax = b \\ & x_i \in \{0,1\} \quad \forall\, i \in I \\ & l \leq x \leq u \end{aligned} \tag{1}$$

- Modeling of practical applications (e.g., portfolio optimization)
- First extension of linear algorithms into nonlinear ones

Mixed-Integer Quadratic Programming problems

We consider *Mixed-Integer Quadratic Programming* (MIQP) pbs.

$$\begin{aligned} \min \quad & \frac{1}{2}x^T Qx + c^T x \\ & Ax = b \\ & x_i \in \{0,1\} \quad \forall\, i \in I \\ & l \le x \le u \end{aligned} \tag{1}$$

- Modeling of practical applications (e.g., portfolio optimization)
- First extension of linear algorithms into nonlinear ones

We say an MIQP is ***convex*** (resp. ***nonconvex***) if and only if the matrix Q is positive semi-definite, $Q \succeq 0$ (resp. indefinite, $Q \not\succeq 0$).

⟶ `IBM-CPLEX` solver can solve both convex and nonconvex MIQPs to proven optimality

Solving MIQPs with CPLEX

Convex 0-1

NLP B&B

Convex mixed

NLP B&B

Solving MIQPs with CPLEX

Convex 0-1

NLP B&B

Convex mixed

NLP B&B

Nonconvex 0-1

convexify + NLP B&B

Convexification: **augment diagonal** of Q, using $x_i = x_i^2$ for $x_i \in \{0, 1\}$:
$x^T Q x \to x^T (Q + \rho \mathbb{I}_n) x - \rho e^T x$, where $Q + \rho \mathbb{I}_n \succeq 0$ for some $\rho > 0$

Solving MIQPs with CPLEX

Convex 0-1

NLP B&B

Convex mixed

NLP B&B

Nonconvex 0-1

convexify + NLP B&B
linearize + MILP B&B

Convexification: **augment diagonal** of Q, using $x_i = x_i^2$ for $x_i \in \{0,1\}$:
$x^T Q x \to x^T(Q + \rho \mathbb{I}_n)x - \rho e^T x$, where $Q + \rho \mathbb{I}_n \succeq 0$ for some $\rho > 0$

Linearization: replace $q_{ij} x_i x_j$ where $x_i \in \{0,1\}$ and $l_j \leq x_j \leq u_j$ with a new variable y_{ij} and **McCormick inequalities**

$\longrightarrow$ Linearization is always full in 0-1 case

Solving MIQPs with CPLEX

Convex 0-1

NLP B&B

Convex mixed

NLP B&B

Nonconvex 0-1

convexify + NLP B&B
linearize + MILP B&B

Nonconvex mixed

Convexification is relaxation - Spatial B&B

Convexification: **augment diagonal** of Q, using $x_i = x_i^2$ for $x_i \in \{0, 1\}$:
$x^T Qx \to x^T(Q + \rho\mathbb{I}_n)x - \rho e^T x$, where $Q + \rho\mathbb{I}_n \succeq 0$ for some $\rho > 0$

Linearization: replace $q_{ij}x_i x_j$ where $x_i \in \{0, 1\}$ and $l_j \leq x_j \leq u_j$ with a new variable y_{ij} and **McCormick inequalities**

$\longrightarrow$ Linearization is always full in 0-1 case

Solving MIQPs with CPLEX

Convex 0-1

NL: NLP B&B
L: linearize + MILP B&B

Convex mixed

NL: NLP B&B
L: linearize + MILP B&B
linearize + NLP B&B

Nonconvex 0-1

NL: convexify + NLP B&B
L: linearize + MILP B&B

Convexification: **augment diagonal** of Q, using $x_i = x_i^2$ for $x_i \in \{0,1\}$:
$x^T Q x \to x^T(Q + \rho \mathbb{I}_n)x - \rho e^T x$, where $Q + \rho \mathbb{I}_n \succeq 0$ for some $\rho > 0$

Linearization: replace $q_{ij} x_i x_j$ where $x_i \in \{0,1\}$ and $l_j \leq x_j \leq u_j$ with a new variable y_{ij} and **McCormick inequalities**
$\longrightarrow$ Linearization is always full in 0-1 MIQP *(may not for mixed ones)*

Linearize vs. not linearize

The linearization approach seems beneficial also for the convex case, but **is linearizing always the best choice?**

Linearize vs. not linearize

The linearization approach seems beneficial also for the convex case, but **is linearizing always the best choice?**

Example[1]: convex 0-1 MIQP, $n = 200$

Linearize

```
Total (root+branch&cut) = 2112.63 sec.

CPLEX 12.6.0.0: optimal integer solution;
objective 29576.27517
474330 MIP simplex iterations
294 branch-and-bound nodes
```

Not linearize

```
Total (root+branch&cut) = 0.42 sec.

CPLEX 12.5.0.1: optimal integer solution;
objective 29576.27517
286 MIP simplex iterations
102 branch-and-bound nodes
```

[1] Fourer R. Quadratic Optimization Mysteries, Part 1: Two Versions.

Linearize vs. not linearize

The linearization approach seems beneficial also for the convex case, but **is linearizing always the best choice?**

Example[1]: convex 0-1 MIQP, $n = 200$

Linearize

```
Total (root+branch&cut) = 2112.63 sec.

CPLEX 12.6.0.0: optimal integer solution;
objective 29576.27517
474330 MIP simplex iterations
294 branch-and-bound nodes
```

```
MIP Presolve added 39800 rows and 19900
columns.
Reduced MIP has 39801 rows, 20100
columns, and 79800 nonzeros.
Reduced MIP has 20100 binaries, 0
general, and 0 indicators.
```

Not linearize

```
Total (root+branch&cut) = 0.42 sec.

CPLEX 12.5.0.1: optimal integer solution;
objective 29576.27517
286 MIP simplex iterations
102 branch-and-bound nodes
```

"[. . .] when one looks at a broader variety of test problems the decision to ***linearize (vs. not linearize)*** *does not appear so clear-cut.[2]"*

[1] Fourer R. Quadratic Optimization Mysteries, Part 1: Two Versions.
[2] Fourer R. Quadratic Optimization Mysteries, Part 2: Two Formulations.
http://bob4er.blogspot.com/2015/03/quadratic-optimization-mysteries-part-2.html

Linearize vs. not linearize - Goal

Goal

Exploit ML predictive machinery to understand whether it is favorable to linearize the quadratic part of an MIQP or not.

- Learn an offline classifier predicting the most suited resolution approach within `CPLEX` framework, in an instance-specific way

Linearize vs. not linearize - Goal

Goal

Exploit ML predictive machinery to understand whether it is favorable to linearize the quadratic part of an MIQP or not.

- Learn an offline classifier predicting the most suited resolution approach within `CPLEX` framework, in an instance-specific way
- Restrict to three types of problems

  0-1 convex, mixed convex, 0-1 nonconvex

Linearize vs. not linearize - Goal

Goal

Exploit ML predictive machinery to understand whether it is favorable to linearize the quadratic part of an MIQP or not.

- Learn an offline classifier predicting the most suited resolution approach within `CPLEX` framework, in an instance-specific way
- Restrict to three types of problems

  0-1 convex, mixed convex, 0-1 nonconvex
- Parameter `qtolin` controls the linearization switch

L	**on**	`CPLEX` linearizes (*all?*) quadratic terms
NL	**off**	`CPLEX` does *not* linearize quadratic terms
`DEF`	**auto**	Let `CPLEX` decide (default)

Data and experiments

Steps to apply supervised learning

Dataset generation

- Generator of MIQPs, spanning over various parameters
- `Q = sprandsym(size, density, eigenvalues)`

Steps to apply supervised learning

Dataset generation

- Generator of MIQPs, spanning over various parameters
- `Q = sprandsym(size, density, eigenvalues)`

Features design

- Static features (21) · Mathematical characteristics
  (variables, constraints, objective, spectrum, ...)
- Dynamic features (2) · Early behavior in optimization process
  (bounds and times at root node)

Steps to apply supervised learning

Dataset generation

- Generator of MIQPs, spanning over various parameters
- `Q = sprandsym(size, density, eigenvalues)`

Features design

- Static features (21) · Mathematical characteristics
  (variables, constraints, objective, spectrum, ...)
- Dynamic features (2) · Early behavior in optimization process
  (bounds and times at root node)

Labeling procedure

- Consider tie cases · Labels in $\{\mathbf{L}, \mathbf{NL}, \mathbf{T}\}$
- 1h, 5 seeds · Solvability and consistency checks
- Look at runtimes to assign a label

Steps to apply supervised learning

Dataset generation

- Generator of MIQPs, spanning over various parameters
- `Q = sprandsym(size, density, eigenvalues)`

Features design

- Static features (21) · Mathematical characteristics
  (variables, constraints, objective, spectrum, ...)
- Dynamic features (2) · Early behavior in optimization process
  (bounds and times at root node)

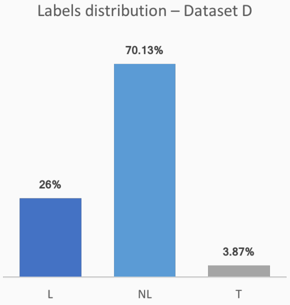
Labeling procedure

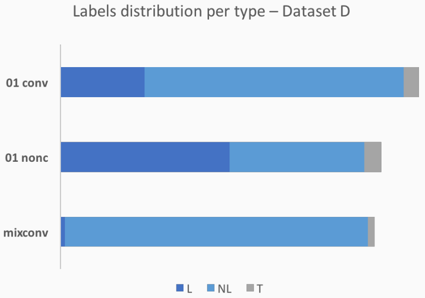
- Consider tie cases · Labels in $\{\mathbf{L}, \mathbf{NL}, \mathbf{T}\}$
- 1h, 5 seeds · Solvability and consistency checks
- Look at runtimes to assign a label

$\{(\mathbf{x}^k, \mathbf{y}^k)\}_{k=1..N}$ where $\mathbf{x}^k \in \mathbb{R}^d$, $\mathbf{y}^k \in \{\mathrm{L}, \mathrm{NL}, \mathrm{T}\}$ for N MIQPs

Dataset $\mathcal{D}$ (nutshell) analysis

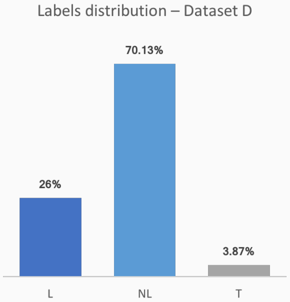
- **2300** instances, $n \in \{25, 50, \ldots, 200\}$, density $d \in \{0.2, 0.4, \ldots, 1\}$

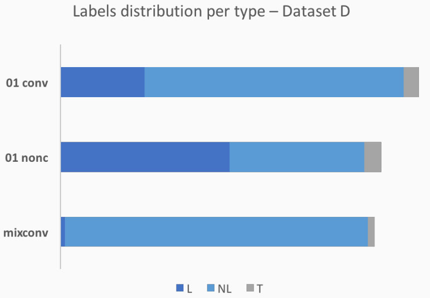




Dataset $\mathcal{D}$ (nutshell) analysis

- **2300** instances, $n \in \{25, 50, \ldots, 200\}$, density $d \in \{0.2, 0.4, \ldots, 1\}$





- **Multiclass classifiers**: SVM and Decision Tree based methods (Random Forests (RF) · Extremely Randomized Trees (EXT) · Gradient Tree Boosting (GTB))
- Avoid overfitting with **ML best practices**
- **Tool**: `scikit-learn` library

Learning results on $\mathcal{D}_{test}$

Classifiers perform well with respect to **traditional classification measures**:

$\mathcal{D}_{test}$ - *Multiclass - All features*				
	SVM	RF	EXT	GTB
Accuracy	0.85	0.89	0.84	0.87
Precision	0.82	0.85	0.81	0.85
Recall	0.85	0.89	0.84	0.87
F1 score	0.83	0.87	0.82	0.86

- A major difficulty is posed by the T class, (almost) always misclassified

Learning results on $\mathcal{D}_{test}$

Classifiers perform well with respect to **traditional classification measures**:

$\mathcal{D}_{test}$ - *Multiclass - All features*				
	SVM	RF	EXT	GTB
Accuracy	0.85	0.89	0.84	0.87
Precision	0.82	0.85	0.81	0.85
Recall	0.85	0.89	0.84	0.87
F1 score	0.83	0.87	0.82	0.86

- A major difficulty is posed by the T class, (almost) always misclassified

⟶ **Binary setting**: remove all tie cases · performance improved
How relevant are ties with respect to the question L vs. NL?

Hints of features importance

Ensemble methods based on Decision Trees provide an **importance score** for each feature.

Top scoring ones are dynamic ft.s and those about eigenvalues:

(dyn. ft.)
- Difference of lower bounds for L and NL at root node

(dyn. ft.)
- Difference of resolution times of the root node, for L and NL
- Value of smallest nonzero eigenvalue
- Spectral norm of Q, i.e., $\|Q\| = \max_i |\lambda_i|$
- ...

Hints of features importance

Ensemble methods based on Decision Trees provide an **importance score** for each feature.

Top scoring ones are dynamic ft.s and those about eigenvalues:

- (dyn. ft.) Difference of lower bounds for L and NL at root node
- (dyn. ft.) Difference of resolution times of the root node, for L and NL
- Value of smallest nonzero eigenvalue
- Spectral norm of Q, i.e., $\|Q\| = \max_i |\lambda_i|$
- ...

$\longrightarrow$ **Static features setting**: remove dynamic features · performance slightly deteriorated

How does the prediction change without information at root node?

Measure the optimization gain

Need

Evaluate classifiers' performance in optimization terms, and quantify the gain with respect to `CPLEX` *default strategy*.

Measure the optimization gain

Need

Evaluate classifiers' performance in optimization terms, and quantify the gain with respect to `CPLEX` *default strategy*.

- Run each test example once more, for all configurations of `qtolin` (on/L, off/NL, `DEF`) and collect resolution times

Measure the optimization gain

Need

Evaluate classifiers' performance in optimization terms, and quantify the gain with respect to `CPLEX` *default strategy*.

- Run each test example once more, for all configurations of `qtolin` (on/L, off/NL, `DEF`) and collect resolution times
- For each classifier *clf* and DEF, build a **times vector $\mathbf{t_{clf}}$**: for every test example, select the runtime corresponding to its label $\in \{L, NL, T\}$, as predicted by *clf*

  Average L and NL times in case of T being predicted

Measure the optimization gain

Need

Evaluate classifiers' performance in optimization terms, and quantify the gain with respect to `CPLEX` *default strategy*.

- Run each test example once more, for all configurations of `qtolin` (on/L, off/NL, `DEF`) and collect resolution times
- For each classifier *clf* and DEF, build a **times vector $\mathbf{t_{clf}}$**: for every test example, select the runtime corresponding to its label $\in$ {L, NL, T}, as predicted by *clf*

  Average L and NL times in case of T being predicted

- Build also t_{best} (t_{worst}) containing times corresponding to the correct (wrong) labels

Complementary optimization measures

Using times vectors $\mathbf{t}_{\mathbf{clf}}$ define

- $\sigma_{\mathbf{clf}}$ **Sum of predicted runtimes**: sum over times in t_{clf}
- $\mathbf{N}\sigma_{\mathbf{clf}} \in [0,1]$ **Normalized time score**: shifted geometric mean of times in t_{clf}, normalized between best and worst cases

Complementary optimization measures

Using times vectors $\mathbf{t_{clf}}$ define

$\sigma_{\mathbf{clf}}$ **Sum of predicted runtimes**: sum over times in t_{clf}

$\mathbf{N}\sigma_{\mathbf{clf}} \in [0,1]$ **Normalized time score**: shifted geometric mean of times in t_{clf}, normalized between best and worst cases

	SVM	RF	EXT	GTB	DEF
$\sigma_{\mathtt{DEF}}/\sigma_{clf}$	3.88	4.40	4.04	4.26	–
$N\sigma_{clf}$	0.98	0.99	0.98	0.99	0.42

Complementary optimization measures

Using times vectors $\mathbf{t_{clf}}$ define

$\sigma_{\mathbf{clf}}$ **Sum of predicted runtimes**: sum over times in t_{clf}

$\mathbf{N}\sigma_{\mathbf{clf}}$ $\in [0,1]$ **Normalized time score**: shifted geometric mean of times in t_{clf}, normalized between best and worst cases

	SVM	RF	EXT	GTB	DEF
$\sigma_{\texttt{DEF}}/\sigma_{clf}$	3.88	4.40	4.04	4.26	–
$N\sigma_{clf}$	0.98	0.99	0.98	0.99	0.42

- DEF could take up to 4x more time to run MIQPs of $\mathcal{D}_{test}$, compared to a trained classifier
- Measures are better for classifiers hitting **timelimit** less frequently (and both L and NL reach timelimit multiple times!)

Ongoing questions

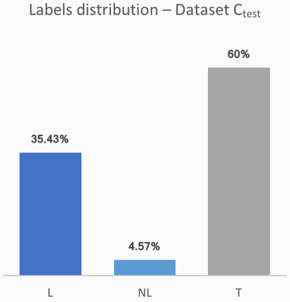
What about other datasets?

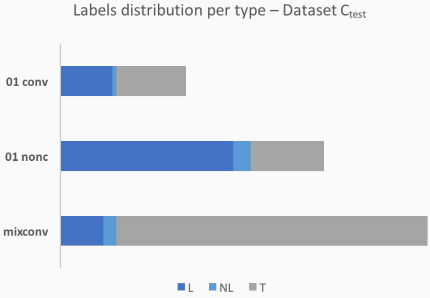
What about other datasets?

- Selection from QPLIB · **24 instances**

What about other datasets?

- Selection from QPLIB · **24 instances**
- Part of `CPLEX` internal testbed · **175 instances**, used as new test set $\mathcal{C}_{test}$ for classifiers trained on the synthetic data. Again **unbalanced**, but with majority of ties and few NL.





Results on $\mathcal{C}_{test}$

- $\mathcal{C}_{test}$ is dominated by few structured combinatorial instances
- **Very different distribution** of features

All classifiers perform very poorly in terms of classification measures (most often T is predicted as NL), **but** . . .

Results on $\mathcal{C}_{test}$

- $\mathcal{C}_{test}$ is dominated by few structured combinatorial instances
- **Very different distribution** of features

All classifiers perform very poorly in terms of classification measures (most often T is predicted as NL), **but** ...

... performance is not bad in optimization terms:

	SVM	RF	EXT	GTB	**DEF**
$\sigma_{\texttt{DEF}}/\sigma_{clf}$	0.48	0.53	0.71	0.42	–
$N\sigma_{clf}$	0.75	0.90	0.91	0.74	**0.96**

Given the high **presence of ties**, runtimes for L and NL are most often comparable, so **the loss in performance is not dramatic**.

Why those predictions?

Why those predictions?

- The **bound at the root** node seems to decide the label!
- **Convexification** and **linearization** clearly affect

  • **formulation size** • **formulation strength** • **implementation efficacy** • ...

  ...each problem type might have its own decision function for the question *L vs. NL*

Why those predictions?

- The **bound at the root** node seems to decide the label!
- **Convexification** and **linearization** clearly affect

  • **formulation size** • **formulation strength** • **implementation efficacy** • ...

  ...each problem type might have its own decision function for the question *L vs. NL*

More experiments to come:

- Employ a **larger** and **heterogeneous dataset**
- Go beyond preliminary **features evaluation**
- Define a custom **loss/scoring function**

Thanks! Questions?