Commentary on "A Systematic Approach to MDD-Based Constraint Programming"

Samid Hoda[1], Willem-Jan van Hoeve[2], and J. N. Hooker[2]

[1] Well Data Labs
samid.hoda@gmail.com
[2] Carnegie Mellon University
vanhoeve@andrew.cmu.edu, jh38@andrew.cmu.edu

Abstract. This retrospective on our CP 2010 paper provides a narrative of the research stream that contains it. It recounts how the research got started, how the paper itself came about, and briefly what has happened since.

1 Introduction

This is a retrospective on our paper "A systematic approach to MDD-based constraint programming," which appeared in the CP 2010 Proceedings [25]. The paper is intended to unify previous work on multivalued decision diagrams (MDDs) as an approach to constraint programming (CP). We briefly recount the story of how MDDs came to be applied to CP, how [25] pulled together some of the ideas, and what has happened since.

First, a bit of background. MDDs are a straightforward generalization of binary decision diagrams (BDDs), which were introduced by S. B. Akers in 1978 as a graphical representation of logic circuits [1]. BDDs and MDDs have since become a widely used tool in circuit design, product configuration, and other areas. Essential to this development was R. E. Bryant's 1986 result that any Boolean function has a unique minimal ("reduced") BDD representation for a given variable ordering [14].

By 2005, BDDs had seen a few applications to optimization and CP. They had been used to help solve certain optimization problems [4, 18, 32], or to reduce domains of set-valued variables in CP [24, 31]. Yet to our knowledge, no general BDD-based solution method had been proposed for optimization or CP.

2 How It Started

The stream of research that led to [25] began when Henrik Reif Andersen invited John Hooker to give a talk at the IT University of Copenhagen in 2005. There, John met Henrik's PhD student Tarik Hadžić. Henrik was using BDDs for product configuration, but Tarik and John saw them as a promising data

structure for discrete optimization and CP. The following year, they wrote a technical report [19] on postoptimality analysis with MDDs. The paper was never submitted for publication, and the idea remained dormant for over a decade (more on this later). At about the same time, however, John presented their MDD-based method for 0–1 programming at a global optimization conference [23], and they published similar ideas in the CPAIOR 2007 proceedings [20].

The basic attraction of MDDs for optimization and CP is a property they share with multilayer neural networks and dynamic programming (DP). The amount of information that can be stored in the diagram (or the DP state transition graph) increases exponentially with the depth of the diagram, even if the width is fixed. So-called deep learning in neural networks became possible through an increase in the number of layers, allowing the network to represent vastly more information. MDDs therefore offered the possibility of compactly representing a complicated feasible set of a CP or optimization problem.

An added attraction of MDDs is that once the MDD is constructed, an optimization problem with a separable objective function can be solved by finding a shortest or longest path in the MDD. This is a simple and fast computation. Furthermore, since a DP state transition graph can be viewed as an MDD (although not a reduced one in general), MDD-based methods are well suited for DP models. This could enable the solution of problems with recursive structure that lack a practical constraint-based formulation.

Returning to our story, Tarik and John quickly discovered that despite the representational power of MDDs, many combinatorial problems require an MDD of exponential width. John had arranged for Tarik to enroll as a visiting PhD student at Carnegie Mellon University, where they worked on the idea of using fixed-width MDDs as a *relaxation* of the problem. The relaxed MDD could be used in much the way as a linear programming relaxation in integer programming, or it could serve as an enhanced constraint store in CP that could be more effective than the traditional domain store.

During a brief return visit to Copenhagen, Tarik learned that Henrik and his student Peter Tiedemann were working on similar ideas. The four investigators combined forces to write a paper on relaxed MDDs as a constraint store, which appeared in CP 2007 [2]. They constructed relaxed MDDs both by node splitting (which refined the relaxation) and node merger (which reduced the size of the relaxed MDD). They found that propagation through MDDs instead of the domain store could drastically reduce the search tree and solution time for a system of all-different constraints, which is the CP equivalent of the famous graph coloring problem. This paper served as the basis for much subsequent work in the area, beginning with an extension to equality constraints [22], an application to configuration problems [21], and the paper discussed here [25].

3 Writing the Paper

In 2008, Samid (Sam) Hoda attended John's PhD course at CMU on Integrated Methods for Optimization, which turned into a book [27]. This was Sam's first

real exposure to discrete modeling using CP, as most of his prior experience had involved traditional optimization methods. He was extremely interested in the possibilities that an MDD store provided as an alternative data structure to encode relaxations. There were other unifying ideas introduced by John in that course around relaxation, duality, consistency and modeling that influenced how Sam viewed the role of MDDs in the broader context of solving CP and OR problems.

Sam, John, and Willem-Jan van Hoeve initially started working together on extending John and Tarik's line of research to AMONG and then hopefully SEQUENCE constraints. A key idea in MDD-based CP is that the MDD passes information from one constraint to the next; i.e., multiple constraints process the same MDD, instead of each constraint processing its individual MDD. In trying to develop an MDD propagator for AMONG constraints, we looked at John and Tarik's existing propagators for inspiration and for what they had in common. On the surface they seemed ad hoc and specialized for each constraint, similar to what one finds in traditional propagation schemes for global constraints.

Branching is a key idea in solving both CPs and MIPs in practice. In CP, if we view the traditional domain store as a relaxation, then branching tree nodes can be identified with relaxations, and edges with imposing additional local constraints. The same can be said for mixed integer programs with LP relaxations at each node and (simple) linear inequalities on edges. Similarly, MDD-based CP identifies MDDs with nodes and (simple) constraints with edges of the branching tree. This is a "macro" branching view of MDDs that is natural when viewing MDDs as relaxations.

However, there is another, more subtle level of branching that occurs when working with MDDs. One characterization of MDDs is that they themselves are branching trees in which isomorphic subtrees have been superimposed. Thus, it is fairly natural to associate traversing an edge in an MDD with adding (conjoining) a simple constraint. Yet what do we associate with nodes?

Sibling nodes in a branching tree can be viewed as disjuncts, and branching can be viewed as "splitting" the parent into its children. Armed with this point of view, we asked ourselves, what does it mean to merge nodes in an MDD? One loose inspiration, at least for Sam, came from the lift-and-project procedure in disjunctive programming. In disjunctive programming, each disjunct is a polyhedron, and the result of the projection is again a polyhedron that faithfully represents the union of the disjuncts. In MDDs, perhaps merging nodes would be similar to projecting their "content" back into the same "space," in which case their merged/projected information should faithfully represent the union of all the disjuncts. The fruitful idea in this (admittedly strained) analogy is that the "content" might be a consistent representation of underlying information, similar to the role of the LP relaxation in disjunctive programming.

By the time we had developed a propagator for AMONG constraints, we realized that the existing propagators were not as ad hoc as they first appeared. Each propagator involved associating a data structure (information) with each node of the MDD (similar to the role of a relaxation in the "macro" branching view).

Following an edge would be similar to adding a simple constraint. We introduced the $\otimes$ notation for operations that process information when traversing an edge to produce new information. When several edges entered a node, we introduced the $\oplus$ notation for the operation that combines the information obtained by traversing all the incoming edges. Processing information using $\otimes$, and combining information using $\oplus$, result in information of the same form (i.e., captured in a consistent data structure/representation) as that associated with the nodes from which the information is derived.

MDD propagation works by deleting infeasible edges, an operation that generalizes conventional domain filtering. Our general scheme localized the decision as to whether to delete an edge to the information stored at either end of the edge. Furthermore, the local information could be assembled during a single top-down pass and a single bottom-up pass through the MDD.

Soon afterwards, Willem had an inspired moment for the shaving argument early in the paper. He showed that establishing MDD consistency only adds a polynomial factor to the time and space complexity of determining feasibility of a single variable assignment with respect to the MDD and a given constraint. Another useful property of MDD propagation is that conventional domain propagators can be applied to determine inconsistency of arcs in the MDD. Interestingly, the resulting propagator may not establish MDD consistency, even if it achieves domain consistency.

In the meantime, John had been working with Ionuț Aron and Tallys Yunes on an integrated solver for optimization (SIMPL) that cleanly integrated CP and mixed integer programming by taking the best of both worlds [3, 40]. Inspired by this idea, Sam was excited to build an MDD-specific solver once it became clear that all the existing propagators could be modeled using a generic framework. Even though MDDs can grow exponentially large to represent a given constraint perfectly, the basis of MDD-based constraint programming is to control the size of a relaxed MDD by specifying a maximum width. When we started, we were worried that we didn't have a space-efficient way to propagate changes in the MDD and decided to write the code in C++ using space-efficient arrays. As we discovered later, some of these design decisions made it harder to extend the code to deal with propagators that didn't fit perfectly into the framework described in the paper.

While building the solver we lacked a full appreciation for how powerful restricted-width MDDs would be in practice. We quickly found out that we didn't need to propagate the constraints until a fixed point is reached (repeated top-down and bottom-up passes), and we defaulted to using a single top-down and bottom-pass for our experiments. We were also encouraged by the enormous reduction in backtracks (several orders of magnitude) even when using relaxed MDDs of very small width. But it wasn't only the decrease in the number of backtracks: because MDD propagation was so fast, we achieved orders-of-magnitude reductions in computation time as well.

By the time we had completed the experiments we hadn't managed to build a custom propagator for the SEQUENCE constraint as we had set out to do. But

we were eager to share our results with the CPAIOR community and decided to write the paper for CP 2010.

4 What Has Happened Since

Since 2010, research on MDD-based CP and optimization has moved in several directions, and we describe only a few highlights here. Comprehensive coverage of research through 2016 can be found in a book written by Willem, John, and their former PhD students David Bergman and André Ciré [10]. More recent results were presented at a Symposium on Decision Diagrams for Optimization at CMU in October 2018 (DDOPT 2018, https://sites.google.com/view/ddopt-2018). Still more research is reported on Willem's website, Decision Diagrams for Optimization [26]. David and André are coauthors of much of this research, and both were honored to receive Doctoral Dissertation Awards from the Association for Constraint Programming in recognition of their work on decision diagrams.

4.1 Toward a General-Purpose MDD-Based Solver

Experience with integer programming (IP) teaches that an effective solver requires both relaxation bounds and primal heuristics. Relaxed MDDs provide bounds, because the shortest path length in a relaxed MDD is a lower bound on the optimal value of a problem. It was found in [8, 12] that MDDs provide tighter bounds, in less time, for set covering and stable set problems than those obtained by the full cutting plane resources of a state-of-the-art IP solver.

A primal heuristic is an algorithm that finds good feasible solutions. A restricted MDD, the opposite of a relaxed MDD, provides a primal heuristic because its shortest paths correspond to good feasible solutions. MDD-based primal heuristics for set covering and set packing problems were found in [11] to be superior to primal heuristics in a commercial IP solver, and again to require less time.

The third ingredient in a general-purpose solver is branching. MDDs enable a novel branching scheme, because rather than branch on variables, one can branch on nodes in the last exact layer of a relaxed MDD. This allows the search process to take advantage of the information encoded in the relaxed MDD. This technique is combined in [9] with MDD-based relaxation and primal heuristics to design a general-purpose solver for discrete optimization.

4.2 Constraint Reasoning with Decision Diagrams

As a direct application of the ideas discussed in the paper [25], observe that MDDs can be incorporated in a general-purpose CP solver by viewing the relaxed MDD as a global constraint. Propagation in the MDD is stronger than traditional propagation on variable domains because a relaxed MDD transmits more information from one constraint to another. For example, the paper [6] studies MDD-based propagation for the SEQUENCE constraint, and demonstrates

that relaxed MDDs of modest width can speed up the solving process several orders of magnitude in some cases.

In [16], MDD-based constraint propagation was developed for general sequencing and routing problems, which may include time windows, setup times, and precedence constraints. This paper also incorporates cost-based propagation based on the objective value. The MDDs realized substantial solution acceleration which resulted in closing several open instances from the sequential ordering problem benchmark set on TSPlib. This work has inspired several other papers to apply decision diagrams to scheduling and routing, including a solution method to solve pickup-and-delivery problems for Grubhub in real time [35].

4.3 Recursive Modeling

MDD-based solvers naturally accept recursive models and therefore provide an alternative method for solving deterministic DP problems. By relying on MDD-based branch and bound rather than state space enumeration, they suggest a new strategy for attacking the "curse of dimensionality" in DP.

The connection with DP is explored further in [28], which shows how one can simplify a DP model by viewing the DP state transition graph as an MDD and reducing it. Arc costs complicate the reduction process, but [28] generalizes Bryant's uniqueness result to MDDs with costs. Specifically, if the costs are distributed in a "canonical" fashion on the arcs, there is a unique reduced MDD representing a given optimization problem. This also allows one to solve problems with nonseparable cost functions. The paper reduces the MDD for a classical inventory management model to a width of one, making the problem trivial to solve. It also shows how MDDs can be generalized to *nonserial* MDDs to represent nonserial DP problems.

Part of the MDD modeling process is designing a scheme for merging or splitting DP states so as to create a valid relaxed MDD. Our paper [25] does not provide a practical way to identify valid node merger schemes, but [29] partially meets this need. It states sufficient conditions on the merger operation that are easily checked and can form the basis for solving deterministic DP models in general.

4.4 Combination with Other Techniques

MDDs can be profitably combined with Lagrangean relaxation and Benders decomposition. Lagrangean relaxation is combined with a relaxed MDD in [7] to obtain tighter bounds within an existing CP solver than are available from the solver alone. This resulted in substantial speedups in the solution of the TSP problem with time windows. General sufficient conditions for when Lagrangian relaxation can be implemented on an MDD are stated in [30]. The paper used a stand-alone MDD (without a CP solver) to obtain very tight bounds for a well-known set of job sequencing instances that had never been solved to optimality, and closed some of the instances.

The master problem in Benders decomposition can take the form of an MDD. Benders cuts can be added to the master problem by solving a separation problem for the MDD that is analogous to the famous separation problem in IP [15]. The Benders subproblem can also be formulated as an MDD, and Benders cuts obtained by manipulating the arc costs in the MDD [34, 33].

Decision diagrams can also be integrated in modern integer programming solvers, which rely on cut generation and tight bounds from linear programming to reduce the branch-and-bound search process. In [39], decision diagrams are used to generate cutting planes that separate a fractional solution from the convex hull of integer solutions to an integer program. While the decision diagram can be compiled from any discrete sub-problem of the integer program, the authors illustrate their method on independent set problems. This approach was extended in [17] in which cutting planes are generated for general integer nonlinear programming problems.

Relaxed decision diagrams can further be embedded inside the branch-and-bound search for integer programming, to improve the bounds [37, 38]. In this case, the decision diagrams are compiled generically, using the conflict graph as primary structure. The conflict graph is a common component in modern MIP solvers and represents the pairs of binary variables that cannot both take a certain pair of values. To strengthen the bound, the authors apply constraint propagation and Lagrangrean relaxation based on the constraints that are not represented exactly in the decision diagram.

4.5 Nonlinear Problems

Decision diagrams have been applied to represent and solve nonlinear optimization problems as well. The paper [5] considers integer programming problems with a nonlinear objective function. They propose to represent the objective function using decision diagrams which are encoded into binary form and added to the integer program.

The paper [13] applies decision diagrams and Benders decomposition to solve quadratically constrained integer programs.

4.6 Postoptimality Analysis

MDDs provide an ideal framework for postoptimality analysis in CP and discrete optimization. They can compactly store a huge collection of optimal and near-optimal solutions. They can then be rapidly queried to learn the structure of these solutions. For example, one can ask which solutions within a given tolerance of the optimal value satisfy stated properties, such as having certain variables fixed to a desired value. Postoptimality analysis is vitally important in practice, but it has been underutilized, perhaps because no satisfactory method has been available.

A recent paper [36] builds on ideas in the early technical report cited above [19] to conduct MDD-based postoptimality analysis for IP. It shows that a reduced MDD containing near-optimal solutions can be further reduced to a

"sound" MDD, a concept introduced in [19], that is equally useful for postoptimality analysis. It proves that a certain node merger operation, applied repeatedly, yields a smallest possible sound MDD for a given problem. It remains a research challenge to adapt MDD-based postoptimality analysis to problems with continuous as well as discrete variables.

References

1. Akers, S.B.: Binary decision diagrams. IEEE Transactions on Computers C-27, 509–516 (1978)
2. Andersen, H.R., Hadžić, T., Hooker, J.N., Tiedemann, P.: A constraint store based on multivalued decision diagrams. In: Bessière, C. (ed.) Principles and Practice of Constraint Programming (CP 2007). LNCS, vol. 4741, pp. 118–132. Springer (2007)
3. Aron, I., Hooker, J.N., Yunes, T.H.: SIMPL: A system for integrating optimization techniques. In: Régin, J.C., Rueher, M. (eds.) CPAIOR Proceedings. LNCS, vol. 3011, pp. 21–36. Springer (2004)
4. Behle, M.: Binary Decision Diagrams and Integer Programming. Ph.D. thesis, Max Planck Institute for Computer Science (2007)
5. Bergman, D., Cire, A.A.: Discrete nonlinear optimization by state-space decompositions. Management Science 64(10), 4700–4720 (2017)
6. Bergman, D., Cire, A.A., van Hoeve, W.J.: MDD Propagation for Sequence Constraints. JAIR 50, 697–722 (2014)
7. Bergman, D., Ciré, A.A., van Hoeve, W.J.: Lagrangian bounds from decision diagrams. Constraints 20, 346–361 (2015)
8. Bergman, D., Ciré, A.A., van Hoeve, W.J., Hooker, J.N.: Optimization bounds from binary decision diagrams. INFORMS Journal on Computing 26, 253–268 (2013)
9. Bergman, D., Ciré, A.A., van Hoeve, W.J., Hooker, J.N.: Discrete optimization with binary decision diagrams. INFORMS Journal on Computing 28, 47–66 (2014)
10. Bergman, D., Ciré, A.A., van Hoeve, W.J., Hooker, J.N.: Decision Diagrams for Optimization. Springer (2016)
11. Bergman, D., Ciré, A.A., van Hoeve, W.J., Yunes, T.: BDD-based heuristics for binary optimization. Journal of Heuristics 20, 211–234 (2014)
12. Bergman, D., van Hoeve, W.J., Hooker, J.N.: Manipulating MDD relaxations for combinatorial optimization. In: Proceedings of CPAIOR. LNCS, vol. 6697, pp. 20–35 (2011)
13. Bergman, D., Lozano, L.: Decision Diagram Decomposition for Quadratically Constrained Binary Optimization. Under Review
14. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Transactions on Computers C-35, 677–691 (1986)
15. Ciré, A., Hooker, J.N.: The separation problem for binary decision diagrams. In: ISAIM Proceedings (2014)
16. Ciré, A.A., van Hoeve, W.J.: Multivalued decision diagrams for sequencing problems. Operations Research 61, 1411–1428 (2013)
17. Davarnia, D., van Hoeve, W.J.: Outer Approximation for Integer Nonlinear Programs via Decision Diagrams. Under Review
18. Hachtel, G.D., Somenzi, F.: A symbolic algorithm for maximum flow in 0–1 networks. Formal Methods in System Design 10, 207–219 (1997)

19. Hadžić, T., Hooker, J.N.: Postoptimality analysis for integer programming using binary decision diagrams. Tech. rep., Carnegie Mellon University (2006)
20. Hadžić, T., Hooker, J.N.: Cost-bounded binary decision diagrams for 0-1 programming. In: Loute, E., Wolsey, L. (eds.) CPAIOR 2007 Proceedings. LNCS, vol. 4510, pp. 84–98. Springer (2007)
21. Hadžić, T., Hooker, J.N., O'Sullivan, B., Tiedemann, P.: Approximate compilation of constraints into multivalued decision diagrams. In: Stuckey, P.J. (ed.) Principles and Practice of Constraint Programming (CP 2008). LNCS, vol. 5202, pp. 448–462. Springer (2008)
22. Hadžić, T., Hooker, J.N., Tiedemann, P.: Propagating separable equalities in an MDD store. In: Perron, L., Trick, M.A. (eds.) CPAIOR 2008 Proceedings. Lecture Notes in Computer Science, vol. 5015, pp. 318–322. Springer (2008)
23. Hadzic, T., Hooker, J.N.: Discrete global optimization with binary decision diagrams. In: GICOLAG 2006. Vienna, Austria (December 2006)
24. Hawkins, P., Lagoon, V., Stuckey, P.: Solving Set Constraint Satisfaction Problems Using ROBDDs. JAIR 24(1), 109–156 (2005)
25. Hoda, S., van Hoeve, W.J., Hooker, J.N.: A systematic approach to MDD-based constraint programming. In: Cohen, D. (ed.) Principles and Practices of Constraint Programming (CP 2010). LNCS, vol. 6308, pp. 266–280. Springer (2010)
26. van Hoeve, W.J.: Decision Diagrams for Optimization website (2019), http://www.andrew.cmu.edu/user/vanhoeve/mdd/
27. Hooker, J.N.: Integrated Methods for Optimization, 2nd ed. Springer (2012)
28. Hooker, J.N.: Decision diagrams and dynamic programming. In: Gomes, C., Sellmann, M. (eds.) CPAIOR 2013 Proceedings. LNCS, vol. 7874, pp. 94–110. Springer (2013)
29. Hooker, J.N.: Job sequencing bounds from decision diagrams. In: Beck, J.C. (ed.) Principles and Practice of Constraint Programming (CP 2017). LNCS, vol. 10416, pp. 565–578. Springer (2017)
30. Hooker, J.N.: Improved job sequencing bounds from decision diagrams. In: de Givry, S., Schiex, T. (eds.) Principles and Practice of Constraint Programming (CP 2019). Springer (2019)
31. Lagoon, V., Stuckey, P.J.: Set domain propagation using ROBDDs. In: Wallace, M. (ed.) Principles and Practice of Constraint Programming (CP 2004). LNCS, vol. 3258, pp. 347–361 (2004)
32. Lai, Y.T., Pedram, M., Vrudhula, S.: EVBDD-based algorithms for integer linear programming, spectral transformation, and function decomposition. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 13, 959–975 (1994)
33. Lozano, L.: A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs. In: Symposium on Decision Diagrams for Optimization. Carnegie Mellon University (2018)
34. Lozano, L., Smith, J.C.: A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs. Mathematical Programming (2018)
35. O'Neil, R.J., Hoffman, K.: Decision diagrams for solving traveling salesman problems with pickup and delivery in real time. Operations Research Letters 47(3), 197–201 (2019)
36. Serra, T., Hooker, J.N.: Compact representation of near-optimal integer programming solutions. Mathematical Programming (published online May 2019)
37. Tjandraatmadja, C.: Decision Diagram Relaxations for Integer Programming. Ph.D. thesis, Carnegie Mellon University (2018)

38. Tjandraatmadja, C., van Hoeve, W.J.: Incorporating Bounds from Decision Diagrams into Integer Programming. Under Review
39. Tjandraatmadja, C., van Hoeve, W.J.: Target Cuts from Relaxed Decision Diagrams. INFORMS Journal on Computing 31(2), 285–301 (2019)
40. Yunes, T.H., Aron, I., Hooker, J.N.: An integrated solver for optimization problems. Operations Research 58, 342–356 (2010)