Learning MILP resolution outcomes before reaching time-limit

CPAIOR · Thessaloniki, Greece · June 6, 2019

Martina Fischetti[1], Andrea Lodi[2], **Giulia Zarpellon**[2]

[1] Vattenfall, Denmark

[2] Polytechnique Montréal, CERC Data Science for real-time Decision Making

On MILP outcomes

$$\textbf{MILP}: \quad \min\{c^T x : Ax \geq b, x \geq 0,\ x_i \in \mathbb{Z}\ \forall i \in \mathcal{I}\}$$

Decades of huge improvements in MILP solving techniques,

... but it could still require hours of computations!

Fair questions on the resolution process and its outcome:

How much time will it take?

Why does it take so much time?

$\longrightarrow$ ***Can it be solved within a given time?***

- Enforce a **time-limit** TL, but get a sense of the optimization trend after only a fraction of TL has passed
- Ideally, tailor the remaining time in a strategic/flexible way

On MILP outcomes – Our question

Given a MILP instance P and a time-limit TL, look at the partial resolution of P, up to a certain time $0 < \tau < TL$.

Will P be solved to proven optimality within TL?

Use **machine learning** statistical tools to

- summarize and **describe the partial resolution** of P,
  - ! Measure MILP optimization progress
  - ! Complex and sequential nature of B&B data
- cast a prediction about it being solved or not within given TL, in a standard **binary classification** framework
  - ! Get enough and heterogeneous data for learning

Plan

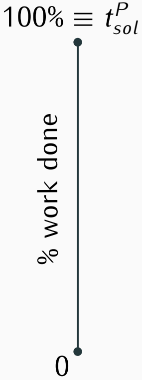
1. Brief formalization
2. MILP data: collection and design
3. Some experiments
4. Outlook

Brief formalization

Measuring the 'work done'

Parameters: a MILP problem $P \in \mathcal{P}$, a time-limit $TL \in \mathbb{R}_{>0}$, and a percentage ratio $\rho \in [0, 100]$, yielding $\tau = \rho \cdot \frac{TL}{100} \in (0, TL)$

e.g., $TL = 3600$ secs, $\rho = 20\% \longrightarrow \tau = 720$ secs

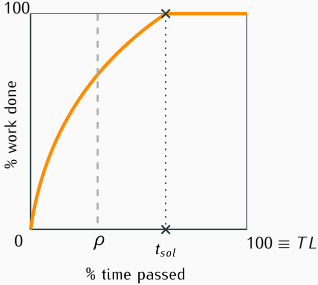


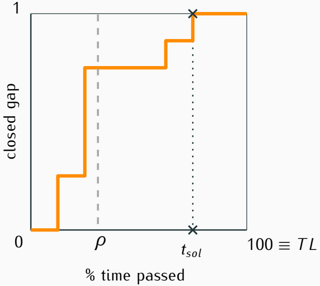
- Evaluate the progress in solving P as ***% work done***
- Reach 100% at t_{sol}^{P} (P solved to proven optimality)

- Reach 100% at TL (total available time)
- Measure the work done up to time τ

Measuring the 'work done' with features

Example: using the *closed gap* as unique *progress measure*





Predict whether $t_{sol}^{P} \leq TL$, describing the work done with

$$\Phi : \mathbb{R}_{>0} \times [0, 100] \times \mathcal{P} \longrightarrow \mathbb{R}^{d} \quad \textbf{feature map}$$

$$(TL, \rho, P) \longmapsto \text{\% work done up to } \tau$$

Feature-based sequence classification

Given B&B **sequential nature**: partial resolution of P as stream of information and events

$\longrightarrow$ ***Multivariate time series*** - ***nodes*** *discretize time dimension*

$$\left.\begin{array}{c} (N^1, \langle v_1^1, \cdots, v_s^1 \rangle) \\ (N^2, \langle v_1^2, \cdots, v_s^2 \rangle) \\ \vdots \\ (N^\eta, \langle v_1^\eta, \cdots, v_s^\eta \rangle) \end{array}\right\} \mathbf{S}_{\mathbf{TL},\rho,\mathbf{P}} \rightsquigarrow \Phi(TL, \rho, P) \in \mathbb{R}^d$$

$\longrightarrow$ ***Feature-based*** *sequence classification task*

Learn **classifier** f for **sequence** $S_{TL,\rho,P}$ with **label** $y \in \{0, 1\}$,

$$f(\Phi(TL, \rho, P)) = \begin{cases} 1 & \text{if } t_{sol}^P \leq TL, \\ 0 & \text{otherwise.} \end{cases}$$

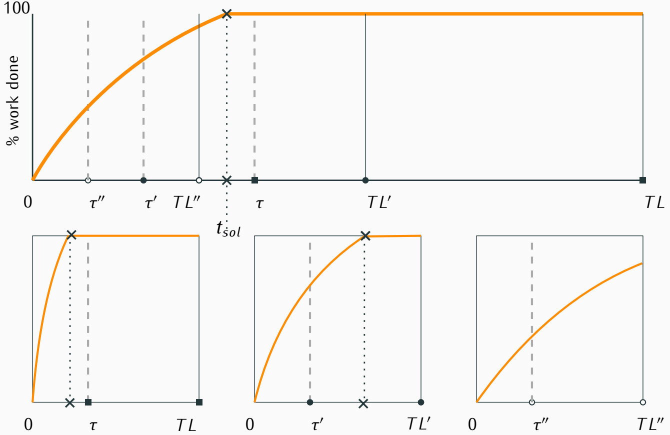
MILP data: collection and design

Producing (enough and) heterogeneous data

To get **multiple data-points** from the same problem P,

(0) vary random seed

(I) vary TL and keep ρ fixed (II) vary ρ and keep TL fixed



$t_{sol} < \tau < TL$, label 1 $\tau' < t_{sol} < TL'$, label 1 $\tau'' < TL'' < t_{sol}$, label 0

MILP data collection

Measuring the 'work done', i.e., MILP progress
requires $S_{TL,\rho,P}$, i.e., **basic B&B data**, and
*comes with a **computational overhead**:*

- might be acceptable from user-perspective:
  spend resources up to τ, to predict on *lengthier horizon TL*
- should *not bias labeling* and invalidate data!

$\longrightarrow$ We opt for a **2-step *proof-of-concept*** implementation:

Step 1. Run P with `CPLEX` with TL and assign label depending on t^P_{sol}.
Record $\#$ of nodes η at time τ.

Step 2. Reproduce the same run, actively collect data up to η nodes.

$\longrightarrow$ **Offline** supervised learning

From raw MILP time series

In practice: 25 raw indicators from each node,

- **Global state** - *gap, best bound, incumbent, nodes*
- **Local (node) state** - *LP objective, iinf, depth*
- **List of open nodes** (only few times) - *length, estimates*

To get features as ***progress measures:***

! Local info should be **combined**

e.g., use nodes' depths to describe tree profile and backtracks

! Global info should be **interpreted** - *development perspective*

e.g., measure quality and distance of bounds updates

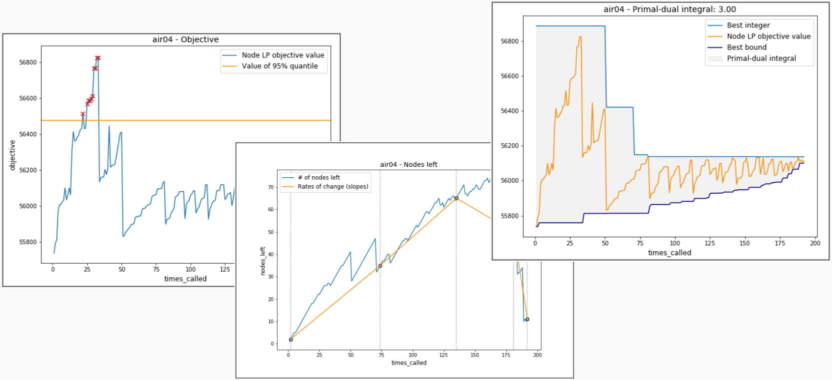
! Measures should be **normalized** across MILP instances

e.g., features on objectives need to be *comparable*

$\longrightarrow$ Make use of ***throughputs*** and statistical functions

$\longrightarrow$ **Domain-knowledge is a key** aspect of feature design!

Features describing serial MILP data



We develop and select **37 features** for learning

$\longrightarrow$ A **single feature** might carry information about **multiple events**!

Some experiments

Dataset composition

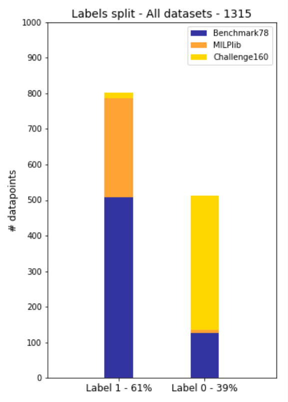
	# pb.s	# seeds
Benchmark* MIPLIB2010	78	3
MILPlib Mittelmann	48	3
Challenge* MIPLIB2010	160	1

* *Primal* and *Infeasible* removed

Initial assessment of runtimes suggested:

$$TL \in \{1200, 2400, 3600\},\ \rho = 20\%$$

to get a balanced dataset.



$\longrightarrow$ **1315** data-points *(will be further reduced)*

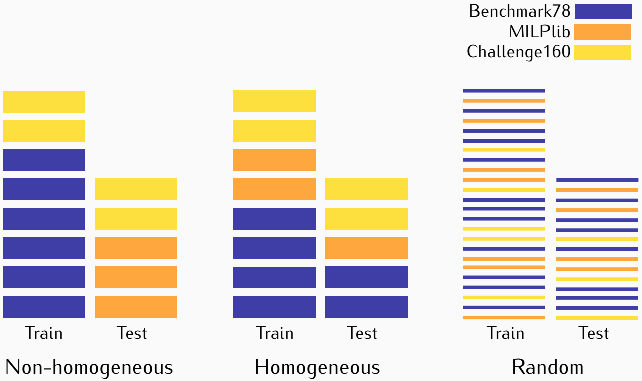
$\longrightarrow$ **Label split**: Class 1 **61% - 39%** Class 0

Learning setting – Train/test split

! *Multiple points* in the dataset come from the *same instance*

! They might resemble each other if *variability* is low

$\longrightarrow$ We test 3 different **train/test splits:**



Learning setting – Models and method

- All train/test splits have comparable labels proportion
- Data **cleaning**: remove missing values $\longrightarrow$ **970** data-points

- We experiment with **5 classifiers:**

`LogReg`	Logistic Regression
`SVM`	Support Vector Machines
`RF`	Random Forest
`ExT`	Extremely Randomized Trees
`MLP`	Multi-layer Perceptron
`Dummy`	Dummy classifier

- Each feature is **normalized** to have 0-mean 1-variance
- Cross-validation to grid-search hyper-parameters

$\longrightarrow$ Implementation tool: `scikit-learn`

Classification results summary

	Dummy	LogReg	SVM	RF	ExT	MLP
Non-homogeneous split						
Accuracy	0.55	0.94	0.94	**0.96**	0.96	0.91
F1-score	0.66	0.96	0.96	**0.97**	0.97	0.94
Homogeneous split						
Accuracy	0.59	0.90	0.91	0.94	**0.95**	0.86
F1-score	0.69	0.93	0.93	0.95	**0.96**	0.89
Random split						
Accuracy	0.57	0.93	**0.94**	0.94	0.93	0.93
F1-score	0.67	0.94	0.95	**0.95**	0.95	0.95

$\longrightarrow$ Overall, `RF` and `ExT` are best performing

. . . bonus interpretability!

Top-scoring features

Ranking	Score	Feature description
1 $*$	0.1856	*Pruned throughput, over processed nodes*
2 $*$	0.1839	*Pruned throughput, over nodes left*
3 $*$	0.0805	*Last seen nodes left / max nodes left*
4 $*$	0.0758	*Proportion of nodes at max objective in open nodes list*
5 $*$	0.0632	*Proportion of nodes at min objective in open nodes list*
6 $*$	0.0622	*Frequency of backtracks*
7 $*$	0.0453	*Frequency of best bound updates*
8 $*$	0.0324	*Last measured gap*
9	0.0250	*Max length of observed dives*
10	0.0211	*Best bound value / best integer value*
11	0.0208	*Distance from last best bound update, normalized*
12	0.0199	*| Best bound - value of objective 5% quantile |*

Top scoring features for `RF`, averaged across split cases.

Outlook

Wrap-up

- Prediction on **MILP outcome**, after only a share of the available time has passed
- **Feature-based** sequence classification task
  - ! Translate MILP progress into a feature vector
  - ! Produce heterogeneous and meaningful data
- Proof-of-concept experiments show that ***there is a statistical pattern to be learned***
- Key **features reflect know-how** and MILP practitioners' experience

Discussing what's next

- Deepen **data analysis** *(too easy? not diverse enough?)*, and frame the role of performance variability

  $\longrightarrow$ Enlarge MILP dataset, play more with parameters TL, ρ

- Consider other ways to tackle sequence classification, e.g., a **pattern-based** method

  $\longrightarrow$ Characterize and detect frequent, early and distinctive patterns in MILP resolution

- Focus on fewer indicators/patterns to move to an **online learning** framework

  $\longrightarrow$ On-the-fly prediction/detection to tailor the use of the remaining time

Thanks! Questions?

Minimal references

- Achterberg T and Wunderling R (2013) Mixed Integer Programming: Analyzing 12 Years of Progress.
- Klotz E and Newman AM (2013) Practical Guidelines for Solving Difficult Mixed Integer Linear Programs.
- Xing Z et al. (2010) A Brief Survey on Sequence Classification.
- Bishop CM (2006) Pattern Recognition and Machine Learning.