Learning (Discrete) Optimization

Andrea Lodi

Canada Excellence Research Chair
École Polytechnique de Montréal, Québec, Canada
andrea.lodi@polymtl.ca

Deep Learning Summit 2017
@ Montréal (Canada), October 10, 2017

CANADA EXCELLENCE RESEARCH CHAIR

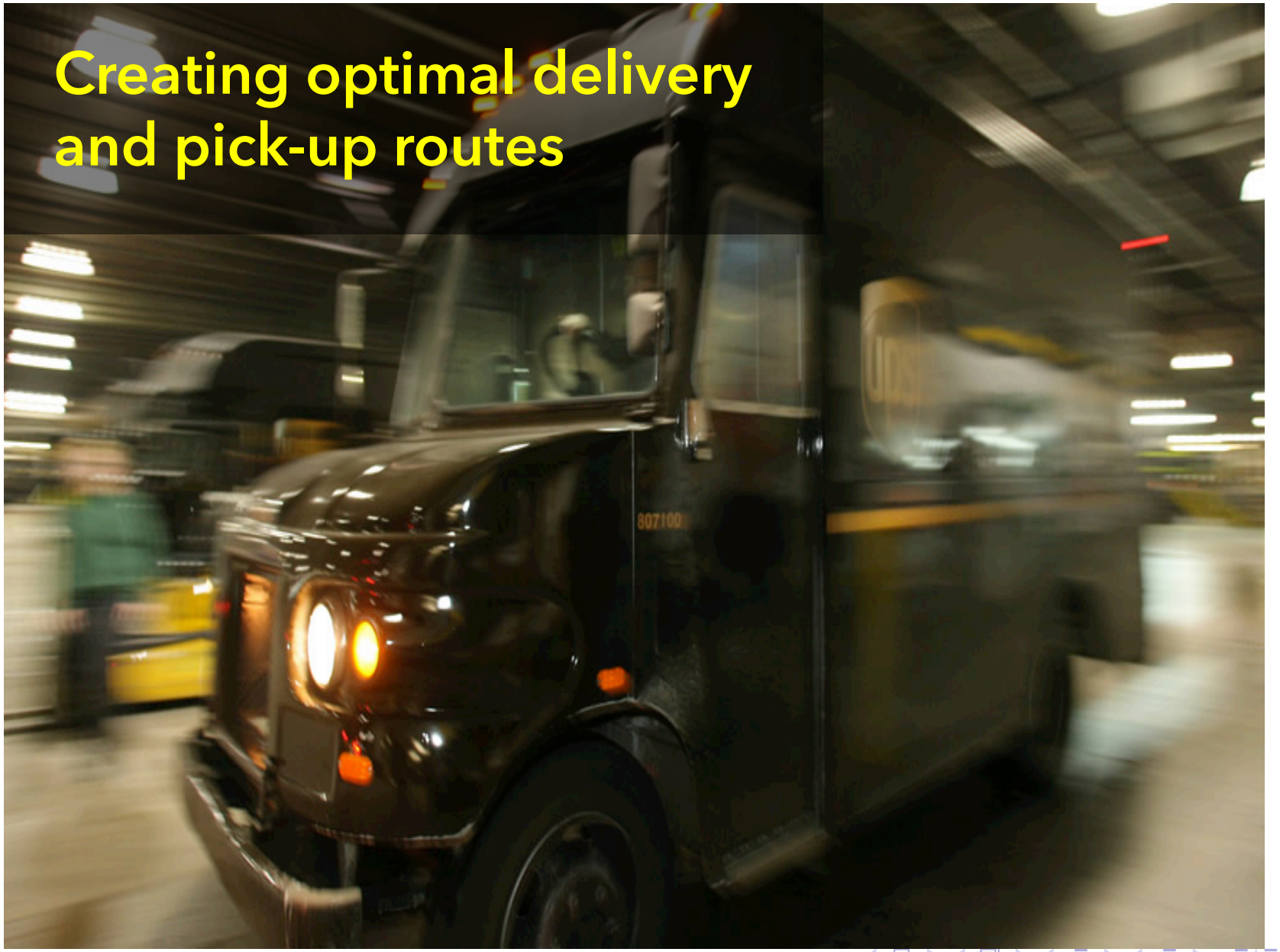
DATA SCIENCE FOR REAL-TIME DECISION-MAKING

Outline

1. Solving Discrete Optimization Problems
2. The role of ML / DL for Discrete Optimization
3. A concrete example: MIQPs classification
4. Open Problems

Discrete Optimization Problems

Discrete Optimization Problems (cont.d)



Solving Discrete Optimization Problems

We consider a general Mixed Integer Linear Programming problem (MIP) in the form

$$\max\{c^T x : Ax \leq b, x \geq 0,\ x_j \in \mathbb{Z}, \forall j \in I\} \tag{1}$$

where matrix A does not have a special structure.

Solving Discrete Optimization Problems

We consider a general Mixed Integer Linear Programming problem (MIP) in the form

$$\max\{c^T x : Ax \leq b, x \geq 0,\ x_j \in \mathbb{Z}, \forall j \in I\} \tag{1}$$

where matrix A does not have a special structure.

Thus, the problem is solved through Branch and Bound and the bounds are computed by iteratively solving the Linear Programming (LP) relaxations through a general-purpose LP solver.

Solving Discrete Optimization Problems

We consider a general Mixed Integer Linear Programming problem (MIP) in the form

$$\max\{c^T x : Ax \leq b, x \geq 0,\ x_j \in \mathbb{Z}, \forall j \in I\} \tag{1}$$

where matrix A does not have a special structure.

Thus, the problem is solved through Branch and Bound and the bounds are computed by iteratively solving the Linear Programming (LP) relaxations through a general-purpose LP solver.

MIP technology has been proven to generalize well: the algorithms developed having specific applications in mind (production planning, routing , staff scheduling, etc.) are now included in both commercial and open-source MIP solvers.

Solving Discrete Optimization Problems

We consider a general Mixed Integer Linear Programming problem (MIP) in the form

$$\max\{c^T x : Ax \leq b, x \geq 0,\ x_j \in \mathbb{Z}, \forall j \in I\} \tag{1}$$

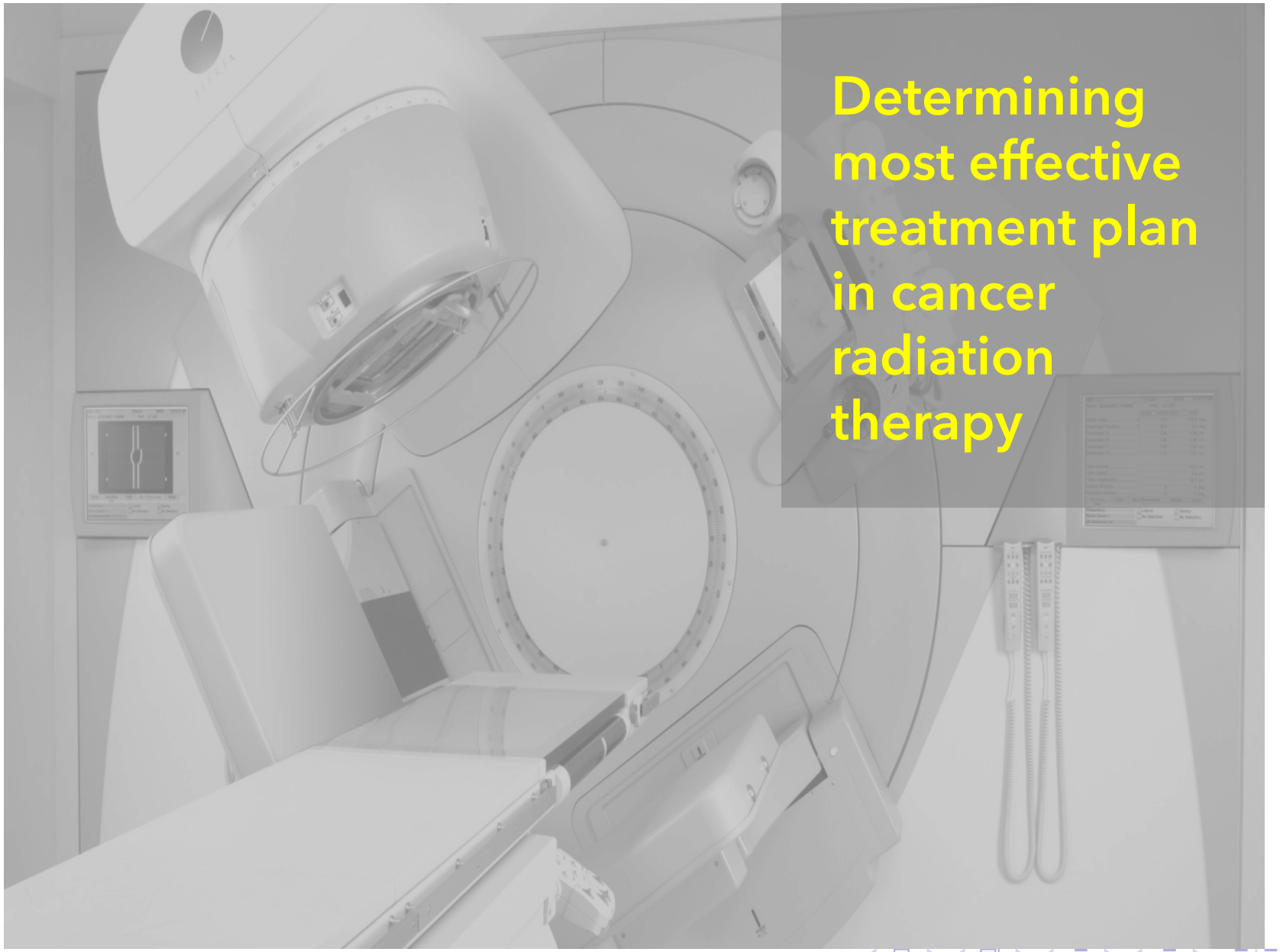
where matrix A does not have a special structure.

Thus, the problem is solved through Branch and Bound and the bounds are computed by iteratively solving the Linear Programming (LP) relaxations through a general-purpose LP solver.

MIP technology has been proven to generalize well: the algorithms developed having specific applications in mind (production planning, routing , staff scheduling, etc.) are now included in both commercial and open-source MIP solvers.

In turn, those MIP solvers are, to a large extent, used at least as a starting point to attack the problems in most of the other applied contexts.

Discrete Optimization Problems (cont.d)



Discrete Optimization Problems (cont.d)

The role of ML / DL for Discrete Optimization

A fast growing literature has started to appear in the last 5 to 10 years on the use of Machine Learning techniques to help Optimization, especially MIP solvers. Among the first in these series, the papers on tuning MIP solvers. [Hoos et al. (2010+)]

The role of ML / DL for Discrete Optimization

A fast growing literature has started to appear in the last 5 to 10 years on the use of Machine Learning techniques to help Optimization, especially MIP solvers. Among the first in these series, the papers on tuning MIP solvers. [Hoos et al. (2010+)]

ML has started to be used within Constraint Programming as well, including Neural Networks and Decision Trees. [Lombardi & Milano (2015+)]

The role of ML / DL for Discrete Optimization

A fast growing literature has started to appear in the last 5 to 10 years on the use of Machine Learning techniques to help Optimization, especially MIP solvers. Among the first in these series, the papers on tuning MIP solvers. [Hoos et al. (2010+)]

ML has started to be used within Constraint Programming as well, including Neural Networks and Decision Trees. [Lombardi & Milano (2015+)]

Learning when to use a decomposition. [Kruber, Lübbecke, Parmentier (2017)]

The role of ML / DL for Discrete Optimization

A fast growing literature has started to appear in the last 5 to 10 years on the use of Machine Learning techniques to help Optimization, especially MIP solvers. Among the first in these series, the papers on tuning MIP solvers. [Hoos et al. (2010+)]

ML has started to be used within Constraint Programming as well, including Neural Networks and Decision Trees. [Lombardi & Milano (2015+)]

Learning when to use a decomposition. [Kruber, Lübbecke, Parmentier (2017)]

MIP solvers are complex software objects implementing a large variety of algorithmic approaches. Strategic decisions on how to combine those approaches in the most effective way have to be taken over and over.

The role of ML / DL for Discrete Optimization

A fast growing literature has started to appear in the last 5 to 10 years on the use of Machine Learning techniques to help Optimization, especially MIP solvers. Among the first in these series, the papers on tuning MIP solvers. [Hoos et al. (2010+)]

ML has started to be used within Constraint Programming as well, including Neural Networks and Decision Trees. [Lombardi & Milano (2015+)]

Learning when to use a decomposition. [Kruber, Lübbecke, Parmentier (2017)]

MIP solvers are complex software objects implementing a large variety of algorithmic approaches. Strategic decisions on how to combine those approaches in the most effective way have to be taken over and over. Such decisions are taken heuristically, often breaking ties in architecture-dependent ways, thus showing the heuristic nature of MIP implementations. [Lodi (2012)]

The role of ML / DL for Discrete Optimization

A fast growing literature has started to appear in the last 5 to 10 years on the use of Machine Learning techniques to help Optimization, especially MIP solvers. Among the first in these series, the papers on tuning MIP solvers. [Hoos et al. (2010+)]

ML has started to be used within Constraint Programming as well, including Neural Networks and Decision Trees. [Lombardi & Milano (2015+)]

Learning when to use a decomposition. [Kruber, Lübbecke, Parmentier (2017)]

MIP solvers are complex software objects implementing a large variety of algorithmic approaches. Strategic decisions on how to combine those approaches in the most effective way have to be taken over and over. Such decisions are taken heuristically, often breaking ties in architecture-dependent ways, thus showing the heuristic nature of MIP implementations. [Lodi (2012)]

ML can help systematize the process that leads to take these decisions, especially when a large quantity of data can be collected.

A concrete example: MIQPs classification

We consider Mixed-Integer Quadratic Programming (MIQP)

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

where $Q = \{q_{ij}\}_{i,j=1\ldots n} \in \mathbb{R}^{n \times n}$ is a real symmetric matrix, either convex or nonconvex, and all integer variables are binary.

A concrete example: MIQPs classification

We consider Mixed-Integer Quadratic Programming (MIQP)

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

where $Q = \{q_{ij}\}_{i,j=1\ldots n} \in \mathbb{R}^{n \times n}$ is a real symmetric matrix, either convex or nonconvex, and all integer variables are binary.

Depending on the problems' structure, we can tackle MIQP in different ways:

- $Q \succeq 0$: perform NLP based B&B (or Outer Approximation algorithms)
- $Q \not\succeq 0$: depending on variables' type,
  - pure 0-1: transform into either a convex MIQP or into a MIP (i.e., linearize it)
  - mixed: perform $Q-$space reformulation/relaxation, run Global Optimization algorithms (Spatial B&B)

MIQPs classification (cont.d)

The linearization approach seems beneficial also for the convex case, both for pure 0-1 and mixed problems. However, is linearizing always the best choice?

"[. . .] when one looks at a broader variety of test problems the decision to linearize (vs. not linearize) does not appear so clear-cut.[1] *"*

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

MIQPs classification (cont.d)

The linearization approach seems beneficial also for the convex case, both for pure 0-1 and mixed problems. However, is linearizing always the best choice?

"[. . .] when one looks at a broader variety of test problems the decision to linearize (vs. not linearize) does not appear so clear-cut.[1] *"*

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

---

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

MIQPs classification (cont.d)

The linearization approach seems beneficial also for the convex case, both for pure 0-1 and mixed problems. However, is linearizing always the best choice?

"[. . .] when one looks at a broader variety of test problems the decision to linearize (vs. not linearize) does not appear so clear-cut.[1] *"*

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

---

- Learn an offline classifier predicting the most suited resolution approach within `IBM-CPLEX` framework (`qtolin` linearization switch parameter)
- Gain theoretical insights about which features of the MIQPs most affect the prediction

[Bonami, Lodi, Zarpellon (2017)]

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

MIQPs classification - Dataset generation

. . . traditional benchmark sets are too small for learning!

MIQPs classification - Dataset generation

. . . traditional benchmark sets are too small for learning!

We define and implement a generator of MIQP instances, spanning a variety of structural parameters and optimization components.

(I) Objective function data generation: real symmetric matrices are generated via the `MATLAB` function

```
Q = sprandsym(size, density, eigenvalues)
```

(II) Variables' type definition: binary/continuous variables are added to the problems with respect to the sign of Q, in different proportions

(III) Constraints generation: different constraints sets are added accordingly to the type of variables of the problem (e.g., cardinality, simplex, multi-dimensional knapsack)

MIQPs classification - Dataset generation

. . . traditional benchmark sets are too small for learning!

We define and implement a generator of MIQP instances, spanning a variety of structural parameters and optimization components.

(I) Objective function data generation: real symmetric matrices are generated via the `MATLAB` function

```
Q = sprandsym(size, density, eigenvalues)
```

(II) Variables' type definition: binary/continuous variables are added to the problems with respect to the sign of Q, in different proportions

(III) Constraints generation: different constraints sets are added accordingly to the type of variables of the problem (e.g., cardinality, simplex, multi-dimensional knapsack)

- Dataset of 2300 instances, three types of MIQPs (0-1 convex, 0-1 nonconvex, mixed convex)
- Plan to compare with traditional benchmark libraries for test/extensions

MIQPs classification - Features design

We define a set of 23 features referring to an MIQP instance, and we divide them into two main blocks:

MIQPs classification - Features design

We define a set of 23 features referring to an MIQP instance, and we divide them into two main blocks:

- Static features describe the mathematical characteristics of the instance, in terms of
  - variables - e.g., number, types, presence in constraints and objective
  - constraints - e.g., coefficients and variables presence
  - quadratic objective function - e.g., coefficients, variables presence, sparsity, spectral properties

  They are extracted via `CPLEX/Python` before any solving (pre)process takes place.

MIQPs classification - Features design

We define a set of 23 features referring to an MIQP instance, and we divide them into two main blocks:

- Static features describe the mathematical characteristics of the instance, in terms of
  - variables - e.g., number, types, presence in constraints and objective
  - constraints - e.g., coefficients and variables presence
  - quadratic objective function - e.g., coefficients, variables presence, sparsity, spectral properties

  They are extracted via `CPLEX/Python` before any solving (pre)process takes place.

- Dynamic features describe the initial behavior with respect to different resolution methods.
  - e.g., bounds and solution times at the root node

  They are extracted from the early stages of the optimization, after the preprocessing and the resolution of the root node relaxation.

MIQPs classification - Labeling procedure

One of three different labels among $\{\mathrm{L}, \mathrm{NL}, \mathrm{T}\}$ can be assigned to an MIQP instance, describing the winner between *linearize*, *not-linearize* or the case of a *tie* of the two methods.

MIQPs classification - Labeling procedure

One of three different labels among $\{L, NL, T\}$ can be assigned to an MIQP instance, describing the winner between *linearize*, *not-linearize* or the case of a *tie* of the two methods.

Each problem of the dataset is run with timelimit of 1h, for 5 different random seeds, with `qtolin` on and off.

To address solvability / consistency issues, we perform

- Solvability check, to discard never-solved instances
- Seed consistency check on each seed, to discard unstable instances w.r.t. the found upper and lower bounds
- Global consistency check on global best upper and lower bounds, to discard unstable instances

Running time is the ultimate compared measure, assessing the final label for each example.

MIQPs classification - Learning experiments

Instances, features and labels give a dataset ready for supervised learning:

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

MIQPs classification - Learning experiments

Instances, features and labels give a dataset ready for supervised learning:

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

Multiclass classifiers such as

- Support Vector Machines (nonlinear RBF kernel) (SVM)

and ensemble methods based on Decision Trees such as

- Random Forests (RF)
- Extremely Randomized Trees (EXT)
- Gradient Tree Boosting (GTB)

MIQPs classification - Learning experiments

Instances, features and labels give a dataset ready for supervised learning:

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

Multiclass classifiers such as

- Support Vector Machines (nonlinear RBF kernel) (SVM)

and ensemble methods based on Decision Trees such as

- Random Forests (RF)
- Extremely Randomized Trees (EXT)
- Gradient Tree Boosting (GTB)

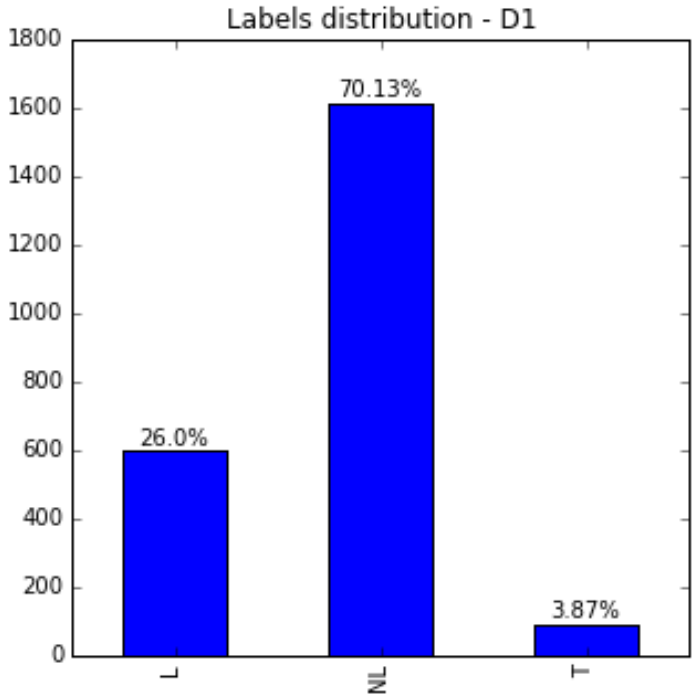
Methodology: follow ML best practices to avoid overfitting

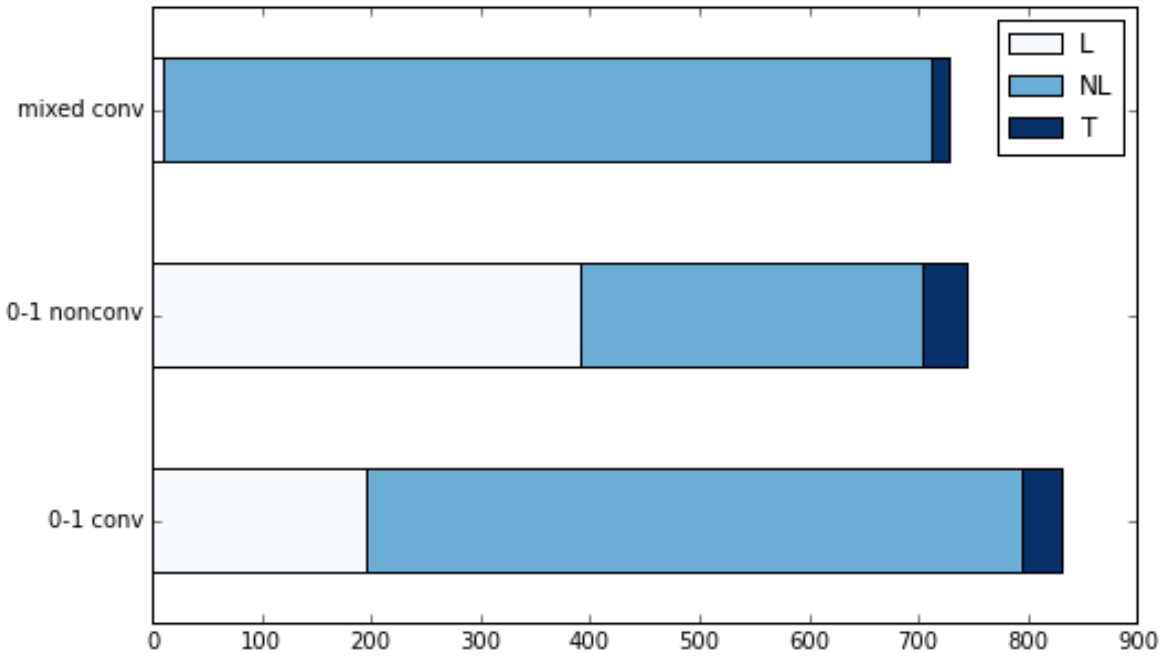
- Training phase to optimize parameters (1725 instances)
- k-fold cross validation and grid search for hyper-parameters selection
- Test phase to assess classifiers' performance (575 instances)

Main implementation tool: `scikit-learn` library.

MIQPs classification - Nutshell analysis

Before learning, look into the dataset! In a nutshell:





- Take care of unbalanced data in the learning procedure
- Can some trends already been recognized w.r.t. different problem types?
- More (statistical) analyses on features and data distribution

MIQPs classification - Some results

Classifiers perform well with respect to traditional classification measures

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

MIQPs classification - Some results

Classifiers perform well with respect to traditional classification measures

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

Top 5 - Features importance scores:

- difference of lower bounds found by L and NL at root node (dynamic ft.)
- difference of root node resolution times (dynamic ft.)
- value of smallest nonzero eigenvalue
- a measure of "diagonal dominance", computed as $\frac{1}{n}\sum_{i=1}^{n}(|q_{ii}| - \sum_{j\neq i}|q_{ij}|)$
- spectral norm of Q, i.e., $\|Q\| = \max_i |\lambda_i|$

MIQPs classification - Learning settings

Simplify the *Multiclass - All features* framework by considering

- Binary setting: remove all tie cases

  How relevant are ties with respect to the question L vs. NL?
  - Classification measures are overall improved, RF is still best performing.

- Static features setting: remove dynamic features

  How does the prediction change without information at root node?
  - Classification is slightly deteriorated, but overall coherent with the original one. The new best performing algorithm is SVM.

MIQPs classification - Learning settings

Simplify the *Multiclass - All features* framework by considering

- Binary setting: remove all tie cases

  How relevant are ties with respect to the question L vs. NL?
  - Classification measures are overall improved, RF is still best performing.

- Static features setting: remove dynamic features

  How does the prediction change without information at root node?
  - Classification is slightly deteriorated, but overall coherent with the original one. The new best performing algorithm is SVM.

- *Binary - Static features* setting, simplified in labels and features
  - Performance is balanced between improvement and deterioration, with SVM as best algorithm.
  - Static features about Q spectrum are the top ones.

What is the best learning setting to integrate predictions and solver?

MIQPs classification - Optimization scores

We need to evaluate classifiers' performance in optimization terms, and quantify the gain with respect to CPLEX default strategy (DEF)

- For each example, select the runtime corresponding to the predicted label (L, NL, T) to build a times vector t_{clf} for each classifier *clf* and DEF
- t_{best} (t_{worst}) contains times corresponding to the correct (wrong) labels

MIQPs classification - Optimization scores

We need to evaluate classifiers' performance in optimization terms, and quantify the gain with respect to CPLEX default strategy (DEF)

- For each example, select the runtime corresponding to the predicted label (L, NL, T) to build a times vector t_{clf} for each classifier *clf* and DEF
- t_{best} (t_{worst}) contains times corresponding to the correct (wrong) labels

σ_{clf} **Sum of predicted runtimes**: sum over times in t_{clf}

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

MIQPs classification - Optimization scores

We need to evaluate classifiers' performance in optimization terms, and quantify the gain with respect to CPLEX default strategy (DEF)

- For each example, select the runtime corresponding to the predicted label (L, NL, T) to build a times vector t_{clf} for each classifier *clf* and DEF
- t_{best} (t_{worst}) contains times corresponding to the correct (wrong) labels

σ_{clf} **Sum of predicted runtimes**: sum over times in t_{clf}

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

	SVM	RF	EXT	GTB	DEF
$\sigma_{clf}/\sigma_{best}$	1.49	1.31	1.43	1.35	5.77
$\sigma_{worst}/\sigma_{clf}$	7.48	8.49	7.81	8.23	1.93
$\sigma_{\text{DEF}}/\sigma_{clf}$	3.88	4.40	4.04	4.26	–
$N\sigma_{clf}$	0.98	0.99	0.98	0.99	0.42

MIQPs classification - CPLEX partial testbed

Preliminary experiments on partial CPLEX internal testbed (175 instances), used as new test set for classifiers trained on the synthetic data.

- Very different distribution of features, problem types and labels: T is the majority class, with very few NL
- All classifiers perform very poorly in terms of classification measures (and most often a T is predicted as NL), but . . .

MIQPs classification - CPLEX partial testbed

Preliminary experiments on partial CPLEX internal testbed (175 instances), used as new test set for classifiers trained on the synthetic data.

- Very different distribution of features, problem types and labels: T is the majority class, with very few NL
- All classifiers perform very poorly in terms of classification measures (and most often a T is predicted as NL), but . . .

  . . . performance is not bad in optimization terms:

	SVM	RF	EXT	GTB
$\sigma_{clf}/\sigma_{best}$	2.55	2.30	1.72	2.91
$\sigma_{worst}/\sigma_{clf}$	2.00	2.22	2.96	1.75
$N\sigma_{clf}$	0.75	0.90	0.91	0.74

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

MIQPs classification - Going further

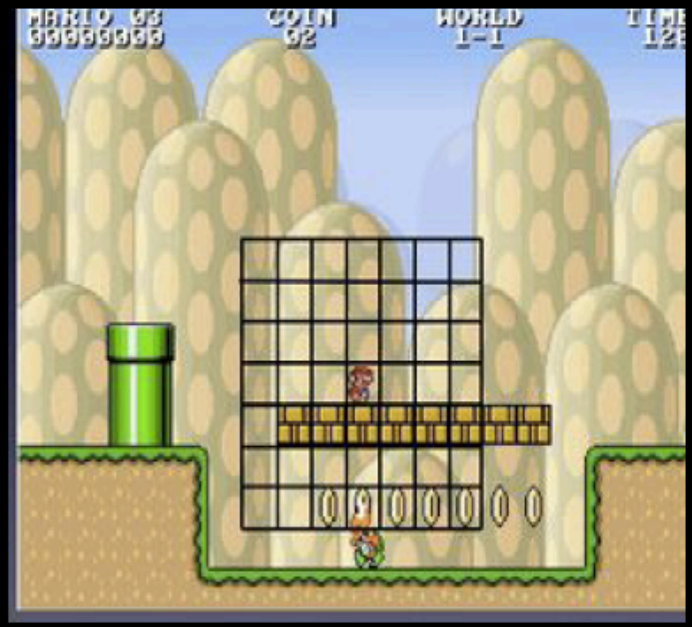
Directions for ongoing and future research:

- Analyze other benchmark datasets, e.g., QPLIB, to understand how representative the synthetic data is of commonly used instances, and enlarge the current training set
- Identify the best learning scenario in order to successfully integrate prediction and solver
- Define a custom loss function to train classifiers, to get a prediction tailored on the optimization aspects and the solver's performance as well

Open Problems: Learning to Search

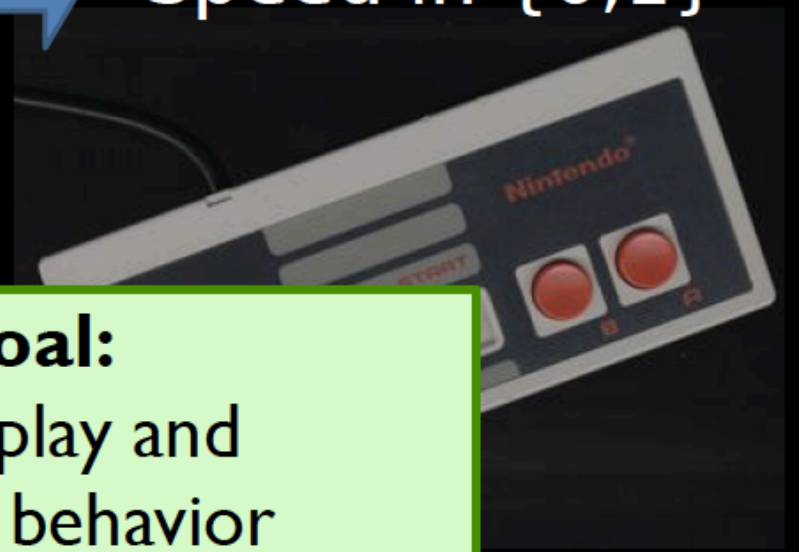
From Mario AI competition 2009

Input:



Output:

Jump in {0,1}
Right in {0,1}
Left in {0,1}
Speed in {0,1}



High level goal:
Watch an expert play and
learn to mimic her behavior

[Langford and Daumé III, 2015]

Open Problems: Variable selection in Branch and Bound

Branch-and-Bound algorithm (B&B):

- most widely used procedure for solving (Mixed-)Integer Programming problems
- implicit enumeration search, mapped into a decision tree
- leave (at least) two big choices:
  1. How to split a problem into subproblems (variable selection)
  2. Which node/subproblem to select for the next exploration

Open Problems: Variable selection in Branch and Bound

Branch-and-Bound algorithm (B&B):

- most widely used procedure for solving (Mixed-)Integer Programming problems
- implicit enumeration search, mapped into a decision tree
- leave (at least) two big choices:
  1. How to split a problem into subproblems (variable selection)
  2. Which node/subproblem to select for the next exploration

. . . decisions play a key role for the algorithm efficiency!

- as of today, decisions are made heuristically and empirically evaluated
- there are good branching strategies, but usually very costly

Open Problems: Variable selection in Branch and Bound

Branch-and-Bound algorithm (B&B):

- most widely used procedure for solving (Mixed-)Integer Programming problems
- implicit enumeration search, mapped into a decision tree
- leave (at least) two big choices:
  1. How to split a problem into subproblems (variable selection)
  2. Which node/subproblem to select for the next exploration

. . . decisions play a key role for the algorithm efficiency!

- as of today, decisions are made heuristically and empirically evaluated
- there are good branching strategies, but usually very costly

Ultimate goal (details to be discussed in slot #4)

Use ML to learn an activation function that can be adopted as approximation / prediction of a good B&B strategy, ideally with a low computational cost.

[Alvarez, Wehenkel & Louveaux (2016), Khalil, Le Bodic, Song, Nemhauser & Dilkina (2016)]

. . . getting on board in Montréal!

MIQPs - Dataset generation - Constraints details

For a generated Q matrix, the constraints set can comprise:

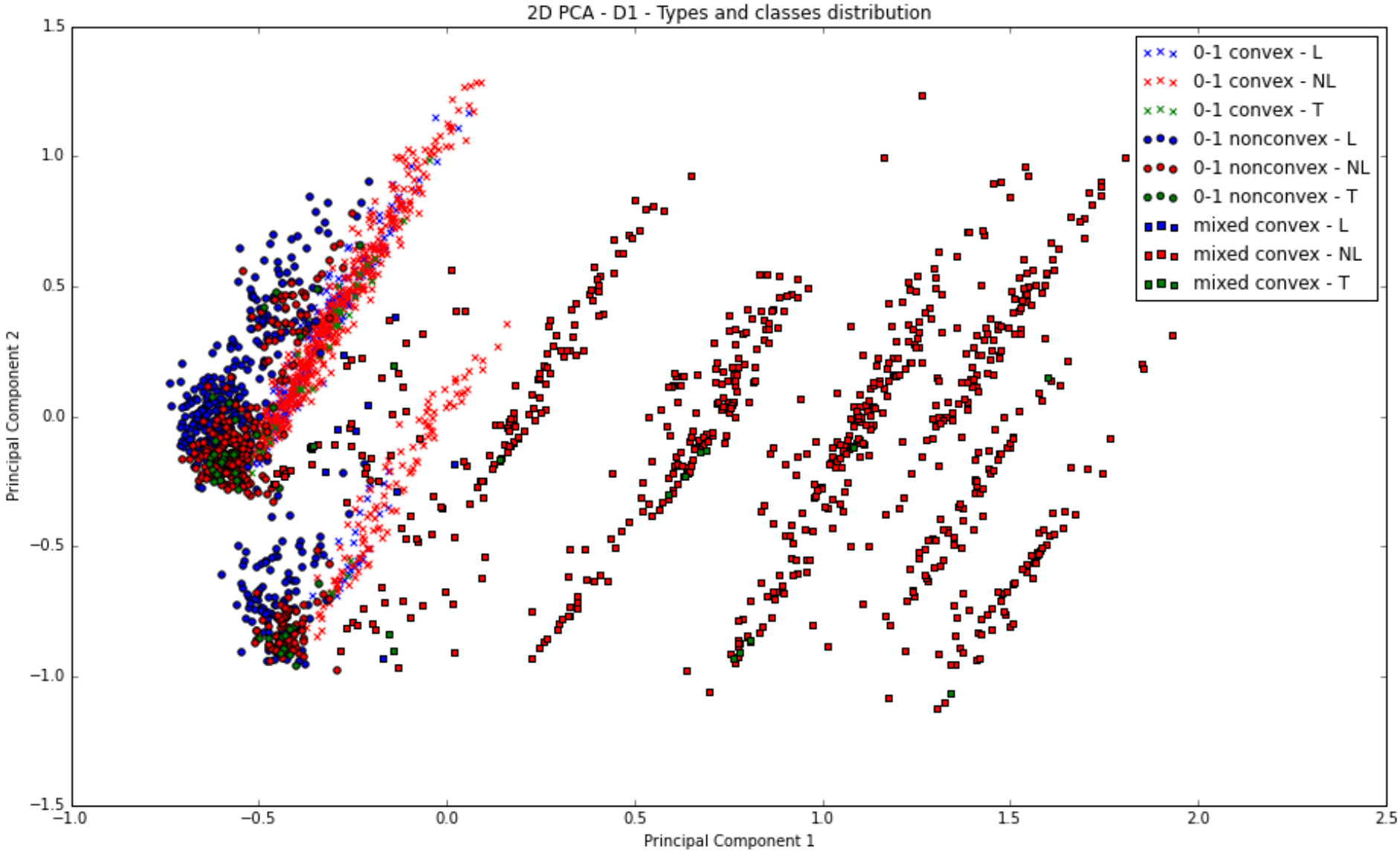
- Empty (unconstrained problem)
- A single *cardinality constraint* of the form $0 \leq \sum_{j=1}^{p} x_j \leq k$, for some random $k \in \mathbb{Z}$, with $k < p$, involving binary variables only. The rhs is chosen among $\{\lfloor \frac{1}{3}p \rfloor, \lfloor \frac{1}{2}p \rfloor, \lfloor \frac{2}{3}p \rfloor\}$
- A standard *simplex constraint* $\sum_{j=p+1}^{n} x_j = 1, x_j \geq 0$, involving continuous variables only
- *Multi-dimensional knapsack* constraints of the form $\sum_{j=1}^{p} w_{ij} x_j \leq c_i,\ i = 1, \ldots, m$, where only binary variables are involved and the dimension is given by $m \in \{1, 5, 10\}$. In our case, w_{ij} are positive integers weights drawn uniformly at random from (0, 1000), $c_i = \alpha \sum_{j=1}^{p} w_{ij}$ are capacities, $\alpha \in \{0.25, 0.5, 0.75\}$ being the tightness ratio.

	MK	Card	Simp	Card + Simp	Card + MK	Simp + MK
Convex 0-1	✓	✓			✓	
Nonconvex 0-1	✓	✓			✓	
Convex Mixed	✓	✓	✓	✓	✓	✓

MIQPs - Dataset generation - Parametric details

ID:	unique identifier for the instance
SIZE:	size $n \in \{25, 50, 75, 100, 125, 150, 175, 200\}$
DEN:	density $d \in \{1, 0.8, 0.6, 0.4, 0.2\}$
RANK:	uniformly drawn as $k \in [2, n] \subset \mathbb{Z}$
RAN_MIN:	specified interval $r_{\min} \in \{[-n, -\frac{n}{2}], [-\frac{n}{2}, 0], [0, \frac{n}{2}], [\frac{n}{2}, n]\}$
RAN_MAX:	specified interval $r_{\max} \in \{[-n, -\frac{n}{2}], [-\frac{n}{2}, 0], [0, \frac{n}{2}], [\frac{n}{2}, n]\}$
EIG_MIN:	$\lambda_{\min}$ uniformly picked from $r_{\min}$
EIG_MAX:	$\lambda_{\max}$ uniformly picked from $r_{\max}$
SIGN:	the sign of the quadratic matrix, determined among $\{\texttt{psd}, \texttt{ind}, \texttt{nsd}\}$
PREC:	number of decimal digits kept for objective function data (quadratic and linear)
LIN_BOOL:	boolean picked uniformly, decides if the linear part of the objective is added
MIX_PERC:	percentage of continuous variable, uniformly picked from $\{0, 20, 40, 60, 80\}$ (fixed to 0 if the matrix is not psd)
CARD:	boolean picked uniformly, deciding if a cardinality constraint is added
RHS:	refers to the rhs of the cardinality constraint (when CARD is True), which is picked uniformly from $\{\frac{1}{3}p, \frac{1}{2}p, \frac{2}{3}p\}$, $p = \#$ of defined binary variables
SIMP:	boolean picked uniformly, deciding if a simplex constraint is added (fixed to False if MIX_PERC is 0)
MK:	boolean picked uniformly, deciding if multi-knapsack constraints are added
DIM:	number m of multi-knapsack constraints, to be chosen uniformly from $\{1, 5, 10\}$ when MK is True
ALPHA:	tightness ratio $\alpha \in \{0.25, 0.5, 0.75\}$ uniformly drawn when MK is True

MIQPs - Dataset generation - 2D PCA visualization



MIQPs - Features design - Complete list (1)

Static features

1. ☐ Ratio between # of binary variables and total # of variables
2. ☐ Size of the problem (total # of variables)
3. ☐ Ratio between total # of binary variables appearing in constraints' matrix with nnz coefficients and total # nnz variables in constraints' matrix
4. ☐ Ratio of magnitudes of smallest on biggest absolute values of the constraints' matrix nnz coefficients
5. ☐ Ratio of # of binary variables with nnz diagonal coefficient and total
6. ☐ Ratio of # of continuous variables with nnz diagonal coefficient and total
7. ☐ Ratio of magnitudes of smallest on biggest absolute values of the objective matrix diagonal nonzero coefficients
8. ☐ Ratio of # of strictly positive eigenvalues (with multiplicities) on size
9. ☐ Ratio of # of strictly negative eigenvalues (with multiplicities, coincides with the dimension of the eigenspace spanned by eigenvectors relative to strictly negative eigenvalues) on size
10. ☐ # of strictly negative eigenvalues
11. ☐ Value of smallest nonzero eigenvalue

MIQPs - Features design - Complete list (2)

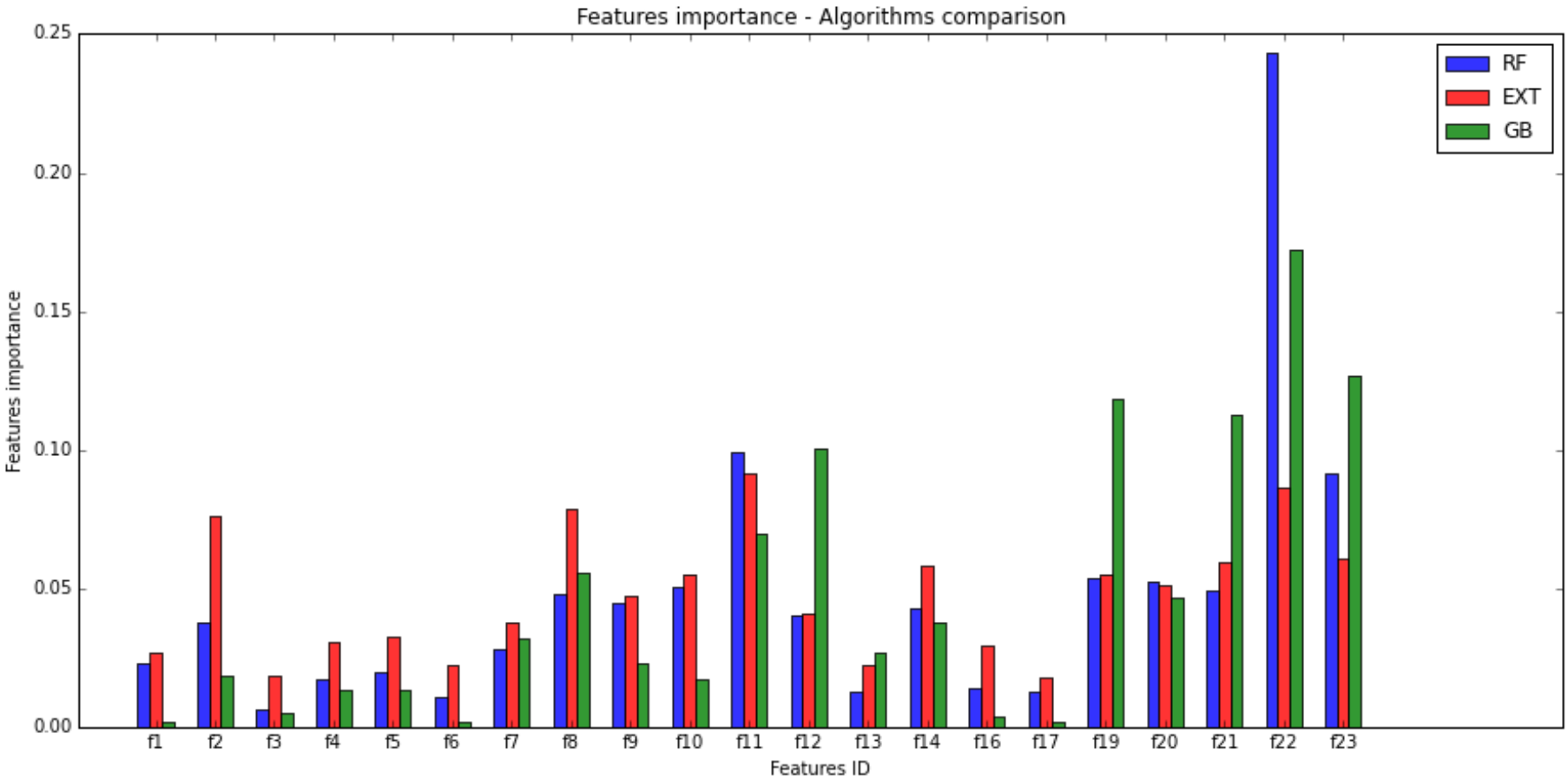
12.☐ Ratio of magnitudes of smallest on biggest nonzero eigenvalues' (in absolute values)
13.☐ Sparsity of Q as # of nonzeros coefficients on total # of entries
14.☐ Rank of Q
15.☐ Determinant of Q
16.☐ # of mixed products (binary·continuous) nonzero (clearly *non-diagonal*) coefficients in Q, as ratio on total # of entries (diagonal included, i.e. n^2)
17.☐ # of products (continuous·continuous) nonzero *non-diagonal* coefficients in Q, as ratio on total # of entries (i.e. n^2)
18.☐ Ratio of biggest nonzero diagonal coefficient on smallest nonzero matrix coefficient (a kind of measure for diagonal dominance), in absolute values
19.☐ Averaged "diagonal dominance" (on rows), defined as $\frac{1}{n}\sum_{i=1}^{n}\left(|q_{ii}| - \sum_{j\neq i}|q_{ij}|\right)$
20.☐ Trace of Q, i.e. $tr(Q) = tr(\Lambda) = \sum_{i=1}^{n}\lambda_i$
21.☐ Spectral norm of Q, i.e. $\|Q\| = \max_i |\lambda_i|$

Dynamic features

22.☐ Difference of lower bounds with L and NL at root node
23.☐ Difference of resolution times of the root node, with L and NL

MIQPs -Learning - Features importance

- Ensemble methods based on Decision Trees provide an importance score for each feature. The scores are in $[0, 1]$ and sum to 1.
- We compare scores for RF, EXT and GTB learning algorithms.



MIQPs - Labeling procedure - Pseudocode (1)

Notation

- `ex.lp` the instance to be labeled, $s(i)$, $i = 1, \ldots, 5$ random seeds
- $l(i, L), l(i, NL), u(i, L), u(i, NL)$ lower and upper bounds obtained in mode L and NL with seed $s(i)$
- $t(i, L), t(i, NL)$ running times of `ex.lp` in mode L and NL with seed $s(i)$
- $status(i, L), status(i, NL)$ the solution status for `ex.lp` solved in mode L and NL with seed $s(i)$

IN: **bounds** $l(i, L), l(i, NL), u(i, L), u(i, NL)$, **times** $t(i, L), t(i, NL)$, **statuses** $status(i, L), status(i, NL)$ for $i = 1, \ldots, 5$
OUT: a label among $\{\mathrm{L}, \mathrm{NL}, \mathrm{T}\}$ for `ex.lp`

Input data collection

1. for each $s(i)$ i=1..5:
2. set mode L
3. solve ex.lp with (i, L)
4. store $l(i, L), u(i, L), t(i, L), status(i, L)$
5. set mode NL
6. solve ex.lp with (i, NL)
7. store $l(i, NL), u(i, NL), t(i, NL), status(i, NL)$

MIQPs - Labeling procedure - Pseudocode (2)

Solvability check

```
 8. if [status(i, L)!= 101,102 AND status(i, NL)!=101,102 ∀ i=1..5]:
 9.     discard the instance
10.     # never solved within timelimit by any mode and seed
```

Seed consistency check

```
11. for i=1..5:
12.     if [(u(i, L) − l(i, NL) < −2 · 10^−4 max{1, |u(i, L)|, |l(i, NL)|})
              OR (u(i, NL) − l(i, L) < −2 · 10^−4 max{1, |u(i, NL)|, |l(i, L)|})]:
13.         discard the instance
14.         break   # stop checking the other seeds
```

Global consistency check

With $l(\cdot) := \max\{l(i,\cdot), i = 1,\ldots,5\}$, $u(\cdot) := \min\{u(i,\cdot), i = 1,\ldots,5\}$

```
15. if [(u(L) − l(NL) < −2 · 10^−4 max{1, |u(L)|, |l(NL)|})
          OR (u(NL) − l(L) < −2 · 10^−4 max{1, |u(NL)|, |l(L)|})]:
16.     discard the instance
```

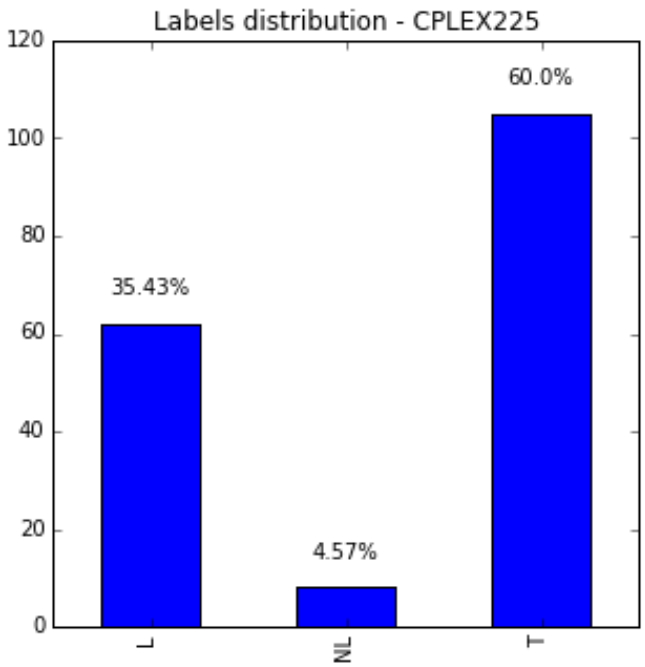
MIQPs - Labeling procedure - Pseudocode (3)

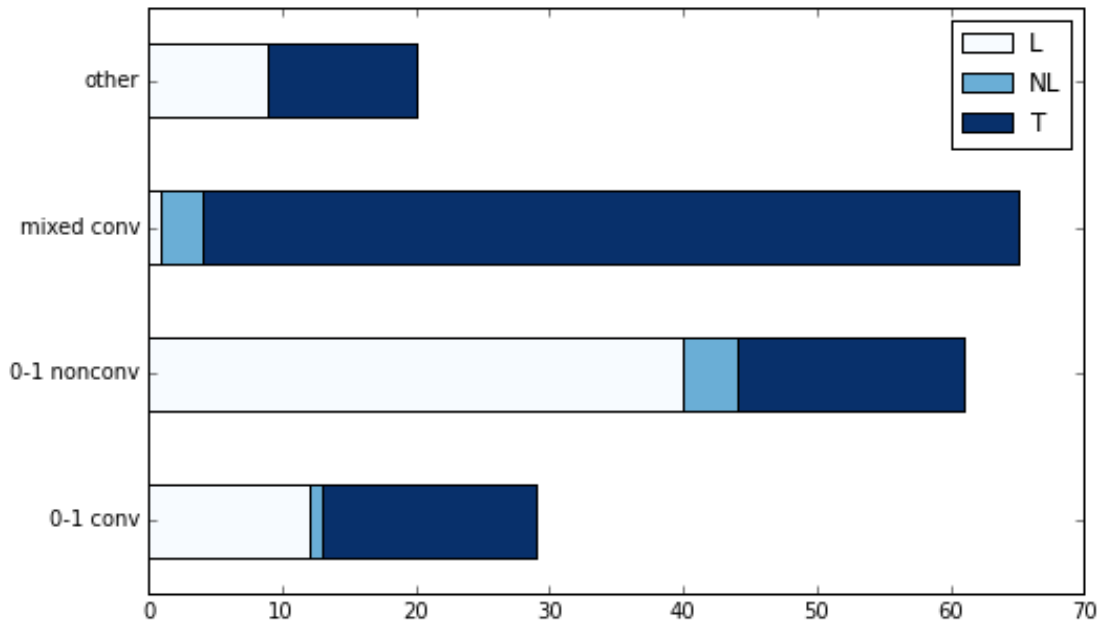
Winner label assignment

```
1.  w(L), w(NL) = 0
2.  for i=1..5:
3.      if t(i, NL) − t(i, L) > x · t(i, NL):      # t(i, L) < t(i, NL) case
4.          w(L) += 1
5.      elif t(i, L) − t(i, NL) > x · t(i, L):     # t(i, L) < t(i, NL) case
6.          w(NL) += 1
7.  if w(L)-w(NL)≥ Δ:
8.      assign label L
9.  elif w(NL)-w(L)≥ Δ:
10.     assign label NL
11. else:
12.     assign label T
```

- Running time is the ultimate compared measure, assessing the final label for each example
- In our dataset $x = 0.1$ and $\Delta = 3$

MIQPs - Nutshell analysis of partial CPLEX testbed





- Note: this partial dataset is not yet fully cleaned (see: “other”)
- Again unbalanced data, but in a completely different way!
- However, the labels distribution could explain the diverse results that we got with respect to classification and optimization quality measures.