Black-Box Value Heuristics for Solving Optimization Problems with Constraint Programming

Augustin Delecluse
TRAIL, ICTEAM, UCLouvain, Belgium

Pierre Schaus
ICTEAM, UCLouvain, Belgium

Abstract

Significant research efforts have focused on black-box variable selection, with less attention given to value heuristics. An ideal value heuristic enables depth-first-search to prioritize high-quality solutions first. The Bound-Impact Value Selection achieves this goal through a look-ahead strategy, trying every value of the selected variable and ranking them based on their impact on the objective. However, this method is generally too computationally intensive for the entire search tree. We introduce two simple yet powerful modifications to improve its scalability. First, a lighter fix point computation involving only the constraints on the shortest path in the constraint graph between the variable and the objective. Second, a reverse look-ahead strategy optimistically fixes the objective variable to its minimum in order to prioritize the remaining values. These two ideas have been empirically validated on a range of academic problems and in the XCSP^3 competition, demonstrating significant improvements in scalability.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming

Keywords and phrases Constraint Programming, Value Selection, Look-Ahead, Optimization

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material *Software (Source Code)*: `https://github.com/augustindelecluse/choco-solver`

Funding *Augustin Delecluse*: This work was supported by Service Public de Wallonie Recherche under grant n°2010235 – ARIAC by DIGITALWALLONIA4.A

This is an extended abstract of a paper accepted at the main conference.

1 Introduction

The Constraint Programming (CP) paradigm enables to solve combinatorial problems in a declarative way. The Holy Grail vision of CP is that the user simply has to state the problem [4]. However, for practical efficiency, this vision has been adjusted to the long-standing CP mantra: *CP = Model + Search*. The capability to program problem-specific searches remains crucial in CP to minimize the search tree size. Over time, the emphasis on search programming has diminished due to the development of effective black-box search strategies by the CP community [5, 3, 8, 9, 11]. A common search procedure involves a two-step decision-making process at each node: selecting an undecided variable and then choosing a value to assign from the domain on the left branch, which is removed upon backtracking. Despite extensive work on deriving efficient problem-independent variable selection heuristics based on the first-fail principle, few black-box strategies have been proposed for value selection. In the context of optimization problems, a simple yet effective generic method is the Bound-Impact Value Selector (BIVS) [7]. This look-ahead heuristic iterates over each value in the domain of a variable, successively assigning the variable to each of these values. After each assignment is made, the fixpoint filtering by the constraints is computed. The

impact on the objective—specifically, the lower bound increase in the case of a minimization problem—is then measured. The value that yields the smallest increase for minimization problem is selected for the left assignment.[1]

Although BIVS finds solutions of better quality compared to other naive strategies, such as picking the minimum value in the domain of a variable, its computational cost does not always compensate its benefits in guiding the search toward promising directions. This is especially true for problems with large domain sizes or when a large number of constraints are involved in the fixpoint. To overcome this shortcoming, one tradeoff is to consider only the smallest and largest values from the domain, instead of every value, especially if the domain size is too large. This is the default value selection performed in Choco-solver [10].

2 Contributions and conclusion

Our work introduces two ideas to reduce the computational cost of BIVS. The first, called the *Restricted Fixpoint* (RF) is to compute the fixpoint on a restricted set of constraints. Specifically, we measure the impact only on a fixpoint that involves the constraints along the shortests paths from the selected variable to the objective variable in a constraint graph, where node constraints are connected to node variables in their scope. RF is only used within the value selection heuristic, in order to select an interesting value to use during branching; the actual search itself still considers all constraints. An example for a Traveling Salesman Problem (TSP) using a circuit model in CP is shown on Figure 1. Variables S_i , $i \in 0..n$ with $n = 4$ nodes are the successors in a TSP tour, D_i the distance between a node and its successor, and $TotD$ the total distance of the tour, which is the objective to minimize.

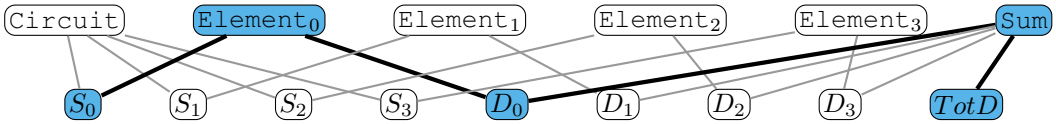

Figure 1 Constraints and variables on a TSP instance with 4 nodes. Variables and constraints in blue are located on the shortest path to the objective $TotD$ when considering variable S_0, and only the two highlighted constraints are propagated by the RF when examining suitable values for S_0.

The second improvement, called *Reverse Look-Ahead* (RLA) starts from the objective assignment. Instead of sequentially fixing the selected variable to each value in BIVS, we suggest optimistically setting the objective variable to its lower bound (assuming minimization) and then computing the fixpoint. The selected value is one of those that remain. If infeasibility is detected, we increase the objective bound and continue until no domain is empty.

The two proposed improvements are designed to be non-intrusive, meaning that they can be implemented without necessitating modifications to the solver's architecture. The main paper describe their implementation and behavior, along with guided examples. Experimental results on both academic problems and on 18 problems from the XCSP^3 2023 competition [6, 2] show that by incorporating a RF in the look-ahead process and employing RLA, we significantly reduced costs, making previous restrictions on BIVS usage less relevant. The methods we proposed do not require any training, are well suited for black-box settings, and substantially improve performance. When utilized alone or within a portfolio approach, these strategies continue to enhance the efficiency of solving constrained optimization problems.

[1] This idea is similar to *Strong Branching* in Mixed-Integer Linear Programming solvers [1].

References

1 David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. Finding cuts in the tsp (a preliminary report), 1995.

2 Gilles Audemard, Christophe Lecoutre, and Emmanuel Lonca. Proceedings of the 2023 xcsp3 competition. *arXiv preprint arXiv:2312.05877*, 2023.

3 Gilles Audemard, Christophe Lecoutre, and Charles Prud'Homme. Guiding backtrack search by tracking variables during constraint propagation. In *International Conference on Principles and Practice of Constraint Programming*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

4 Roman Barták. Constraint programming: In pursuit of the holy grail. In *Proceedings of the Week of Doctoral Students (WDS99)*, volume 4, pages 555–564. MatFyzPress Prague, 1999.

5 Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In *ECAI*, volume 16, page 146, 2004.

6 Frédéric Boussemart, Christophe Lecoutre, Gilles Audemard, and Cédric Piette. Xcsp3: an integrated format for benchmarking combinatorial constrained problems. *arXiv preprint arXiv:1611.03398*, 2016.

7 Jean-Guillaume Fages and Charles Prud'Homme. Making the first solution good! In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1073–1077. IEEE, 2017.

8 Steven Gay, Renaud Hartert, Christophe Lecoutre, and Pierre Schaus. Conflict ordering search for scheduling problems. In *Principles and Practice of Constraint Programming: 21st International Conference, CP 2015, Cork, Ireland, August 31–September 4, 2015, Proceedings 21*, pages 140–148. Springer, 2015.

9 Christophe Lecoutre, Lakhdar Saïs, Sébastien Tabary, and Vincent Vidal. Reasoning from last conflict (s) in constraint programming. *Artificial Intelligence*, 173(18):1592–1614, 2009.

10 Charles Prud'homme and Jean-Guillaume Fages. Choco-solver: A java library for constraint programming. *Journal of Open Source Software*, 7(78):4708, 2022. doi:10.21105/joss.04708.

11 Philippe Refalo. Impact-based search strategies for constraint programming. In *Principles and Practice of Constraint Programming–CP 2004: 10th International Conference, CP 2004, Toronto, Canada, September 27-October 1, 2004. Proceedings 10*, pages 557–571. Springer, 2004.