Extended Abstract for: Peel-and-Bound: Generating Stronger Relaxed Bounds with Multivalued Decision Diagrams

Isaac Rudich
Mathematics and Industrial Engineering Department, Polytechnique Montréal, Canada

Quentin Cappart
Computer Engineering and Software Engineering Department, Polytechnique Montréal, Canada

Louis-Martin Rousseau
Mathematics and Industrial Engineering Department, Polytechnique Montréal, Canada

1 Introduction

Multivalued decision diagrams (MDDs) are a graphical tool for compactly storing the solution space of discrete optimization problems. MDDs are particularly useful for generating strong dual bounds [1–4], especially on optimization problems where linear relaxations perform poorly. We further the work on decision diagram based branch-and-bound by introducing a method, referred to as *peel-and-bound*, of reusing the graphs generated at each iteration of the algorithm. In the full paper[1][2], (1) we present the *peel-and-bound* algorithm, (2) we identify and discuss heuristic decisions that can be used to adjust *peel-and-bound*, (3) we show that *peel-and-bound* outperforms *branch-and-bound* on the *sequence ordering problem* (SOP), and (4) we discuss how the algorithm can be applied to other problems.

2 Motivation

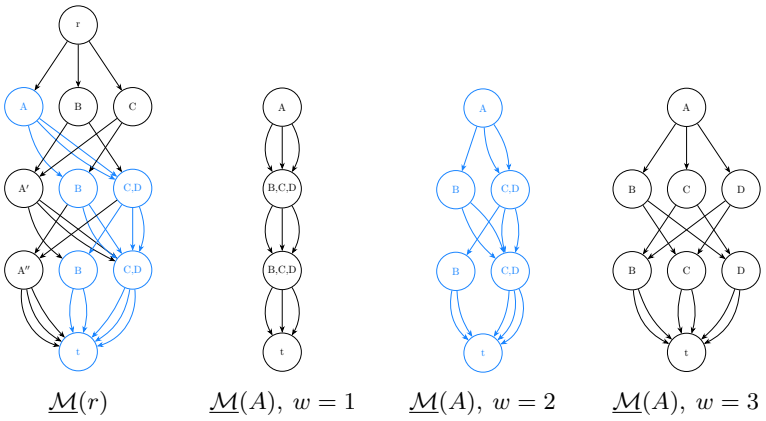
A relaxed decision diagram for a minimization problem $\mathcal{P}$ is a layered directed graph with a root r, and a terminal t, such that each path from r to t encodes a potential solution to $\mathcal{P}$. Furthermore, all feasible solutions to $\mathcal{P}$ must be encoded in paths from r to t, and infeasible solutions are allowed to be encoded in paths from r to t. Thus, the shortest path from r to t is a lower bound for $\mathcal{P}$. A maximum width w is chosen, and a relaxed decision diagram is constructed such that no layer of the diagram has more nodes than w. Diagrams with larger values of w contain fewer infeasible solutions, but take longer to generate. In a typical branch-and-bound algorithm, the branching takes place by splitting on the domain of the variables. In decision diagram based branch-and-bound, branching takes place on the nodes themselves by selecting a set of nodes $\mathcal{U}$, such that every path from r to t passes through at least one node in $\mathcal{U}$ [5]. The set $\mathcal{U}$ becomes a queue of nodes to be processed, and a relaxed decision diagram is generated for each node u in $\mathcal{U}$, with u as the root. This process can replace the linear relaxation typically used in branch-and-bound algorithms.

Observe that when a decision diagram $\mathcal{M}$ is split into a representative set $\mathcal{U}$, all of the solutions represented by the diagrams that result from processing the nodes in $\mathcal{U}$, are already represented as sub-graphs of $\mathcal{M}$. Peel-and-bound is a method of reading those sub-graphs to use a starting point, instead of generating a new decision diagram from scratch at each iteration. Complete details can be found in the full paper, but Figures 1 and 2 demonstrate the idea visually. Figure 3 shows our results from comparing decision diagram based branch-and-bound to peel-and-bound with several different values of w.

[1] Rudich is the main student author, Cappart and Rousseau are advisors.
[2] The full version of this paper was accepted at the main CP conference.

28th International Conference on Principles and Practice of Constraint Programming (CP 2022).
Editor: Christine Solnon; Article No. 15; pp. 15:1–15:3

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



$\underline{\mathcal{M}}(r)$ $\underline{\mathcal{M}}(A),\ w=1$ $\underline{\mathcal{M}}(A),\ w=2$ $\underline{\mathcal{M}}(A),\ w=3$

Figure 1 Example of an induced sub-graph for a SOP instance (shown in blue), and the associated relaxed decision diagram with the same root. The notation $\underline{\mathcal{M}}(u)$ refers to a relaxed decision diagram that has a root of u.

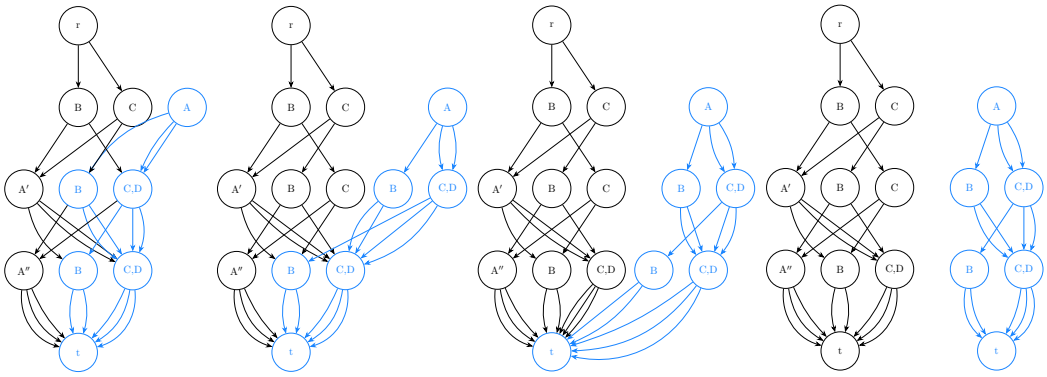


Figure 2 An example of a peel operation. In (1), A is selected to induce the peel process and removed from the the original diagram ($\underline{\mathcal{M}}(r)$ from Figure 1). In (2) the arcs that connect A to the original diagram are moved to copies of the nodes they originally ended at, and infeasible arcs are filtered. In (3) and (4) the process is repeated until the diagrams are disconnected.

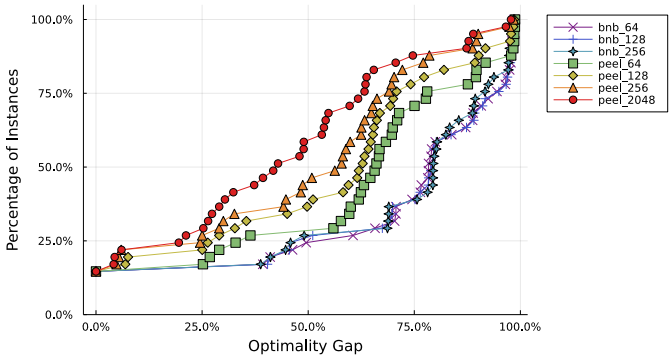


Figure 3 Performance Profiles: the optimality gap $= \frac{upper\_bound - lower\_bound}{upper\_bound}$

References

1 Quentin Cappart, Emmanuel Goutierre, David Bergman, and Louis-Martin Rousseau. Improving optimization bounds using machine learning: Decision diagrams meet deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1443–1451, 2019.

2 Margarita Castro, Chiara Piacentini, Andre Cire, and J. Beck. Solving delete free planning with relaxed decision diagram based heuristics. *Journal of Artificial Intelligence Research*, 67:607–651, 03 2020.

3 Willem-Jan Hoeve. Graph coloring with decision diagrams. *Mathematical Programming*, 05 2021.

4 Johannes Maschler and Günther Raidl. Multivalued decision diagrams for prize-collecting job sequencing with one common and multiple secondary resources. *Annals of Operations Research*, 302, 07 2021.

5 David Bergman, André A. Cire, Ashish Sabharwal, Horst Samulowitz, Vijay Saraswat, and Willem-Jan van Hoeve. Parallel combinatorial optimization with decision diagrams. *Proceedings of the International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 351–367, 2014.