MIP Insights

The newsletter of the Mixed Integer Programming Society

TABLE OF CONTENTS

Page 1:

- *Check out what happened in recent months*

  MIP South America 2025 in Viña del Mar, Chile
- *Keep an eye out for upcoming events*

  MIP Europe 2026 in Rome, Italy

  MIP 2026 in Stamford, Connecticut
- *Check out a new challenge from DIMACS*
  13th DIMACS Challenge: Network Flows 2.0
- *Upcoming talks in the* **Discrete Optimization Talks (DOTs) series**

Page 2:

- Read about the history and future of Computational Discrete Optimization, as told by Marc Pfetsch

MIP SOUTH AMERICA 2025

The Mixed Integer Programming Workshop South America 2025 was held from December 9 to 12, 2025, at the Universidad Adolfo Ibáñez campus in Viña del Mar, Chile. With more than 140 participants, the event represented an important step in strengthening regional engagement while showcasing state-of-the-art research in mixed-integer programming, discrete optimization, and operations research.

A highlight of the workshop was the poster session, which showcased high-quality student work. Honorable Mentions were awarded to Felipe Keim (Universidad de Chile), Domingo Araya (Pontificia Universidad Católica de Chile), Daniel Yamin (Carnegie Mellon University), and Macarena Fredes (Universidad de Santiago de Chile). The Runner-Up distinction went to Macarena Navarro (Carnegie Mellon University), and the MIP Workshop Best Poster Award was presented to Sebastián Vásquez (Carnegie Mellon University).

The Organizing Committee consisted of Víctor Bucarey (Universidad de O'Higgins, Chile), Rodolfo Carvajal (Universidad Adolfo Ibáñez), Margarida Carvalho (Université de Montréal, Canada), Andrés Gómez (University of Southern California, USA), Javier Marenco (Universidad Torcuato Di Tella, Argentina), Gonzalo Muñoz (Universidad de Chile, Chile), and Eduardo Uchoa (Universidade Federal Fluminense, Brazil).

The workshop benefited from the generous support of its sponsors and partners. Primary support was provided by the Department of Industrial Engineering of Universidad de Chile and Hexaly, with additional contributions from Google, the Institute of Engineering Sciences of Universidad de O'Higgins, Alicanto Labs, the Universidad Adolfo Ibáñez Business School, Cardinal Optimizer, the Institute for Mathematical and Computational Engineering of Pontificia Universidad Católica de Chile, and the Institute for Complex Engineering Systems.

Overall, MIP South America 2025 marked an important milestone for the mixed-integer programming community in the region and laid a strong foundation for future editions of the workshop.

THE 2026 MIP WORKSHOP

The 2026 Mixed Integer Programming Workshop will take place on May 18-21, 2026, at the University of Connecticut, Stamford campus. Registration is open, with early registration ending on April 14.

This will be the twenty-third edition of the series. MIP 2026 will carry on the traditions that have made this workshop a central gathering place for our community, and we invite you to join us for:

- A single track of 21 invited talks from experts across a wide range of topics topics in the theoretical, computational, and applied aspects of mixed integer programming and discrete optimization.
- A poster session, including a best-poster competition among selected student finalists.
- The 2026 Land-Doig MIP Competition on GPU-Accelerated Primal Heuristics for MIP.
- Contributed flash talks. More information coming up soon.
- The Mixed Integer Programming Society (MIPS) business meeting to be held on Thursday afternoon.

For updates and more information, please visit the website.

We look forward to seeing you in Stamford, Connecticut!

Program committee: Beste Basciftci, Yatharth Dubey, Cheng Guo, Sebastian Perez-Salazar, Matthias Walter

Local committee: David Bergman, Carlos Cardonha, Robert Day, Nicholas Lownes, Laurent Michel, Matthew Stuber, Bin Zou

Competition Committee: Beste Basciftci, Akif Çördük, Gerald Gamrath, Christopher Hojny, Jan Kronqvist, Haihao Lu, Christian Tjandraatmadja

MIP EUROPEAN WORKSHOP 2026

The 2nd MIP European Workshop (EURO-MIP 2026) will take place 19–21 October 2026 in Rome, Italy, hosted by the DIAG Department (Department of Computer, Control and Management Engineering), Sapienza University of Rome. The workshop is part of the MIP International Workshop series and will feature a single-track program on recent advances and open challenges in mixed-integer programming and discrete optimization. We will strongly support junior researchers and students, including a poster session and/or a short "junior spotlight" segment, and aim to offer a limited number of student travel grants (subject to sponsorship). For more information, please see the workshop's website.

Program Committee: Fabio Furini, Veronica Piccialli, Ivana Ljubic, Claudia D'Ambrosio, Christopher Hojny

Local committee: Fabio Furini, Veronica Piccialli, Fabio Ciccarelli

DIMACS CHALLENGE: NETWORK FLOWS 2.0

The 13th DIMACS Implementation Challenge: Network Flows 2.0 is currently underway and invites both participation in the computational competition and contribution of test instances. The over-arching purpose of an Implementation Challenge is to assess the practical performance of algorithms for a particular problem class, while moving them closer to what is theoretically possible and fostering interactions among contributing researchers.

This Challenge focuses on the minimum-cost flow problem and its variants and special cases, including: minimum-cost flow problem; transportation problem; assignment problem; maximum-flow problem; bipartite matching problem; and global minimum cut problem.

The organizing committee for this challenge includes Andrew V. Goldberg (chair), Tamra Carpenter, Giuseppe F. Italiano, Loukas Georgiadis, Daniel A. Spielman, Clifford Stein, and David P. Williamson.

For information on how to participate please visit: https://coral.ise.lehigh.edu/flow-challenge-2-0/.

DISCRETE OPTIMIZATION TALKS (DOTS)

The Mixed Integer Programming Society supports Discrete Optimization Talks (DOTs), a virtual seminar series on all aspects of integer and combinatorial optimization. This semester, DOTs will be take place on the last Friday of the month at 12:00pm Eastern Time over Zoom.

Visit our website to find information on the Spring 2026 season of DOTs and view recordings of previous talks. To receive the link to participate, join the mailing list and add "lists@mixedinteger.org" to your approved addresses. If you are interested in giving a DOT, let us know. We look forward to seeing you!

On the Future of Computational Discrete Optimization Research

By Marc E. Pfetsch, Department of Mathematics, TU Darmstadt, Germany

Introduction

Computations have been an important cornerstone of Discrete Optimization since its beginnings in the middle of the 20th century. In this article, we discuss some of the challenges that this area currently faces. These challenges are motivated by the following recent anecdotic observations:

- The focus of many (maybe the majority of) new articles and conference talks that involve computations is on modeling aspects, i.e., models are presented and discussed. Then they are inserted into a black-box global solver to derive conclusions about the quality of the models.
- With very few exceptions, a single commercial solver ("the best solver in the world") is used, both for exact solution as well as a heuristic.
- The number of researchers that generate significant implementations seems to be declining.

This article will argue in the following that these issues are connected.

Clearly, these observations are somewhat subjective. However, if the they represent a trend, this will have consequences on computational work in Discrete Optimization, e.g., diminishing the role of such work in the long run. Whether a decline of such scientific computational work is problematic or not, depends on the viewpoint, but this article will present arguments why significant scientific computational work is needed in – or at least is advantageous for – the Discrete Optimization research areas. Moreover, this article will try to identify reasons for the occurrence of the above observations and propose some remedies.

Research Areas In this article, Discrete Optimization broadly includes the areas of mixed-integer programming (MIP), mixed-integer nonlinear programming (MINLP), and combinatorial optimization (CO). There are many intersections between them, but also specific challenges. The main focus in the following is on computing globally optimal solutions (in particular the historical review below), but most comments apply to heuristics as well.

Similar considerations as the ones in this article might be valid for other areas of mathematical optimization and beyond. However, this documents restricts attention to Discrete Optimization, because this is the area best known to the author. Moreover, one unique property of Discrete Optimization seems to be the strong influence of commercial software on scientific computational work.

Why Computational Research is Important

Before discussing the importance of computational research, we should first recall some basics.

Scientific computational research means that new *algorithms* are developed, implemented, and tested on meaningful test instances.[1] Thus, computational research refers to solution algorithms, not just models. Its main goal is to *understand* the underlying algorithmic implications. Computations contribute the important feature of deriving conclusions from empirical data. To this end, computational experiments should attempt to be generalizable, which mandates a good choice of (large) test sets. Furthermore, to make the results reproducible, a clear algorithmic description is necessary and, if possible, data and source code should be publicly available.

The following arguments for the importance of computational research are probably well known, but it might help to reconsider them here.

- Implementations can help to identify errors in theoretical results (see, e.g., Sanders [25] for examples) and theory can improve implementations.
- Computational work and its experimental data often call for a theoretical foundation and conversely, one may (or should!) ask whether theoretical arguments have real-life manifestations.
- MIP-solver implementations are an important economical factor. One sign for this is the success and increase – both in number and size – of commercial software companies.
- Discrete Optimization techniques can also be used in other areas of research, e.g., as subroutines for deciding whether certain discrete structures exist. Thus, (computational) methods from this area enable research in other areas. Often new techniques need to be developed in order to fulfill this role, for example, verifiable and numerically exact results; see, e.g., Cheung et al. [7].
- Computational Discrete Optimization is a very well established line of research (see the next section), although it may not often be visible in isolation. Indeed, it seems that there are fewer workshops/conferences that focus on computational work than there are purely theoretical meetings.

All these goals will only be supported in the future, if there is scientific progress on computational work.

A Brief Historical Overview

It seems helpful to briefly recall some history on computational Discrete Optimization research. Note that this overview concentrates on certain aspects and cannot be comprehensive. More details are given, for example, in the excellent review of Bixby [6].

One of the cornerstones in Discrete Optimization is undoubtedly the simplex algorithm of Dantzig [12] for solving linear programs (LPs). One of the main reasons for its success is the ability to solve linear programs in practice. With the advent of computers, starting in the 1950s, implementations were used to solve ever growing linear programs. A big jump in the use of the simplex algorithm came with the "personal computer" in the 1980s and then in the 1990s where the dual simplex algorithm was the workhorse of branch-and-cut algorithms. And it has stayed like this for modern MIP- and MINLP-solvers. In hindsight, a computational solution of LPs (including alternatives like interior point algorithms) has always been an important computational research topic since the early beginnings.

Computational work in combinatorial optimization has a long history as well. A first important milestone was the 1954 paper of Dantzig, Fulkerson, and Johnson [11] on the solution of the traveling salesman problem (TSP). Indeed, the TSP has stayed *the* showcase of the success of computational work: we refer to the book of Applegate, Bixby, Chvátal, and Cook [1] for an excellent overview and a description of how computational work develops. More generally, branch-and-cut algorithms and its connected "facet hunting" emerged in the 1990s. This time has seen many implementations of branch-and-cut solvers for all kinds of combinatorial optimization problems.

With a somewhat more focus on practical usefulness and generality, MIP has developed in parallel (with many connections to LP and CO). Similar to the simplex algorithm, the introduction of branch-and-bound by Land and Doig [21] is of essential importance; see Cook [8] for a historical discussion. Further milestones were the papers of Crowder, Johnson, and Padberg [10] and van Roy and Wolsey [27] in the 1980s, which introduced generic cutting plane methods to MIP. Another milestone came with the demonstration of Balas, Ceria, Cornuéjols, and Natraj [3] that Gomory (mixed-integer) cuts could be made practically successful at the end of the 1990s; see also Cornuéjols [9] for a historical perspective. In a sense, this discovery (re)connected computational MIP-research back to the origins of the theoretical algorithm of Gomory [15].

Finally, computational work for MINLP has evolved quite naturally as an extension of MIP. Selected milestones include the approximation of separable expressions, e.g., McCormick envelopes [22]. For convex MINLP, the outer approximation algorithm of Duran and Grossmann [14] was a major step, which reduces the problem to a sequence of MIPs and nonlinear programs. Based on spatial branching, the branch-and-reduce algorithm of Tawarmalani and Sahinidis [26] was a major step towards practical implementations, improving the commercial solver Baron [24]. We refer to the overview articles of Belotti et al. [5] and Kronqvist et al. [20] for more information. Computational (nonconvex) MINLP and its software has seen a significant change in the last years, because many commercial solvers extended their capabilities to MINLP. Still, in this area there are several state-of-the-art open-source solvers around like Couenne [4] and SCIP [16].

Looking back at this history, one important fact is the strong relation to and dependence on commercial solver development. One important driving force was the need for stable LP-solvers in branch-and-cut algorithms. Thus, many or most research work in the decades between 1990 and 2010 used CPLEX as an LP-solver. Indeed, this is one of the historical origins of the CPLEX LP-solver [6]. However, significant implementation work was needed on top of CPLEX for branch-and-cut algorithms. The MIP-solving capability of CPLEX

[1] Thus, "computational research" in this article means that computations are performed on an actual computer, not the theoretical investigations of computational models.

(introduced in 1991) was also used as a comparison to show that a specialized solver is faster than a generic MIP-solver. In addition, the callback interface was used to introduce new MIP techniques.

Note that already at these times, CPLEX almost had the monopoly as LP- and MIP-solver. This is despite the fact that one needed to buy a license and that other alternatives existed. For instance, MIP-solving capabilities were introduced into the XpressMP solver in 1989. The academic MINTO solver [23] was developed starting at the beginning of the 1990s. In 1996, the academic LP-solver SoPlex [28] appeared and the development of the framework SCIP started in 2002. In the meantime, the COIN-OR initiative (started in 2000) made the LP-solver CLP and MIP-solver CBC available under an open-source license.

With the first release of Gurobi 2009, the focus on CPLEX has slowly shifted towards Gurobi, which is very often used as a framework. Nevertheless, several additional companies also developed their own MIP-solver, e.g., MOSEK, COPT, and Hexaly. Similarly, the LP- and MIP-solver HiGHS is available as open-source as well [19].

It is difficult to estimate the role of software licences over time. Today, all major MIP-solvers have an academic license and there are more and more open-source codes with admissible licenses. What definitely has not appeared is a common and community-driven open-source platform for MIP-solving.

Status-Quo: Challenges & Commercial Solvers

One challenge that Discrete Optimization faces, both theoretically and computationally, is that it has become more mature. For most research topics, it is not easy to improve the performance of previous methods and generic MIP- or MINLP-solvers. For example, an improvement of a few percent in running time is considered as a good progress for general MIP-solvers. Thus, different to a decade ago, to show an improvement today, one needs a larger test set and to run several random seeds. Therefore, more computational resources are needed.

This development also affects combinatorial optimization. While in the 1990s it was easy to be better than a general MIP-solver, this not the case anymore. Moreover, many specialized codes have been developed (not always publicly available) and it is hard to identify the best for a particular application.

Now, one could argue that then researchers should just move on and investigate other topics. In fact, this has partly been done by investigating MINLP more intensely. However, here the observations at the beginning of the article can be made in a similar way. Moreover, this move seems to be too early from a scientific viewpoint: There are many open issues and our understanding of the behavior of solvers is still limited; as two examples, consider cutting plane selection (Dey and Molinaro [13]) and smoothed analysis of the simplex algorithm (Bach et al. [2]). Moreover, the need for fast solutions in practice is still there.

One reason for the declining rate of improvement is that we predominantly focus on performance (which is undoubtedly important for solver-development). Already Hooker [18] criticized this. The goal should actually be to understand what is happening, at least in mathematical oriented research.

Moreover, today's MIP-solvers are relatively complex (as is the behavior of branch-and-cut in general). It is not easy to change/improve them. Indeed, some in the mathematical optimization community have claimed that solver development is an engineering task that should be left to others (outside their field or academia). However, while engineering is an important aspect, it requires a deeper understanding to improve the state-of-the-art. Such statements also support separation rather than inclusion.

All of the above might be one motivation to choose a black-box solver, i.e., as observed above usually a commercial solver. This lifts the user from the burden to actually understand what is going on in the solver. But, it has the severe drawback that performance now depends on a black-box. This is scientifically very problematic, because the original goal, namely to understand what the best methods are, is now tied to this solver. Especially if conclusions are drawn with respect to MIP-models, this is disputable, because a certain solver might not capture the structure of the particular models well. Moreover, if changes to the solver happen, the results might change as well – without a transparent reason.

One reason for choosing a single commercial solver might also be ignorance or laziness. If only one solver is mentioned in presentations or was used in the past, this is likely the one that is used in further work. It seems that many MIP-researchers have started with Gurobi (or CPLEX and then moved to Gurobi).

It is also important to discuss the role of commercial software in the area. On the one hand, today's performance is to some extent the result of methods (both theoretical and computational) developed in Discrete Optimization (but also software engineering). Thus, people in the area are somewhat proud of this development and it serves as a selling point to the general public or when applying for grants or positions. On the other hand, this success makes it hard to get further improvements. Actually, this development also is not in the interest of commercial software companies, since they would like to hire people that already have experience in the implementation of MIP or MINLP-solvers. Ironically, this will never be commercial solvers.

It also seems that the marketing of solver companies work well on many scientists, making them believe that the particular solver is best for every problem.

One important observation here is that often the results of research papers almost exclusively rely on a relative improvement, i.e., a given baseline is improved. Choosing this baseline is clearly a delicate business, but supposedly not always done extremely carefully. For instance, the right baseline is not necessarily the "best general solver". Of course, it often does not make sense to use an outdated baseline. However, the claim that a solver performs best on a certain set of benchmark instances does not mean that it will do so on others. Thus, taking the requirement of the best baseline seriously, the baseline would need to be chosen by comparing all available solvers. This is something that most articles do not take into account, because the results remain valid even if just a reasonable choice is taken. It seems that Gurobi has somehow been established as this reasonable choice. If a different solver is chosen in an article, one might even be forced to compare to Gurobi by reviewers. This choice has been a decision and is not mandated by scientific reasons alone.

With respect to the last observation at the beginning of this article, it might also be that today's students are not as interested in digging deeper into coding and high-performance programming languages like C or Julia. The lure of using AI (vibe coding) and the dominant role of Python in this area, might explain this effect partly.

One final issue is that the reputation that comes along with excellent theoretical work is often perceived to be higher than for excellent computational research. And sadly, excellence for the latter is often not easily accessible: Just ask yourself how you would define excellence in both domains …

Conclusions and Future Directions

One of the most important factors is *awareness*. Considering the arguments raised here (and further ones) is one important first step.

Moreover, here are some suggestions:

- Keep an eye on the overall goals of research. What do you want to achieve and what are the right tools for this? See also Hooker [17].
- Develop sound scientific methods to compare different optimization models, both theoretically and computationally.
- Try to support computational work, e.g., in university curricula and with Ph.D. students.
- Distinguish between performance measures for scientific research and practice. These are only more or less aligned.
- Be mindful when using a black-box solver as the basis of your work.
- Be reminded that the goals of commercial companies do not always align with the goals of science.
- As an editor or reviewer of journals try to uphold the quality level of computational work. The solution cannot be to lower the standards.
- Organize workshops/conferences that focus on computational Discrete Optimization.
- Highlight important open computational challenges. Similar to the solution of famous open mathematical questions or conjectures, this can make significant computational breakthroughs more visible.
- Possibly support the development of open-source software.

The role of open-source software should be discussed elsewhere, e.g., how to make them accessible to a community effort.

Acknowledgements I thank Christopher Hojny, Alexandra Lassota, and Joseph Paat for their helpful comments.

[1] David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The traveling salesman problem. A computational study.* Princeton University Press, Princeton, NJ, 2006.

[2] Eleon Bach, Alexander E. Black, Sophie Huiberts, and Sean Kafer. Beyond smoothed analysis: Analyzing the simplex method by the book. Technical Report 2510.21613, arXiv, 2025. URL https://arxiv.org/abs/2510.21613

[3] E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19(1):1–9, 1996. doi: https://doi.org/10.1016/0167-6377(96)00007-7.

[4] Pietro Belotti, Jon Lee, Leo Liberti, Francois Margot, and Andreas Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods Software*, 24(4-5):597–634, 2009. doi: 10.1080/10556780903087124.

[5] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013. doi: 10.1017/S0962492913000032.

[6] Robert E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, Extra Vol.:107–121, 2012. doi: 10.4171/dms/6/16.

[7] Kevin K. H. Cheung, Ambros Gleixner, and Daniel E. Steffy. Verifying integer programming results. In Friedrich Eisenbrand and Jochen Koenemann, editors, *Integer Programming and Combinatorial Optimization*, pages 148–160, Cham, 2017. Springer. doi: 10.1007/978-3-319-59250-3_13.

[8] William Cook. Markowitz and Manne + Eastman + Land and Doig = branch and bound. *Documenta Mathematica*, Extra Vol.:227–238, 2012. doi: 10.4171/dms/6/25.

[9] Gérard Cornuéjols. The ongoing story of Gomory cuts. *Documenta Mathematica*, Extra Vol.:221–226, 2012. doi: 10.4171/dms/6/24.

[10] Harlan Crowder, Ellis L. Johnson, and Manfred Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983. doi: 10.1287/opre.31.5.803.

[11] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2(4):393–410, 1954. doi: 10.1287/opre.2.4.393.

[12] George Dantzig. Programming in a linear structure. Technical report, U.S. Air Force Comptroller, USAF, Washington, D.C., 1948.

[13] Santanu .S. Dey and Marco Molinaro. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170:237–266, 2018. doi: 10.1007/s10107-018-1302-4.

[14] Marco A. Duran and Ignacio E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, 1986. doi: 10.1007/BF02592064.

[15] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958. doi: 10.1090/S0002-9904-1958-10224-4.

[16] Christopher Hojny, Mathieu Besançon, Ksenia Bestuzheva, Sander Borst, Antonia Chmiela, João Dionísio, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Adrian Göß, Alexander Hoen, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Stephen J. Maher, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Sebastian Pokutta, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Matthias Walter, Dieter Weninger, and Liding Xu. The SCIP optimization suite 10.0. Technical report, Optimization Online, 2025. URL https://optimization-online.org/2025/11/the-scip-optimization-suite-10-0/

[17] J. N. Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42(2):201–212, 1994. doi: 10.1287/opre.42.2.201.

[18] John N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1:33–42, 1995. doi: 10.1007/BF02430364.

[19] Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5.

[20] Jan Kronqvist, David E. Bernal Neira, and Ignacio E. Grossmann. 50 years of mixed-integer nonlinear and disjunctive programming. *European Journal of Operational Research*, 2025. doi: 10.1016/j.ejor.2025.07.016. To appear.

[21] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960. doi: 10.2307/1910129.

[22] Garth P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976. doi: 10.1007/BF01580665.

[23] George L. Nemhauser, Martin W. P. Savelsbergh, and Gabriele C. Sigismondi. MINTO, a Mixed INTeger Optimizer. *Operations Research Letters*, 15(1):47–58, 1994. doi: 10.1016/0167-6377(94)90013-2.

[24] Nikolaos V. Sahinidis. BARON: a general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996. doi: 10.1007/BF00138693.

[25] Peter Sanders. Algorithm engineering – an attempt at a definition. In Susanne Albers, Helmut Alt, and Stefan Näher, editors, *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 321–340. Springer, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-642-03456-5_22.

[26] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.

[27] Tony J. van Roy and Laurence A. Wolsey. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35:45–57, 1987. doi: 10.1287/opre.35.1.45.

[28] Roland Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. Dissertation, TU Berlin, 1996.

In 2022, the Mixed Integer Programming Society (MIPS) was established as a technical section of the Mathematical Optimization Society. In the MIP Insights *newsletter, we announce important news for the community, promote and summarize events supported by the society, and host expository presentations. Tami Carpenter, Silvia Di Gregorio, Fabio Furini, Christopher Hojny, Gonzlo Muñoz, Joseph Paat, Veronica Piccialli, Marc E. Pfetsch, and Sebastian Perez-Salazar contributed to this issue.*