Beyond Optimal Solutions for Real-World Problems

Maria Garcia de la Banda
Department of Data Science and AI, Faculty of IT, Monash University, Clayton, Victoria, Australia
ARC Industrial Training and Transformation Centre OPTIMA, Carlton, Victoria, Australia

Abstract

Combinatorial optimisation technology has come a long way. We now have mature high-level modelling languages in which to specify a model of the particular problem of interest [18, 7, 24, 6]; robust complete solvers in each major constraint paradigm, including Constraint Programming (CP) [1, 19], MaxSAT [5, 11], and Mixed Integer Programming (MIP) [2, 3]; effective incomplete search techniques that can easily be combined with complete solvers to speed up the search such as Large Neighbourhood Search [23]; and enough general knowledge about modelling techniques to understand the need for our models to incorporate components such as global constraints [25], symmetry constraints [8], and more. All this has significantly reduced the amount of knowledge required to apply this technology successfully to the many different combinatorial optimisation problems that permeate our society.

And yet, not many organisations use such advanced optimisation technology; instead, they often rely on the solutions provided by problem-specific algorithms that are implemented in traditional imperative languages and lack any of the above advances. Further, while advanced optimisation technology is particularly suitable for the kind of complex human-in-the-loop decision-making problems that occur in critical sectors of our society, including health, transport, energy, disaster management, environment and finance, these decisions are often still made by people with little or no technological support. In this extended abstract I argue that to change this state of affairs, our research focus needs to change from improving the technology on its own, to improving it so that users can better trust, use, and maintain the optimisation systems that we develop with it. The rest of this extended abstract discusses my personal experiences and opinion on these three points.

Trust

I highlight trust (which focuses on the user's point of view) rather than trustworthiness (which is a characteristic of the software itself) because I think it is the former rather than the latter that is at stake for the adoption of optimisation technology.

One of the biggest hurdles I have found for trust in the context of optimisation systems is for the domain experts to (feel like they) *understand the underlying model*. While many users will never do (or have to), I believe it is key for domain experts to have a high-level understanding of the constraints in the model, since their (dis)trust will likely spread through the organisation, impacting the adoption of the system. Thanks to the use of high-level modelling languages in CP, our group has achieved this [13] by documenting the constraints in a language the user knows (mathematics) and linking each constraint to the particular part of the model that implements it (via comments). While domain experts do not completely understand the model, the similarity between the format they understand (mathematics) and the model constraint has helped them verify our perception of their problem and improved their trust in the model. However, more needs to be done in this direction via the development of formal techniques. For example, our group is exploring the use of *domain-specific languages* [10] as a bridge between domain experts and modellers that helps both trust and maintenance (see later). This [27] and other approaches need to be explored.

A very significant source of trust for our domain experts (and of trustworthiness for the software) has been the development of two different models implemented by two different people for the same problem [13]. While this can be seen as a prohibitively expensive exercise, it did not take that long once the first model was mature, is a good way to onboard new optimisation team members, and has helped up detect not only bugs but also differences in the interpretation of domain expert information. For optimisation problems where it is not possible to verify the optimality (or even correctness) of the solution, we see such *redundant modelling* as the only solution for now. Interestingly, a significant

29th International Conference on Principles and Practice of Constraint Programming (CP 2023).
Editor: Roland H. C. Yap; Article No. 1; pp. 1:1–1:4

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

step forward in obtaining the trust of our domain experts has been the *generation of an optimality gap* whenever an optimal solution could not be found due to time constraints. While explaining this concept took time, once understood it has boosted their trust, particularly when tackling problems where the solution is not easy verifiable or when approximated models/data are used (needed for speed, see later). This makes it difficult to work with CP and SAT solvers, as they usually lack tight lower bounds. Finally, trust is often developed through the use of the system, which I discuss below.

Use

Usability is known to be key for the deployment of software systems. By "system" in our context, I refer to the combination of the problem model(s), the associated solver(s) and, importantly, the User Interface (UI) that often integrates them and is fundamental to their success. In addition to the traditional usability characteristics of software systems, I believe an optimisation system requires particular care in the following areas. *Interaction*, i.e., the system must allow users to interact with the UI not only to provide and modify the input data, but also to modify the constraints (at the very least by turning some on/off) as well as explore and compare solutions, as argued in [17, 15]. Incremental compilers and solvers would significantly help in making this easier, as well as generic ways for the UIs to communicate with them. *Conflict resolution*, that is, ensuring the system can not only detect infeasible instances, but also support users in understanding the data/constraints that cause infeasibility and how to modify the instance to make it feasible. Any interactive optimisation system that has users, will likely have conflicts. Thus, it is mandatory for CP to improve its conflict resolution technology which, while existent [16, 14, 22], is not widespread and it is often still problem-dependent, overwhelming (in the number of constraints shown to the user) and slow. Without it, users will be "stumped" when (rather than if) infeasibility is reached. *Solution diversity*, that is, supporting users in obtaining a diverse set of (close-to-optimal) solutions, where diversity is measured by a user-provided metric modelled somehow. While some solver-independent technology has been developed and implemented for this [9, 20, 12], it should be easier to use and more widespread. Further, it requires sophisticated solution comparison capabilities and, importantly, for optimal solutions to be found in seconds rather than hours. This brings me to *speed*, an area where CP solvers are falling behind. Most of our research group applications now use MIP solvers due to the need for floats (which precludes us from using learning solvers such as Chuffed [4]), but also to the lack of effective warm-start processes that are available in MIP solvers. Interestingly, data and model approximations have been proved to achieve orders of magnitude speedups with small reductions in optimality [13]. Developing generic (i.e., problem independent) accurate approximations would be extremely useful for complex decision systems. Other areas where I think generic CP methods are worth investigating more include dealing with *uncertainty* and *online* problems, ensuring solution *fairness* (even if it is over time), and studying *predict + optimise* approaches.

Maintain

I know very few papers devoted to the issue of maintenance in optimisation technology. While this may be due to my lack of knowledge, I suspect it is also due to the limited adoption of optimisation technology. While the issues in this area are again common to other software systems, I believe the solutions for CP require special attention. For example, the issue of changes in user requirements (that our research group calls *problem drift*) seems particularly prevalent in decision-making systems, as such problems can evolve rapidly due to unforeseen circumstances. This can make optimisation systems obsolete faster than expected. Our research group has proposed to tackle problem drift by developing a *requirements model* implemented in the above-mentioned MDSLs and created by both domain experts and modellers that, when modified re-generates parts of the model to support the modifications [27]. This and other approaches such as the creation of reusable models components [21, 26], or instantiatable classes for common problem domains, are worth investigating.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Theory of computation → Integer programming; Human-centered computing → Information visualization

Keywords and phrases Combinatorial optimisation systems, usability, trust, maintenance

Digital Object Identifier 10.4230/LIPIcs.CP.2023.1

Category Invited Talk

Acknowledgements Any good idea here is the result of working in many different projects with my colleagues at the Optimisation and at the Data Visualisation and Immersive Analytics research groups in the Faculty of IT, Monash University. I have learned a lot from them.

References

1 GECODE - An open, free, efficient constraint solving toolkit. URL: https://www.gecode.org/.
2 Gurobi Optimizer Reference Manual. URL: https://www.gurobi.com/documentation/9.5/refman/index.html.
3 IBM ILOG CPLEX optimizer. URL: https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer.
4 Geoffrey Chu. Improving combinatorial optimization - extended abstract. In Francesca Rossi, editor, *23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*, pages 3116–3120. IJCAI/AAAI, 2013. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6687.
5 Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In Jimmy Ho-Man Lee, editor, *17th International Conference on Principles and Practice of Constraint Programming - CP 2011*, Lecture Notes in Computer Science, pages 225–239. Springer, 2011. doi:10.1007/978-3-642-23786-7_19.
6 Robert Fourer, David M Gay, and Brian W Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.
7 Alan M. Frisch, Warwick Harvey, Chris Jefferson, Bernadette Martínez-Hernández, and Ian Miguel. Essence: A constraint language for specifying combinatorial problems. *Constraints*, 13(3):268–306, September 2008. doi:10.1007/s10601-008-9047-y.
8 Ian P. Gent, Karen E. Petrie, and Jean-François Puget. Symmetry in constraint programming. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 329–376. Elsevier, 2006. doi:10.1016/S1574-6526(06)80014-3.
9 Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In Manuela M. Veloso and Subbarao Kambhampati, editors, *20th National Conference on Artificial Intelligence, AAAI 2005*, pages 372–377. AAAI Press / The MIT Press, 2005. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-059.php.
10 Paul Hudak. Domain-specific languages. *Handbook of programming languages*, 3(39-60):21, 1997.
11 Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019. doi:10.3233/SAT190116.
12 Linnea Ingmar, Maria Garcia de la Banda, Peter J. Stuckey, and Guido Tack. Modelling diversity of solutions. In *34th Conference on Artificial Intelligence, AAAI 2020*, pages 1528–1535. AAAI Press, 2020. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5512.
13 Matthias Klapperstueck, Frits de Nijs, Ilankaikone Senthooran, Jack Lee-Kopij, Maria Garcia de la Banda, and Michael Wybrow. Exploring hydrogen supply/demand networks: Modeller and domain expert views. In Roland Yap, editor, *29th International Conference on Principles and Practice of Constraint Programming - CP23*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, to appear, 2023.

14 Niklas Lauffer and Ufuk Topcu. Human-understandable explanations of infeasibility for resource-constrained scheduling problems. In *ICAPS 2019 Workshop XAIP*, 2019.

15 Jie Liu, Tim Dwyer, Guido Tack, Samuel Gratzl, and Kim Marriott. Supporting the problem-solving loop: Designing highly interactive optimisation systems. *IEEE Trans. Vis. Comput. Graph.*, 27(2):1764–1774, 2021. doi:10.1109/TVCG.2020.3030364.

16 João Marques-Silva and Alessandro Previti. On computing preferred muses and mcses. In Carsten Sinz and Uwe Egly, editors, *17th International Conference on Theory and Applications of Satisfiability Testing - SAT 2014*, Lecture Notes in Computer Science, pages 58–74. Springer, 2014. doi:10.1007/978-3-319-09284-3_6.

17 David Meignan, Sigrid Knust, Jean-Marc Frayret, Gilles Pesant, and Nicolas Gaud. A review and taxonomy of interactive optimization methods in operations research. *ACM Trans. Interact. Intell. Syst.*, 5(3):17:1–17:43, 2015. doi:10.1145/2808234.

18 Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. MiniZinc: Towards a Standard CP Modelling Language. In Christian Bessière, editor, *13th International Conference on Principles and Practice of Constraint Programming – CP 2007*, Lecture Notes in Computer Science, pages 529–543. Springer, 2007. doi:10.1007/978-3-540-74970-7_38.

19 Laurent Perron and Vincent Furnon. Or-tools. URL: https://developers.google.com/optimization/.

20 Thierry Petit and Andrew C. Trapp. Finding diverse solutions of high quality to constraint optimization problems. In Qiang Yang and Michael J. Wooldridge, editors, *24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 260–267. AAAI Press, 2015. URL: http://ijcai.org/Abstract/15/043.

21 Sophia Saller and Jana Koehler. Easy, adaptable and high-quality modelling with domain-specific constraint patterns. *CoRR*, abs/2206.02479, 2022. doi:10.48550/arXiv.2206.02479.

22 Ilankaikone Senthooran, Matthias Klapperstueck, Gleb Belov, Tobias Czauderna, Kevin Leo, Mark Wallace, Michael Wybrow, and Maria Garcia de la Banda. Human-centred feasibility restoration in practice. *Constraints*, to appear, 2023.

23 Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael J. Maher and Jean-Francois Puget, editors, *4th International Conference on Principles and Practice of Constraint Programming - CP98*, Lecture Notes in Computer Science, pages 417–431. Springer, 1998. doi:10.1007/3-540-49481-2_30.

24 Pascal Van Hentenryck. *The OPL optimization programming language*. MIT Press, Cambridge, MA, USA, 1999.

25 Willem-Jan van Hoeve and Irit Katriel. Global constraints. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 169–208. Elsevier, 2006. doi:10.1016/S1574-6526(06)80010-6.

26 Toby Walsh. Constraint patterns. In Francesca Rossi, editor, *9th International Conference on Principles and Practice of Constraint Programming - CP 2003*, Lecture Notes in Computer Science, pages 53–64. Springer, 2003. doi:10.1007/978-3-540-45193-8_4.

27 Sameela Suharshani Wijesundara, Maria Garcia de la Banda, and Guido Tack. Addressing problem drift in UNHCR fund allocation. In Roland Yap, editor, *29th International Conference on Principles and Practice of Constraint Programming - CP23*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, to appear, 2023.