



## The MIP Workshop 2023 Computational Competition on Reoptimization

Suresh Bolusani, Mathieu Besançon, Ambros Gleixner, Timo Berthold,  
Claudia d'Ambrosio, Gonzalo Muñoz, Joseph Paat, Dimitri Thomopoulos

### ► To cite this version:

Suresh Bolusani, Mathieu Besançon, Ambros Gleixner, Timo Berthold, Claudia d'Ambrosio, et al..  
The MIP Workshop 2023 Computational Competition on Reoptimization. 2023. hal-04314780

**HAL Id: hal-04314780**

**<https://hal.science/hal-04314780>**

Preprint submitted on 29 Nov 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# THE MIP WORKSHOP 2023 COMPUTATIONAL COMPETITION ON REOPTIMIZATION

A PREPRINT

Suresh Bolusani<sup>\*1</sup>  Mathieu Besançon<sup>1</sup>  Ambros Gleixner<sup>1,2</sup>  Timo Berthold<sup>3</sup>   
 Claudia D'Ambrosio<sup>4</sup>  Gonzalo Muñoz<sup>5</sup>  Joseph Paat<sup>6</sup>  Dimitri Thomopulos<sup>7</sup> 

<sup>1</sup> Zuse Institute Berlin, Berlin, Germany

<sup>2</sup> HTW Berlin, Berlin, Germany

<sup>3</sup> TU Berlin & FICO, Berlin, Germany

<sup>4</sup> LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

<sup>5</sup> Universidad de O'Higgins, Rancagua, Chile

<sup>6</sup> University of British Columbia, Sauder School of Business, Vancouver, Canada

<sup>7</sup> Università di Pisa, DESTEC, Pisa, Italy

\*E-mail: bolusani@zib.de

24 November 2023

## ABSTRACT

This paper describes the computational challenge developed for a computational competition held in 2023 for the 20<sup>th</sup> anniversary of the Mixed Integer Programming Workshop. The topic of this competition was reoptimization, also known as warm starting, of mixed integer linear optimization problems after slight changes to the input data for a common formulation. The challenge was to accelerate the proof of optimality of the modified instances by leveraging the information from the solving processes of previously solved instances, all while creating high-quality primal solutions. Specifically, we discuss the competition's format, the creation of public and hidden datasets, and the evaluation criteria. Our goal is to establish a methodology for the generation of benchmark instances and an evaluation framework, along with benchmark datasets, to foster future research on reoptimization of mixed integer linear optimization problems.

**Keywords** Mixed integer optimization · Reoptimization · Warm starting · Computational optimization

## 1 Introduction

The Mixed Integer Programming Workshop<sup>1</sup> is an annual single-track workshop highlighting the latest trends in mixed integer optimization and discrete optimization. Since the computational development of optimization tools is a crucial component within the mixed integer linear optimization community, the workshop established a computational competition in 2022 to encourage and provide recognition to the development of novel practical techniques in the software for solving mixed integer linear optimization problems (MILPs). The first edition of the competition focused on *primal heuristics*, i.e., finding good quality primal solutions of general MILPs selected mainly from the MIPLIB 2017 benchmark library [15]. This paper discusses the second edition of the competition held in 2023 [1]. From an organizational point of view, this edition involved not only the development of the competition topic, its structure, and an evaluation framework but also the creation of new benchmarking instances.

Traditional benchmarks for MILPs often focus on the performance of optimizing a given instance from scratch. In many practically relevant settings, however, MILP solvers are used to repeatedly solve a series of similar instances of the same type and only slight modifications of the input data. This motivated the 2023 competition topic: the development of effective techniques for reoptimization, also known as warm starting, of MILPs in this setting.

<sup>1</sup><https://www.mixedinteger.org/#mipworkshops>

In addition to the use case where practitioners solve the same MILP model with slightly perturbed input data, this setting also appears algorithmically:

- Row generation algorithms, for instance based on generalized Benders' decompositions for single- and multilevel MILPs (e.g., bilevel optimization problems), solve MILP subproblems that vary only in the right-hand side vector across iterations, see, e.g., [10].
- Column generation algorithms, for instance based on a Dantzig-Wolfe decomposition for MILPs, solve subproblems (pricing problems) that vary only in the objective vector across iterations, see, e.g., [26].
- Scenario decomposition algorithms for stochastic MILPs solve subproblems that vary only in the scenario-dependent components both within and across iterations, see, e.g., [16].
- Primal heuristics, such as diving and neighborhood heuristics, may solve similar MILPs with varying input data both within and across calls to the heuristics, see, e.g., [14].

Despite the broad applicability of reoptimization of MILPs, the research in this area is limited. For example, [11] discusses a reoptimization algorithm for solving the shortest path problem with time windows in a dynamic programming setting, [6, 23] develop frameworks for reoptimization of combinatorial optimization problems and derive specific algorithms for certain classes of such problems, and [21, 26] discuss reoptimization techniques for general MILPs but only when specific components of the MILP vary from one instance to another. Most of the existing literature in this area either deals with specific applications or needs to be more scalable to be applied in practice in the reoptimization settings mentioned above. This further motivated the competition topic.

To evaluate reoptimization for the different settings mentioned above, the competition provided a set of MILP instance series, each of which comprised 50 related instances of the same size. For each instance series, the type of change that may occur and the names of varying columns and/or rows as applicable for this type are known.

The competition participants were asked to provide a general solver to optimize a series of related MILPs in sequential order, thereby reusing information from the previous runs in the series. The participants were free to build on any software that is available in source code, and that can be used freely for the evaluation of the competition. The intention of this competition was not to perform offline training on different types of applications but to reuse information from current and previous solving processes to accelerate the solution of future instances.

The remainder of the paper has the following structure. Section 2 introduces the competition dataset, mentions the two GitHub repositories in which this dataset and its generation scripts are available, discusses the dataset creation process and the two metrics used for this purpose, and provides detailed explanations of all series of instances. Section 3 discusses the competition's evaluation criteria and presents a novel scoring function developed to measure the computational performance of the submissions. Finally, Section 4 presents the competition results, concluding remarks, and future outlook.

## 2 The Dataset

The competition dataset consists of a set of MILP instance series of 50 related instances each. Each instance series is based on an MILP taken from a specific application or benchmark library in the literature. There are a total of seven public and five hidden instance series with constant constraint matrix, and three additional instance series with varying constraint matrix that were not part of the competition evaluation.

The instances in each series comply with the following specifications:

- the number of constraints, and the number, order, and meaning of variables remain the same across the instances in a series, and
- some or all of the following input can vary:
  - objective function coefficients,
  - variable bounds,
  - constraint sides, and
  - coefficients of the constraint matrix.

Due to the more challenging nature of constraint coefficient changes, the corresponding three series were not part of the official computational evaluation. They were nonetheless included to see if some proposed approaches were applicable and efficient for these series.

Table 1 summarizes all the instance series where

Table 1: The Dataset

| Instance series           | Varying component |    |     |     |     |     | Time limit |
|---------------------------|-------------------|----|-----|-----|-----|-----|------------|
|                           | LO                | UP | OBJ | LHS | RHS | MAT |            |
| <i>Public</i>             |                   |    |     |     |     |     |            |
| bnd_series_1              |                   | ✓  |     |     |     |     | 600 s      |
| bnd_series_2              | ✓                 | ✓  |     |     |     |     | 300 s      |
| obj_series_1              |                   |    | ✓   |     |     |     | 400 s      |
| obj_series_2              |                   |    | ✓   |     |     |     | 300 s      |
| rhs_series_1              |                   |    |     | ✓   | ✓   |     | 400 s      |
| rhs_series_2              |                   |    |     |     | ✓   |     | 60 s       |
| rhs_obj_series_1          |                   |    | ✓   | ✓   | ✓   |     | 500 s      |
| <i>Hidden</i>             |                   |    |     |     |     |     |            |
| bnd_series_3              | ✓                 | ✓  |     |     |     |     | 600 s      |
| obj_series_3              |                   |    | ✓   |     |     |     | 350 s      |
| rhs_series_3              |                   |    |     | ✓   | ✓   |     | 550 s      |
| rhs_series_4              |                   |    |     | ✓   | ✓   |     | 60 s       |
| rhs_obj_series_2          |                   |    | ✓   | ✓   | ✓   |     | 250 s      |
| <i>Out of competition</i> |                   |    |     |     |     |     |            |
| mat_series_1              |                   |    |     |     |     | ✓   | 300 s      |
| mat_rhs_bnd_series_1      | ✓                 | ✓  |     | ✓   | ✓   | ✓   | 400 s      |
| mat_rhs_bnd_obj_series_1  | ✓                 | ✓  | ✓   | ✓   | ✓   | ✓   | 180 s      |

- LO, UP, OBJ, LHS, RHS, and MAT, denote lower bound vector, upper bound vector, objective function vector, left-hand side vector, right-hand side vector, and the constraint matrix, respectively, and
- the last column states the time limit imposed to solve one instance in the series.

To the best of our knowledge, there is currently no dedicated reference benchmark for the reoptimization of MILPs. We, therefore, created these series of instances based on various algorithms, applications, and existing MILP instances from the literature and made them available at the GitHub repository [1]. Furthermore, the scripts used to generate these series are also available at another GitHub repository [2], offering a template for generating more and longer series for future research beyond the context of the competition.

To build a single series, we generated numerous instances satisfying the series requirements, solved them to optimality from scratch, and selected 50 suitable instances for the series by applying the following two metrics.

1. The time taken to solve an instance to optimality.
2. The similarity between the varying components of the series, whenever applicable. The similarity between two vectors  $c$  and  $\bar{c}$  is defined as the angle between the vectors [26].

$$\text{Similarity} = \frac{\langle c, \bar{c} \rangle}{\|c\| \|\bar{c}\|}$$

For example, let  $c$  and  $\bar{c}$  be the objective function vectors of two instances. If the similarity between these vectors is very high, then the effort required to solve these instances to optimality without reoptimization is likely to be similar. Furthermore, most of the primal information generated in the solving process of the instance with the objective vector  $c$  will be valid for the instance with the objective vector  $\bar{c}$  (and vice versa).

As mentioned earlier, the competition aims to encourage reoptimization techniques that can accelerate the solution of future instances in a given series compared to solving these instances from scratch. Accordingly, we chose the time limit for an instance in a given series based on the individual solving times (from scratch) of the 50 instances in this series. Then, the scoring function discussed in Section 3 penalized any techniques that could not solve an instance to optimality within this time limit.

We now discuss the details of each instance series.

**bnd\_series\_1** : This series is based on the instance rococoC10-001000 from the MIPLIB 2017 benchmark library [18]. The instances were generated by perturbing the upper bounds of general integer variables selected via a discrete uniform distribution up to  $\pm 100\%$  of the bound value.

bnd\_series\_2 : This series is based on the instance `csched007` from the MIPLIB 2017 benchmark library [18]. The instances were generated via random fixings of 15% to 25% of the binary variables selected via a discrete uniform distribution w.r.t. the original instance.

bnd\_series\_3 : This series is also based on the instance `csched007` from the MIPLIB 2017 benchmark library [18]. The instances were generated via random fixings of 5% to 20% of the binary variables selected via a discrete uniform distribution w.r.t. the original instance. Accordingly, these instances are relatively harder to solve as compared to the instances in the series bnd\_series\_2 as indicated by the time limits in Table 1.

obj\_series\_1 : This series is based on the stochastic multiple binary knapsack problem and the associated instance set introduced in [5]. The problem has the formulation

$$\begin{aligned} \min_{x,z} \quad & c^\top x + d^\top z + Q(x) \\ \text{s.t.} \quad & Ax + Cz \geq b, \\ & x \in \{0, 1\}^n, z \in \{0, 1\}^n, \end{aligned}$$

where  $Q(x) = \sum_{\omega \in \Omega} p_\omega Q_\omega(x)$ , with

$$\begin{aligned} Q_\omega(x) := \min_y \quad & q_\omega^\top y \\ \text{s.t.} \quad & Wy \geq h - Tx, \\ & y \in \{0, 1\}^n, \end{aligned}$$

where  $\omega \in \Omega$  denotes a scenario,  $p_\omega$  denotes probability of the scenario  $\omega$ , and the second-stage objective vector  $q_\omega$  is random, following a discrete distribution with finitely many scenarios.

We adapted the given dataset and generated instances by considering one scenario at a time. This resulted in a series of instances with one-third of the objective vector (corresponding to  $y$  variables) varying across instances.

obj\_series\_2 : This series is based on the instance `ci-s4` from the MIPLIB 2017 benchmark library [15]. The instances were generated via random perturbations and random rotations of the objective vector.

obj\_series\_3 : This series is based on the UCI Machine Learning repository dataset `magic` [9]. The instances are subproblems of a column generation algorithm for improving decision trees. The final set of instances were generated based on a submission that was received in response to a public call for additional datasets.

rhs\_series\_1 : This series is based on the stochastic server location problem and the associated dataset proposed in [19]. The problem has the formulation

$$\begin{aligned} \min_x \quad & \sum_{i \in \mathcal{I}} c_i x_i - Q(x) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{I}} x_i \leq u, \\ & x_i \in \{0, 1\} \quad \forall i \in \mathcal{I}, \end{aligned}$$

where  $Q(x) = \sum_{\omega \in \Omega} p_\omega Q_\omega(x)$ , with

$$\begin{aligned} Q_\omega(x) := \min_y \quad & \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} (-q_{ij} y_{ij}) + \sum_{i \in \mathcal{I}} g_{i0} y_{i0} \\ \text{s.t.} \quad & \sum_{j \in \mathcal{J}} d_{ij} y_{ij} - y_{i0} \leq \tau_i x_i, \quad \forall i \in \mathcal{I}, \\ & \sum_{i \in \mathcal{I}} y_{ij} = r_{j\omega}, \quad \forall j \in \mathcal{J}, \\ & y_{ij} \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \quad y_{i0} \geq 0, \quad \forall i \in \mathcal{I}, \end{aligned}$$

where  $\omega \in \Omega$  denotes a scenario and  $p_\omega$  denotes probability of the scenario  $\omega$ .

We adapted the given dataset and generated instances by considering 25 scenarios at a time. This resulted in a series of instances with only the right-hand side vector of equality constraints varying across instances.

rhs\_series\_2 : This series is based on a synthetic MILP and the associated dataset proposed in [17]. The problem has the following formulation.

$$\begin{aligned} \min_{x,y} \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b, \\ & l^\top y \leq x \leq u^\top y, \\ & x \in \mathbb{R}^n, y \in \{0, 1\}^n \end{aligned}$$

We adapted the given dataset and generated instances by taking a convex combination of two different RHS vectors.

rhs\_series\_3 : This series is based on the instance `glass4` from the MIPLIB 2017 benchmark library [3]. The instances were generated by perturbing the non-negative LHS and RHS vector components selected via a discrete uniform distribution up to  $\pm 70\%$  of their values.

rhs\_series\_4 : This series is also based on the synthetic MILP and the associated dataset proposed in [17]. We adapted the given dataset and generated instances by taking a convex combination of two different RHS vectors (different than the ones used for generating `rhs_series_2`).

mat\_series\_1 : This series is based on the optimal vaccine allocation problem and the associated dataset proposed in [24]. The problem formulation is

$$\begin{aligned} \min_x \quad & \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} c_{ik} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in \mathcal{J}} x_{ij} = 1, \quad \forall i \in \mathcal{I}, \\ & \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} a_{ij\omega} x_{ij} \leq 1 + M z_\omega, \quad \forall \omega \in \Omega, \\ & \sum_{\omega \in \Omega} p_\omega z_\omega \leq 1 - \alpha, \\ & x_{ij} \in [0, 1], \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \quad z_\omega \in \{0, 1\}, \quad \forall \omega \in \Omega, \end{aligned}$$

where  $\omega \in \Omega$  denotes a scenario,  $p_\omega$  denotes probability of the scenario  $\omega$ , and  $M$  denotes a big-M parameter.

We adapted the given dataset and generated instances by considering 500 scenarios at a time. This resulted in a series of instances with the constraint matrix corresponding to inequality constraints varying across instances.

rhs\_obj\_series\_1 : This series is based on the hydro unit commitment (HUC) problem modeled as an MILP. Considering a fixed hydro valley, the input data potentially changing is restricted to the electricity prices, the inflows, and the initial and target water volume in the reservoirs. These appear only in the objective function or constraint sides. Thus, reoptimizing this problem is practically interesting because the great majority of the input data remains unchanged. Moreover, utility companies often solve the HUC problem as a subproblem of a decomposition method. Consequently, for converging to the optimal solution of the whole unit commitment problem, a HUC has to be solved at each iteration. Detailed mathematical formulation is available in [25].

rhs\_obj\_series\_2 : This series is based on a hydroelectric valley (Ain River) industrial use case in France. Six dams and their different turbines are modeled for the next four days with an hourly time step. The differences across instances are electricity prices (in the objective function) and the varying flows of the different affluents (in the RHS vector) of the river Ain, which were collected from [4, 12]. This series is based on the instances that were received in response to a public call for additional datasets. The final instances were generated by perturbing the LHS, RHS, and objective function vector components, selected via a discrete uniform distribution up to  $\pm 20\%$  of their values.

mat\_rhs\_bnd\_series\_1 : This series of instance are based on the MILP formulation of the multilevel supply chain of a fictitious cell phone company. Detailed description is available at [22].

mat\_rhs\_bnd\_obj\_series\_1 : This series of instances is also based on the hydro unit commitment problem similar to the series `rhs_obj_series_1`, but every data component of the given MILP can vary here.

### 3 Evaluation Criteria

This section discusses the criteria used to evaluate the proposed approaches for the competition. Participants were asked to provide a written report and code base corresponding to their submission. Then, the following two criteria were used for the final evaluation.

1. **Novelty and scope:** innovativeness of the approach and its general applicability w.r.t. the varying components and magnitude of their variation.
2. **Computational excellence:** ranking of the approach in terms of the performance score defined later in this section.

The written report had to include the following:

- description of the methods developed and implemented, including any citations to the literature and software used,
- computational results on the public instance series with constant constraint matrix,
- analysis with at least `reltime`, `gap`, and `nofeas` scores (defined later in the section) averaged over all 50 instances and additionally over the five batches of instances 1 to 10, 11 to 20, 21 to 30, 31 to 40, and 41 to 50, and
- any further analysis including the applicability of the approach to the instance series with varying constraint matrix.

Participants were allowed to use any existing software available in source code and freely usable for the evaluation. A submission was required to solve the instances of one series sequentially in the order specified by the input files. It was not permitted to parse and analyze instances in one series ahead of time, i.e., while solving the  $i^{\text{th}}$  instance in the series, a submission may use only information from the first  $i - 1$  instances. A submission was not allowed to modify the solution for an instance after moving on to solve the next instance. A submission must run sequentially (1 CPU thread), use no more than 16 GB of RAM, and respect the total time limit for each instance series. Violations of the time limit for a single instance are penalized in the performance score.

Evaluating and comparing the performance of submissions necessitated the development of an appropriate scoring metric. Our goal was to create a comprehensive scoring system that takes into account various potential situations, ranked from the most favorable to the least favorable:

- the instance is solved to optimality within a certain runtime, shorter runtimes are preferred;
- the instance times out but provides an incumbent solution, smaller gaps are preferred;
- the instance times out without yielding any feasible solution, which is generally regarded as undesirable.

While some existing measures, such as [7], work across these different situations, we also sought to reflect the aforementioned hierarchy of favorability in our scoring. It was crucial to strike a balance; we wanted to penalize, to a certain extent, failure to achieve optimality or to find a solution, but we also aimed to avoid unduly restricting approaches that engage in an early exploration phase, which might intentionally deteriorate performance for the first few instances of a series.

To achieve this balance, we established a scoring range:

- the instances solved to optimality within the time limit were assigned scores between 0 and 1,
- the instances that timed out with an incumbent solution received scores between 1 and 2, and
- the instances that timed out without any feasible solution were given a score of 3.

Specifically, let  $s = 1, 2, \dots, S$  denote the index of the instance series, each consisting of  $i = 1, 2, \dots, 50$  MILP instances. Then, the performance on a single instance  $(s, i)$  is measured via the scoring function

$$f(s, i) = \text{reltime} + \text{gap} + \text{nofeas},$$

where

$$\text{reltime} = \begin{cases} \frac{\text{time spent}}{\text{time limit}}, & \text{if the instance is solved to optimality,} \\ \max\left\{1, \frac{\text{time spent}}{\text{time limit}}\right\}, & \text{otherwise,} \end{cases}$$



$$\text{gap} = \begin{cases} 0, & \text{if primal bound (pb) and dual bound (db) are 0,} \\ 1, & \text{if pb or db are infinite, or pb} \times \text{db} < 0, \\ \frac{|\text{pb} - \text{db}|}{\max(|\text{pb}|, |\text{db}|)}, & \text{otherwise,} \end{cases}$$

and

$$\text{nofeas} = \begin{cases} 1, & \text{if no feasible solution is returned,} \\ 0, & \text{otherwise.} \end{cases}$$

Smaller scores  $f(s, i)$  are better. In our mind, this scoring framework allows for nuanced evaluation, rewarding efficiency while maintaining fairness across different methodologies. Note that the transition between solving an instance and timing out with a small gap is “smooth” in the sense that the score approaches 1 from below as the solve time gets close to the time limit and 1 from above as the final gap approaches 0 for unsolved instances. In contrast, the term `nofeas` adds a fixed penalty term for not producing a primal solution.

For technical reasons, we also had to address the situations where submissions exceed the time limit (usually only slightly) or stop a solution process prematurely before reaching the time limit. If the submission exceeds the time limit, `reltime` will be larger than 1. Submissions that stop before the time limit without reaching zero gap will receive `reltime` = 1. These considerations were mainly taken to close loopholes for the competition, such as intentionally ignoring a time limit to gather more information or aborting unpromising runs to keep the score low.

Then, for each instance  $(s, i)$ , all participants were ranked according to their score  $f(s, i)$ , resulting in each team receiving a rank  $r(s, i)$ , where smaller is better. Teams with the same score received the same rank. For instances for which the primal solution was not feasible or the dual bound was not valid, a team received two times the worst possible rank independently of their  $f(s, i)$  value.

Following the motivation of the challenge to reward methods that use information from previous solving processes in order to gain performance, we assigned a gradually increasing weight to later instances in each series, i.e., we computed the final score as

$$C = \sum_{s=1}^S \sum_{i=1}^{50} (1 + 0.1i) r(s, i).$$

The lower this final score, the better.

## 4 Results and Outlook

The jury selected one winner and awarded one honorable mention from a total of twelve registrations and six final submissions containing both a written report and implementation code.

The winning submission *Progressively Strengthening and Tuning MIP Solvers for Reoptimization* by Krunal Patel described in detail in [20] convinced the jury not only through its computational excellence that was displayed by the top ranked performance on almost all public and hidden data sets, but also through its broad applicability and attention to algorithmic details. Building on top of an existing, open-source LP-based branch-and-bound solver SCIP [8], the approach distinguished itself by targeting multiple aspects of the solving process in combination, reusing primal information and pseudo costs and improving parameter configurations online despite the limited number of observations.

An honorable mention was awarded to the submission *Influence branching for learning to solve mixed integer programs online* by the team of Paul Strang, Zacharie Ales, Come Bissuel, Olivier Juan, Safia Kedad-Sidhoum, and Emmanuel Rachelson. The jury was impressed by the successful adaptation of influence branching [13] to the reoptimization setting of the competition. The submission employed online hyperparameter tuning of different influence models via multi-armed bandit selection and consistently performed well on both public and hidden datasets.

With the created instance series available on the repository [1] and presented in Section 2, and their extensibility based on the generation scripts in [2], we hope to offer the research community a set of benchmarks to foster and evaluate future research efforts towards reoptimization. We aim to continuously develop [2] as a benchmarking library in the long run for the research on the reoptimization of MILPs. Accordingly, we are expanding the current dataset by generating additional series of instances and making them available to the research community via this repository.

**Acknowledgements** We wish to thank Daniel Bienstock for providing us with computational infrastructure in order to evaluate the submissions, Dimitri Kniasew from SAP for submitting one hidden series of supply chain network planning instances, Felipe Serrano for discussing initial suggestions for the topic of the competition, and Domenico Salvagnin for participating in the final evaluation process.



## Conflict of Interest

The authors declare that they have no conflict of interest.

## References

- [1] MIP Computational Competition 2023. URL <https://github.com/ambros-gleixner/MIPcc23/>. Accessed: 14 November 2023
- [2] MIP Computational Competition 2023 Dataset Generation Scripts. URL <https://github.com/sbolusani/MILP-WS-Lib>. Accessed: 14 November 2023
- [3] Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. *Operations Research Letters* **34**(4), 361–372 (2006). DOI 10.1016/j.orl.2005.07.009. URL <https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/862>
- [4] Andréassian, V., Delaigue, O., Perrin, C., Janet, B., Addor, N.: CAMELS-FR: A large sample, hydroclimatic dataset for france, to support model testing and evaluation (2021). URL <https://doi.org/10.5194/egusphere-egu21-13349>
- [5] Angulo, G., Ahmed, S., Dey, S.S.: Improving the integer L-shaped method. *INFORMS Journal on Computing* **28**(3), 483–499 (2016). URL <https://doi.org/10.1287/ijoc.2016.0695>
- [6] Ausiello, G., Bonifaci, V., Escoffier, B.: Complexity and approximation in reoptimization. In: *Computability in Context: Computation and Logic in the Real World*, pp. 101–129. World Scientific (2011). URL [https://doi.org/10.1142/9781848162778\\_0004](https://doi.org/10.1142/9781848162778_0004)
- [7] Berthold, T., Csizmadia, Z.: The confined primal integral: a measure to benchmark heuristic MINLP solvers against global MINLP solvers. *Mathematical Programming* **188**(2), 523–537 (2021). URL <https://doi.org/10.1007/s10107-020-01547-5>
- [8] Bestuzheva, K., Besançon, M., Chen, W.K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S.J., Matter, F., Mühmer, E., Müller, B., Pfetsch, M.E., Rehfeldt, D., Schlein, S., Schlösser, F., Serrano, F., Shinano, Y., Sofranac, B., Turner, M., Vigerske, S., Wegscheider, F., Wellner, P., Weninger, D., Witzig, J.: The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin (2021). URL <https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/8530>
- [9] Bock, R.: MAGIC gamma telescope. UCI Machine Learning Repository (2007). URL <https://doi.org/10.24432/C52C8B>
- [10] Bolusani, S., Ralphs, T.K.: A framework for generalized Benders’ decomposition and its application to multi-level optimization. *Mathematical Programming* **196**(1-2), 389–426 (2022). URL <https://doi.org/10.1007/s10107-021-01763-7>
- [11] Desrochers, M., Soumis, F.: A reoptimization algorithm for the shortest path problem with time windows. *European Journal of Operational Research* **35**(2), 242–254 (1988). URL [https://doi.org/10.1016/0377-2217\(88\)90034-3](https://doi.org/10.1016/0377-2217(88)90034-3)
- [12] Energetici, G.M.: Historical data day ahead market (2022). URL <https://www.mercatoelettrico.org/En/download/DatiStorici.aspx>
- [13] Etheve, M., Alès, Z., Bissuel, C., Juan, O., Kedad-Sidhoum, S.: Reinforcement learning for variable selection in a branch and bound algorithm. In: E. Hebrard, N. Musliu (eds.) *CPAIOR2020: Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 176–185 (2020). URL [https://doi.org/10.1007/978-3-030-58942-4\\_12](https://doi.org/10.1007/978-3-030-58942-4_12)
- [14] Gamrath, G., Berthold, T., Heinz, S., Winkler, M.: Structure-driven fix-and-propagate heuristics for mixed integer programming. *Mathematical Programming Computation* **11**(4), 675–702 (2019). URL <https://doi.org/10.1007/s12532-019-00159-1>
- [15] Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P.M., Jarck, K., Koch, T., Linderoth, J., Lübbecke, M., Mittelmann, H.D., Ozyurt, D., Ralphs, T.K., Salvagnin, D., Shinano, Y.: MIPLIB 2017: Data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation* **13**(3), 443–490 (2021). URL <https://doi.org/10.1007/s12532-020-00194-3>
- [16] Hassanzadeh, A.: Two-stage stochastic mixed integer optimization. Ph.D. thesis, Lehigh University (2015)

- [17] Jiménez-Cordero, A., Morales, J.M., Pineda, S.: Warm-starting constraint generation for mixed-integer optimization: a machine learning approach. *Knowledge-Based Systems* **253**, 109570 (2022). URL <https://doi.org/10.1016/j.knosys.2022.109570>
- [18] Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelman, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010: Mixed integer programming library version 5. *Mathematical Programming Computation* **3**, 103–163 (2011). URL <https://doi.org/10.1007/s12532-011-0025-9>
- [19] Ntamo, L.: Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations research* **58**(1), 229–243 (2010). URL <https://doi.org/10.1287/opre.1090.0693>
- [20] Patel, K.: Progressively strengthening and tuning MIP solvers for reoptimization. *Mathematical Programming Computation* (2023). Under review
- [21] Ralphs, T., Güzelsoy, M.: Duality and warm starting in integer programming. In: The proceedings of the 2006 NSF design, service, and manufacturing grantees and research conference (2006). URL <http://coral.ie.lehigh.edu/~ted/files/papers/DMII06.pdf>
- [22] SAP SE or an SAP affiliate company: MILP benchmarks cellphoneco. (2023). URL <https://github.com/SAP-samples/ibp-sop-benchmarks-milp-cellphoneco>
- [23] Schieber, B., Shachnai, H., Tamir, G., Tamir, T.: A theory and algorithms for combinatorial reoptimization. *Algorithmica* **80**, 576–607 (2018). URL <https://doi.org/10.1007/s00453-017-0274-8>
- [24] Tanner, M.W., Ntamo, L.: IIS branch-and-cut for joint chance-constrained stochastic programs and application to optimal vaccine allocation. *Eur. J. Oper. Res* **207**(1), 290–296 (2010). URL <https://doi.org/10.1016/j.ejor.2010.04.019>
- [25] Thomopulos, D., d’Ambrosio, C., van Ackooij, W., Stéfanon, M.: Generating hydro unit-commitment instances. *TOP An Official Journal of the Spanish Society of Statistics and Operations Research* (2023). URL <https://doi.org/10.1007/s11750-023-00660-w>
- [26] Witzig, J.: Reoptimization techniques in MIP solvers. Master’s thesis (2014)