The Separation Problem for Binary Decision Diagrams

André A. Ciré and **J. N. Hooker**
Carnegie Mellon University, Pittsburgh, USA
jh38@andrew.cmu.edu, andrecire@cmu.edu

Abstract

The separation problem is central to mathematical programming. It asks how a continuous relaxation of an optimization problem can be strengthened by adding constraints that separate or cut off an infeasible solution. We study an analogous separation problem for a discrete relaxation based on binary decision diagrams (BDDs), which have recently proved useful in optimization and constraint programming. The algorithm modifies a relaxed BDD so as to exclude a single solution or family of solutions specified by a partial assignment. A key issue is the growth of the separating BDD when a sequence of solutions are cut off. We prove lower and upper bounds on the growth rate. We also examine growth empirically in a logic-based Benders method for a home health care scheduling problem. We find that the separating BDD tends to grow only linearly with the number of Benders cuts.

Introduction

The *separation problem* is fundamental for mathematical programming methods. It is generally understood as the problem of finding a constraint that "separates" or "cuts off" a given infeasible solution. The separation problem normally arises when a relaxation of the problem is solved to obtain a bound on the optimal value of the problem. When the solution of the relaxation is infeasible in the original problem, the relaxation is augmented with one or more constraints that cut off the solution. The tighter relaxation that results can then be solved to obtain a different solution, perhaps one that is feasible in the original problem or provides a better bound.

For example, integer programming (IP) solvers typically solve a continuous relaxation of the problem. When the solution is infeasible, *separating cuts* in the form of linear inequality constraints may be generated and added to the relaxation. The cuts may be general Gomory cuts or mixed-integer rounding cuts, or they may be special-purpose cuts that exploit the problem structure (see (Marchand et al. 2002) for a survey).

Given this, one may ask whether separation can be useful for other kinds of relaxations in an optimization context. One possible relaxation is a discrete relaxation based on *binary decision diagrams* (BDDs), which have recently been applied to optimization and constraint programming (CP). BDDs have long been used for circuit design, configuration, and related purposes (Akers 1978; Lee 1959; Bryant 1986; Hu 1995), but *relaxed* BDDs can provide an enhancement to the traditional CP domain store, bounds for branch-and-bound methods, and a master problem formulation for Benders decomposition (Andersen et al. 2007; Hadžić et al. 2008; Hoda, van Hoeve, and Hooker 2010; Hadžić and Hooker 2006; 2007; Becker et al. 2005; Behle and Eisenbrand 2007).

In this paper, we study the separation problem for BDDs. We present a general separation algorithm and investigate the complexity of separation, both analytically and empirically. The algorithm and analysis can be easily extended to *multivalued* decision diagrams (MDDs).

Separating BDDs

For our purposes, a BDD is a directed acyclic graph in which the nodes are partioned into *layers*, with the *root* node r in layer 1 and the *terminal* node t in layer $n+1$. Layers $1,\dots,n$ correspond to binary variables $x_1,\dots,x_n$, and each arc leaving a node in layer i is labeled 0 or 1 to indicate a value of x_i. Each path from the root to the terminal node therefore corresponds to a possible assignment to $x=(x_1,\dots,x_n)$. Examples of BDDs appear in Fig. 1, where dashed arcs are 0-arcs and solid arcs are 1-arcs.

A BDD is *reduced* when it is the smallest BDD that represents a given set of assignments to x. It can be shown that there is a unique reduced BDD for a given variable ordering (Bryant 1986).

Any 0-1 optimization problem with variables x can be represented by a BDD whose r-t paths correspond to feasible solutions of the problem. A separable objective function $\sum_i f_i(x_i)$ can be represented by assigning length $f_i(0)$ to 0-arcs leaving layer i and $f_i(1)$ to 1-arcs. An optimal solution of the problem corresponds to a shortest (or longest) path in the BDD. Nonseparable objective functions can be represented by assigning canonical costs as described in (Hooker 2013).

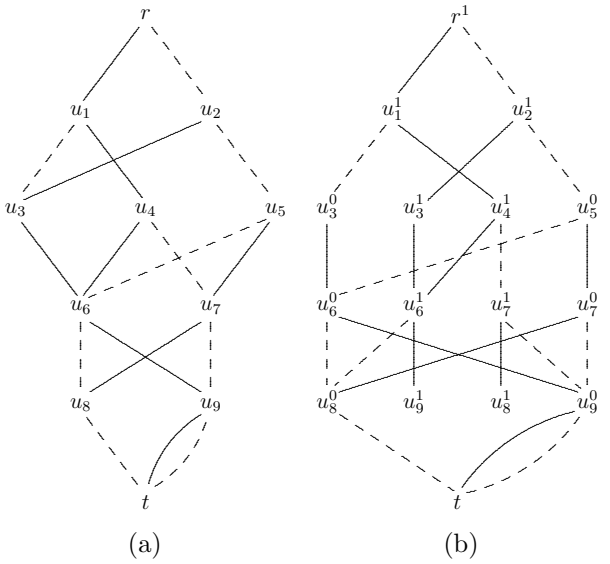

Figure 1: (a) A BDD with layers corresonding to 0-1 variables $x_1, \ldots, x_5$. (b) BDD after separation of partial assignment $(x_2, x_4) = (1, 1)$ from (a).

Because the BDD can grow exponentially with the problem size, it is useful to consider a smaller, *relaxed* BDD that represents a superset of the feasible solutions (Andersen et al. 2007; Hoda, van Hoeve, and Hooker 2010). A relaxed BDD is normally built so as to limit its *width*, or the maximum number of nodes in a layer. This kind of relaxation can serve two purposes. First, constraint propagation in CP can be strengthened by replacing the traditional domain store with a relaxed BDD, in which edge deletion replaces domain filtering. Second, the length of a shortest r-t path in a relaxed BDD provides a lower bound on the optimal value.

When a solution obtained from a relaxed BDD is infeasible, the relaxation can be tightened by removing the corresponding path, much as one cuts off a solution of a continuous relaxation. The separation problem therefore becomes one of efficiently finding a modified BDD that excludes a specified solution. We will refer to such a diagram as a *separating BDD*.

Growth of Separating BDDs

A key issue for separating BDDs is how fast they grow as solutions are separated. In IP, a solution can be separated with a single inequality, so that the continuous relaxation grows only linearly with the number of solutions separated. We will show, however, that a separating BDD can grow exponentially with the number of solutions separated, even if the BDD is reduced.

We therefore investigate the growth of separating BDDs when they are used in an optimization context. One potential application of BDD-based separation is to logic-based Benders decomposition, which has yielded substantial speedups over IP and CP solvers in a number of applications (Hooker 1995; Hooker and Yan 1995; Hooker 2000; Hooker and Ottosson 2003; Jain and Grossmann 2001; Harjunkoski and Grossmann 2002; 2001; Cambazard et al. 2004; Chu and Xia 2004; Maravelias and Grossmann 2004b; 2004a; Terekhov, Beck, and Brown 2005; Benini et al. 2005). In a Benders method, the solution of a *master problem* assigns values to some of the variables. This defines a subproblem that is solved to determine values of the remaining variables. If the subproblem is infeasible, a *Benders cut* is added to the master problem so as to exclude value assignments that create infeasibility. Typically, the Benders cut excludes a *partial assignment*; that is, it excludes all solutions in which certain variables are fixed to certain values.

The Benders master problem typically consists of an IP or CP model. However, if the master problem contains a relaxed BDD, one can reap the potential advantages of BDD-based optimization—provided Benders cuts are incorporated into the BDD. We therefore study a generalized separation problem for BDDs in which the goal is to cut off a given partial assignment. This is illustrated by Fig. 1(b), which shows a BDD that separates the partial assignment $(x_2, x_4) = (1, 1)$ from the BDD in Fig. 1(a).

In principle, a partial assignment can be separated from a BDD simply by conjoining the BDD with a second BDD (of width 2) that excludes precisely the solutions represented by the partial assignment. BDDs can be conjoined using a standard composition algorithm (Andersen 1997).

However, we present an algorithm that is designed specifically for separation and operates directly on the given BDD. We do so for two reasons. (i) There is no need for an additional data structure to represent the second BDD. (ii) The algorithm contains only logic that is essential to separation, which allows us to obtain a sharper bound on the size of the separating BDD.

In the worst case, the separation algorithm produces BDDs that grow exponentially with the number of partial solutions excluded. But this does not imply exponential growth if the BDD is reduced after each cut is added. We will see that, in some cases, the BDD created by the algorithm can in fact be simplified even if the original BDD is reduced. Yet we will show that, despite this, even reduced separating BDDs can grow exponentially.

One the other hand, we will prove an upper bound on growth that exploits properties of the partial assignments. For example, growth is linear if complete solutions are cut off. Furthermore, only the top part of the BDD grows if the partial assignments involve only variables in this part of the BDD.

Because the separating BDD can grow exponentially in principle, it is important to observe whether it grows rapidly when presented with a realistic sequence of partial assignments to be cut off. We observe the growth rate in the context of a logic-based Benders algorithm for home health care scheduling (Cheng and Rich 1998; Steeg and Schröder 2007; Bertels and Fahle 2006;

Rendl et al. 2012; Nielsen 2006; Ciré and Hooker 2012) and find that it is close to linear in nearly all instances.

Notation

We let L_i be the set of nodes in layer i of a BDD, where $r \in L_1$, and $t \in L_{n+1}$. Every arc in the BDD runs from layer i to layer $i+1$ for some $i \in \{1,\dots,n\}$. Each arc leaving a node $u \in L_i$ is labeled with a value $v \in \{0,1\}$, which corresponds to setting $x_i = v$. At most one arc with label v leaves u, and we denote this arc by $(u, v, u(v))$, where $u(v)$ is the node at the other end of the arc. Each path from r to t corresponds to an assignment $x = \bar{x}$, where $x = (x_1,\dots,x_n)$, and the arc leaving layer i on the path has label $\bar{x}_i$. For notational purposes we identify a path with the tuple of arcs on the path.

Let $x_i = \bar{x}_i$ for $i \in I$ be a partial assignment to x. The *separation problem* for this partial assignment and a given BDD B is to find a BDD B' that excludes this partial assignment from B. That is, the r-t paths in B' consist of those in B except those corresponding to assignments x with $x_i = \bar{x}_i$ for $i \in I$.

Separating a Partial Assignment

Given BDD B and partial assignment $x_i = \bar{x}_i$ for $i \in I$, we wish to construct a separating BDD B'. Let L'_i be the set of nodes on layer i of B'. In the course of the separation algorithm, we will associate a state $\delta \in \{0,1\}$ with each node of B'. A node with state δ is denoted u^δ when it is generated from node u in B. A node $u^\delta \in L_i$ has state $\delta = 1$ when on every path into u^δ, $x_j = \bar{x}_j$ for all $j \in I \cap \{1,\dots,i-1\}$. Otherwise it has state $\delta = 0$.

The algorithm builds B' one layer at a time beginning at the top. Layer 1 contains only the node r^1. To build layer $i+1$, it looks at each node u^δ in layer i. If $\delta = 0$, then it simply copies the arcs leaving u in B, and assigns state 0 to the nodes at the other end of the arcs. If $\delta = 1$, it looks at each of the arcs $(u, v, u(v))$ leaving u in B. If $i \notin I$, it adds to B' a corresponding arc $(u^1, v, u(v)^1)$ leading to a node with state 1. (This adds $u(v)^1$ to L'_{i+1} if it is not already present.) If $i \in I$, it does the following. If there is an arc $(u, \bar{x}_i, u(\bar{x}_i))$ with label $\bar{x}_i$ from u in B, a corresponding arc $(u^1, \bar{x}_i, u(\bar{x}_i)^1)$ to a node with state 1 is added to B'. For all other arcs $(u, v, u(v))$ leaving u in B, a corresponding arc $(u^1, v, u(v)^0)$ to a node with state 0 is added to B'. The state switches to 0 because the assignment differs from $\bar{x}_i$.

The procedure is slightly different in layer n for nodes u^1 with state 1, because paths consistent with the partial assignment must be cut off. If $n \notin I$, no arcs from u^1 to t are added. If $n \in I$, no arc with label $\bar{x}_n$ is added, but arcs with other labels are added. The algorithm concludes by removing all nodes from which there is no path to t. The precise algorithm appears in Fig. 2, where $V(u)$ is the set of labels on arcs leaving node u in B.

```
L'_1 ← r^1
For i = 1,...,n:
    L'_{i+1} ← ∅
    For all u^δ ∈ L'_i:
        If δ = 0 then
            For all v ∈ V(u):
                Add arc (u^0, v, u(v)^0) to B'
        Else
            If i ∉ I and i < n then
                For all v ∈ V(u):
                    Add arc (u^1, v, u(v)^1) to B'
            If i ∈ I then
                If x̄_i ∈ V(u) and i < n then
                    Add arc (u^1, v, u(v)^1) to B'
                For all v ∈ V(u) \ {x̄_i}:
                    Add arc (u^1, v, u(v)^0) to B'
Remove all nodes of B' from which there is no
    path to t.
```

Figure 2: Separation algorithm for a partial assignment $\{\bar{x}_i \mid i \in I\}$.

As an example, suppose the separation algorithm is applied to the BDD of Fig. 1(a) to cut off the partial assignment $(x_2, x_4) = (1, 1)$. The resulting BDD is shown in Fig. 1(b). Because nodes u_8^1 and u_9^1 do not lie on an r-t path, they and the two arcs adjacent to them may be dropped from the diagram.

The following is proved by induction on the layers. This and subsequent proofs are omitted due to space constraints.

Theorem 1 *The algorithm of Fig. 2 generates a separating BDD for a given BDD and partial assignment $x_i = \bar{x}_i$ for $i \in I$.*

Size of the Separating BDD

An important issue is how large a separating BDD can grow, particularly when numerous partial solutions are cut off sequentially, a common situation in practice. We first derive upper bounds on the size of the BDD that results from separating one partial solution. We then show that even a reduced separating BDD can grow exponentially when a sequence of partial solutions is removed. However, the growth may be much slower in practice, as illustrated by the computational results.

We first observe that it may be possible to reduce a separating BDD even when the original BDD is irreducible. Consider, for example, the BDD with two binary variables x_1, x_2 shown in Fig. 3(a). Using the algorithm to separate the partial assignment $x_2 = 1$ results in the BDD of Fig. 3(b), which can be reduced to the BDD of Fig. 3(c).

Suppose the separation algorithm is applied to BDD B to obtain BDD B'. A simple bound on the size of B' results from observing that each layer of B' contains at most two copies of the nodes in the corresponding layer of B. When a single solution is separated from B,

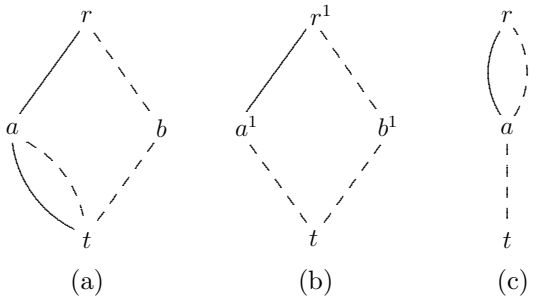

Figure 3: (a) Initial BDD, (b) BDD after separation of partial assignment $x_2 = 1$, and (c) reduced version of (b).

at most node per layer can have state 1. We therefore have the following.

Theorem 2 *The algorithm of Fig. 2 at most doubles number of nodes in the given BDD. When separating a single solution, it adds at most 1 node to each layer.*

We also have a bound on the complexity of the separation algorithm. Creating layer $i+1$ of B' requires examining at most $2|L_i|$ nodes in layer i of B', where L_i is the set of nodes in layer i of B. For each node, the two possible values of x_i are examined in the worst case. So the complexity is $\mathcal{O}(\sum_i |L_i|) = \mathcal{O}(m)$, where m is the number of nodes in B.

We can give a sharper bound on the size of B' before reduction as follows.

Theorem 3 *Suppose the algorithm of Fig. 2 is applied to a BDD B to obtain B'. Let k be the smallest index and ℓ the largest index in I. Then the number of nodes in layer i of B' is bounded above by U_i, where*

$$U_i = \begin{cases} |L_i| & \text{for } i = 1, \dots, k \\ |L_i| + \psi_i & \text{for } i = k+1, \dots, \ell \\ |L_i| & \text{for } i = \ell+1, \dots, n \end{cases}$$

The quantities ψ_i are given by $\psi_k = |L_k|$, and

$$\psi_i = \begin{cases} \min\{|L_i|, \psi_{i-1}\} & \text{if } i-1 \in I \\ \min\{|L_i|, 2\psi_{i-1}\} & \text{otherwise} \end{cases} \quad (1)$$

for $i = k+1, \dots \ell$.

An interesting special case is $I \subset \{1, \dots, \ell\}$, so that the partial assignment to be separated involves only a subset of the first ℓ variables. In this case, nodes are added only to the first ℓ layers of B to obtain B'. No nodes are added to the remainder of the BDD. We will see in the computational section below that this can be useful in practice in assignment and sequencing problems, such as the home health care scheduling problem.

Corollary 4 *If $I \subset \{1, \dots, \ell\}$, then layer i of B' contains at most $|L_i|$ nodes for $i = \ell+1, \dots, n$. In particular, if $I = \{1, \dots, \ell\}$, then layer i of B' contains at most $|L_i| + 1$ nodes for $i = 1, \dots, \ell$ and at most $|L_i|$ nodes for $i = \ell+1, \dots, n$.*

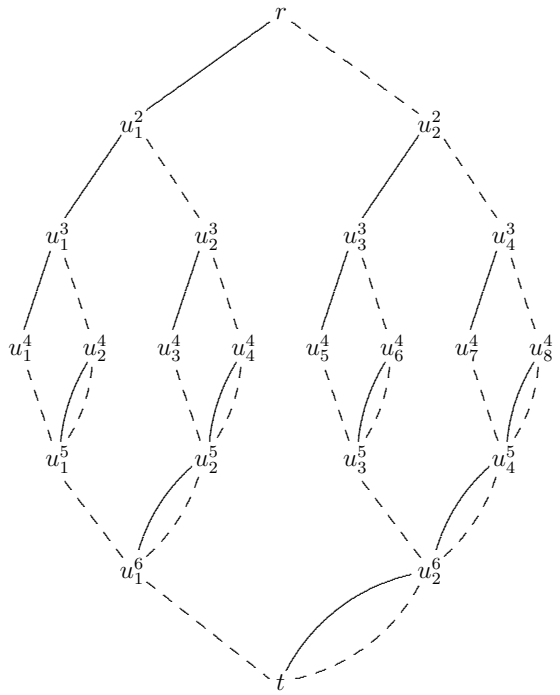

Figure 4: BDD that excludes partial assignments $x = (1, \cdot, \cdot, \cdot, \cdot, 1)$, $(\cdot, 1, \cdot, \cdot, 1, \cdot)$, and $(\cdot, \cdot, 1, 1, \cdot, \cdot)$.

The bound in Theorem 3 grows exponentially with the number of partial assignments separated. We might ask whether the separating BDD can itself grow exponentially. One might expect that it can, because otherwise one can solve the set covering problem in polynomial time, which implies $P = NP$. This is because a set covering constraint can be written $\sum_{i \in I} x_i \geq 1$, which is equivalent to excluding the solutions in $\bar{x}(I)$ where $\bar{x}_i = 0$ for $i \in I$. Thus we can let B be the diagram representing all 0-1 tuples x, and B' the diagram that separates all the set covering constraints. The set covering problem, which minimizes $\sum_i x_i$, can now be solved by placing a cost of 1 on all 1-arcs and 0 on all 0-arcs, and finding a shortest path in B'.

Theorem 5 *Given a BDD B, the number of nodes in the reduced BDD that separates the solutions in $\bar{x}^1(I_1), \dots, \bar{x}^m(I_m)$ from B can grow exponentially with respect to m and the size of B.*

The theorem is proved by letting B be a BDD of width 1 that represents all possible 0-1 solutions, and then separating the partial assignments $(x_1, x_n) = (1, 1)$, $(x_2, x_{n-1}) = (1, 1)$, $(x_3, x_{n-2}) = (1, 1)$, etc. The separating BDD for $n = 6$ appears in Fig. 4. The BDD grows exponentially with n and is reduced as well. As it happens, the above separation algorithm yields this reduced BDD.

Computational Experiments

We showed in Theorem 5 that a separating BDD can grow exponentially as partial solutions are cut off. A

key issue is whether it grows rapidly in practice. This issue can be addressed by using a BDD to solve a realistic problem. We consider the home health care scheduling problem, which has recently been solved by logic-based Benders decomposition (Ciré and Hooker 2012). The Benders master problem assigns nurses to jobs at patient homes subject to skill requirements, and the subproblem schedules visits of each nurse to the assigned patients. The objective is to find a feasible solution subject to a number of additional constraints. The Benders cuts provide a realistic sequence of partial assignments to be separated.

We let the master problem consist of a relaxed decision diagram for the assignment problem. Because the assignment variables are multivalued, we used an MDD rather than a BDD. We placed costs on the arcs to reflect each nurse's labor costs for performing the jobs, so as to favor low-cost solutions. Yet the primary objective is to find a feasible solution, which previous experience (Ciré and Hooker 2012) suggests can be difficult for this problem. The master problem is solved by finding a shortest path in the MDD, and the corresponding assignment is sent to the subproblems. Each subproblem i uses a CP solver to find a schedule for nurse i that minimizes hours on the job.

When one or more subproblems are infeasible, we identify a partial assigment $x_i = \bar{x}_i$ for $i \in I$ that creates the infeasibility, as described in (Ciré and Hooker 2012). This partial assignment serves as a Benders cut and is separated from the MDD using the algorithm of Fig. 2, slightly geeneralized to apply to MDDs.

Let x_j indicate the nurse to which job j is assigned. Let Q_j be the skill set required by job j, and $\bar{Q}_i$ the skill set of nurse i. The MDD is built for variables $x_1, \dots, x_n$ to represent a relaxation of the problem as described in (Ciré and Hooker 2012), subject to $Q_j \subseteq \bar{Q}_{x_j}$ for each j. If the diagram exceeds a specified width, it is further relaxed by merging nodes as described in (Hoda, van Hoeve, and Hooker 2010). Each arc $(u, x_j, u(x_j))$ is given cost $C_{x_j} P_j$, where C_i is the hourly cost of nurse i and P_j is the duration of job j.

In each Benders iteration, we compute the shortest-path assignment $\bar{x}$ in the MDD and solve the following scheduling subproblem for each nurse i. Each nurse begins at a home depot (job 0), travels to assigned patient homes, and returns to the depot. Each job j must be completed within time window $[R_j, D_j]$, and each nurse i's workday within $[A_i, B_i]$. If the total work time exceeds $T_{\max}$, the nurse must take a break of specified duration. The work time before or after the break be at most $T_{\max}$.

We can obtain a tighter relaxation by enlarging the MDD to include sequencing variables y_{ik}, which denotes the kth job performed by nurse i (zero if the nurse is assigned fewer than k jobs). These variables can be added to the bottom of the MDD. Corollary 4 (generalized to MDDs) implies that the Benders cuts will never enlarge this part of the diagram, because the separated partial assignments involve only the first n variables $x_1, \dots, x_n$.

Table 1: Summary results for home health care scheduling instances.

Instance	Iterations	Time (sec)
set1-n30r0	9	7.5
set1-n30r1	24	24.4
set1-n30r2	116	69.7
set1-n30r3	46	40.1
set1-n30r4	31	19.3
set1-n30r5	78	64.3
set1-n30r6	30	29.6
set1-n30r7	29	18.0
set1-n30r8	1	10.2
set1-n30r9	2	11.3
set2-n30r0	9	8.5
set2-n30r1	24	23.8
set2-n30r2	51	31.7
set2-n30r3	46	39.4
set2-n30r4	33	22.1
set3-n30r0	8	3.1
set3-n30r1	56	9.4
set3-n30r2	4	0.7
set3-n30r3	242	80.3
set3-n30r4	820	568.6

In this sense, such assignment and sequencing problems as the home health care problem are structurally suited to a separation algorithm. We omit the sequencing variables from our tests here, because they do not affect the growth rate of the separating diagram, and we solved all but one of the instances rather easily without them.

We solved 20 instances that are scaled-down versions of real-world nurse scheduling problems in Germany, first studied in (Nielsen 2006). Each instance requires that 30 jobs be assigned to 6 nurses. The subproblems were solved by the ILOG CP Optimizer, and the tests were run on an Intel Core 2 Quad with 8 GB RAM.

Table 1 summarizes the results. All instances but one were solved in roughly a minute or less, while one instance required almost 10 minutes. Aside from the hardest instance, the number of iterations is reasonable and consistent with prior applications in which logic-based Benders has been successfully applied.

Of primary interest here is the rate at which the separating MDD grows. The results appear in Figs. 5–7. The growth is close to linear in all but the hardest instance. Most of the MDDs never reach a width of 100, even though a diagram with width 1000 or more is easily managed and can be processed in a small fraction of a second. The growth rate for the hardest instance, shown in Fig. 7, is somewhat superlinear, and the separating diagram reaches a width of 16,496. However, this size was achieved only after 820 iterations. The final iteration required only 2.9 seconds, including both the processing time for the BDD and the solution of the subproblem.

Conclusion

We introduced the concept of separation for BDDs, in analogy with polyhedral separation in mathemat-

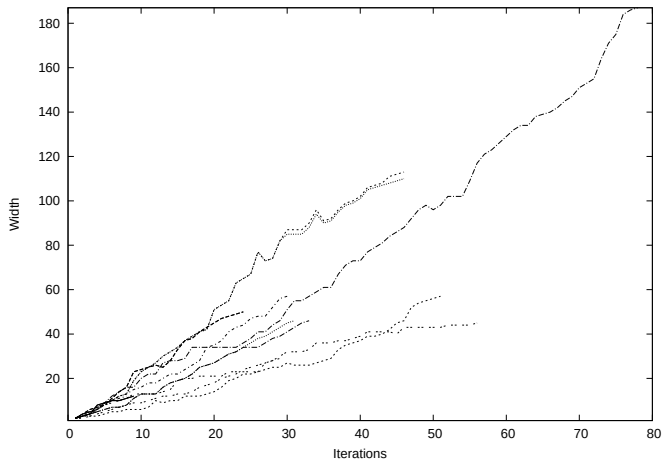

Figure 5: Growth of separating MDD for all but 3 instances.

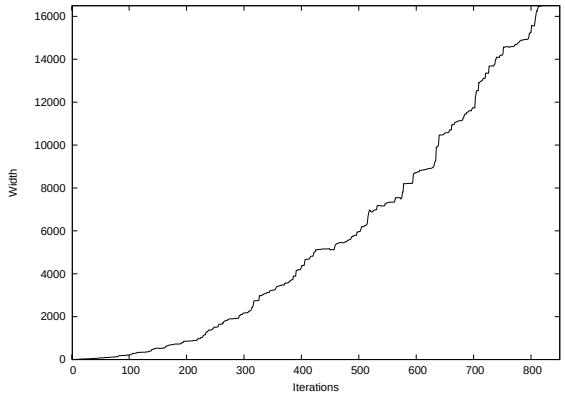

Figure 6: Growth of separating MDD for instances set1-n30r2 and set3-n30r3.

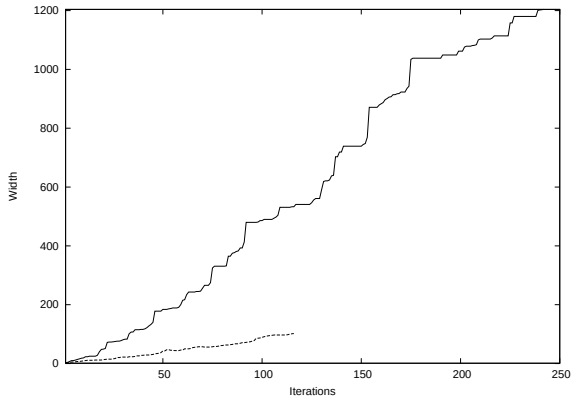

Figure 7: Growth of separating MDD for instance set3-n30r4.

ical programming. We proposed a specialized algorithm for separating partial assigments, analyzed its complexity, and proved upper and lower bounds on the growth of the separating BDD when a sequence of partial solutions is cut off. We demonstrated the use of separation in a logic-based Benders algorithm for the home health care scheduling problem. We found that the separating BDD grows only linearly with the number of Benders iterations in almost every case. This indicates that BDD-based separation could be a practical mechanism to implement Benders cuts. More generally, it suggests that BDD-based separation could play an algorithmic role in optimization and constraint programming as a technique for strengthening discrete relaxations, much as cutting planes strengthen continuous relaxations in mathematical programming.

References

Akers, S. B. 1978. Binary decision diagrams. *IEEE Transactions on Computers* C-27:509–516.

Andersen, H. R.; Hadžić, T.; Hooker, J. N.; and Tiedemann, P. 2007. A constraint store based on multivalued decision diagrams. In Bessiere, C., ed., *Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *Lecture Notes in Computer Science*, 118–132. Springer.

Andersen, H. R. 1997. An introduction to binary decision diagrams. Lecture notes, available online, IT University of Copenhagen.

Becker, B.; Behle, M.; Eisenbrand, F.; and Wimmer, R. 2005. BDDs in a branch and cut framework. In Nikoletseas, S., ed., *Experimental and Efficient Algorithms, Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA 05)*, volume 3503 of *Lecture Notes in Computer Science*, 452–463. Springer.

Behle, M., and Eisenbrand, . 2007. 0/1 vertex and facet enumeration with BDDs. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments and the 4th Workshop on Analytic Algorithms and Combinatorics*, 158–165.

Benini, L.; Bertozzi, D.; Guerri, A.; and Milano, M. 2005. Allocation and scheduling for MPSoCs via decomposition and no-good generation. In *Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, 107–121. Springer.

Bertels, S., and Fahle, T. 2006. A hybrid setup for a hybrid scenario: Combining heuristics for the home health care problem. *Computers and Operations Research* 33:2866–2890.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* C-35:677–691.

Cambazard, H.; Hladik, P.-E.; Déplanche, A.-M.; Jussien, N.; and Trinquet, Y. 2004. Decomposition and learning for a hard real time task allocation problem. In Wallace, M., ed., *Principles and Practice of Constraint*

Programming (CP 2004), volume 3258 of *Lecture Notes in Computer Science*, 153–167. Springer.

Cheng, E., and Rich, J. L. 1998. A home health care routing and scheduling problem. Technical Report TR98-04, Department of CAMM, Rice University, USA.

Chu, Y., and Xia, Q. 2004. Generating Benders cuts for a class of integer programming problems. In Régin, J. C., and Rueher, M., eds., *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011 of *Lecture Notes in Computer Science*, 127–141. Springer.

Ciré, A., and Hooker, J. N. 2012. A heuristic logic-based benders method for the home health care problem. In *manuscript, presented at Matheuristics 2012*.

Hadžić, T., and Hooker, J. N. 2006. Postoptimality analysis for integer programming using binary decision diagrams, presented at GICOLAG workshop (Global Optimization: Integrating Convexity, Optimization, Logic Programming, and Computational Algebraic Geometry), Vienna. Technical report, Carnegie Mellon University.

Hadžić, T., and Hooker, J. N. 2007. Cost-bounded binary decision diagrams for 0-1 programming. Technical report, Carnegie Mellon University.

Hadžić, T.; Hooker, J. N.; O'Sullivan, B.; and Tiedemann, P. 2008. Approximate compilation of constraints into multivalued decision diagrams. In Stuckey, P. J., ed., *Principles and Practice of Constraint Programming (CP 2008)*, volume 5202 of *Lecture Notes in Computer Science*, 448–462. Springer.

Harjunkoski, I., and Grossmann, I. E. 2001. A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering* 25:1647–1660.

Harjunkoski, I., and Grossmann, I. E. 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering* 26:1533–1552.

Hoda, S.; van Hoeve, W.-J.; and Hooker, J. N. 2010. A systematic approach to MDD-based constraint programming. In *Proceedings of the 16th International Conference on Principles and Practices of Constraint Programming*, Lecture Notes in Computer Science. Springer.

Hooker, J. N., and Ottosson, G. 2003. Logic-based Benders decomposition. *Mathematical Programming* 96:33–60.

Hooker, J. N., and Yan, H. 1995. Logic circuit verification by Benders decomposition. In Saraswat, V., and Hentenryck, P. V., eds., *Principles and Practice of Constraint Programming: The Newport Papers*, 267–288. Cambridge, MA: MIT Press.

Hooker, J. N. 1995. Logic-based Benders decomposition. In *INFORMS National Meeting (INFORMS 1995)*.

Hooker, J. N. 2000. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. New York: Wiley.

Hooker, J. N. 2013. Decision diagrams and dynamic programming. In Gomes, C., and Sellmann, M., eds., *CPAIOR 2013 Proceedings*, 94–110.

Hu, A. J. 1995. Techniques for efficient formal verification using binary decision diagrams. Thesis CS-TR-95-1561, Stanford University, Department of Computer Science.

Jain, V., and Grossmann, I. E. 2001. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing* 13:258–276.

Lee, C. Y. 1959. Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal* 38:985–999.

Maravelias, C. T., and Grossmann, I. E. 2004a. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers and Chemical Engineering* 28:1921–1949.

Maravelias, C. T., and Grossmann, I. E. 2004b. Using MILP and CP for the scheduling of batch chemical processes. In Régin, J. C., and Rueher, M., eds., *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011 of *Lecture Notes in Computer Science*, 1–20. Springer.

Marchand, H.; Martin, A.; Weismantel, R.; and Wolsey, L. 2002. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics* 123:397–446.

Nielsen, A. 2006. Optimization of a health care system. Master's thesis, Züse Institute Berlin.

Rendl, A.; Prandtstetter, M.; Hiermann, G.; Puchinger, J.; and Raidl, G. 2012. Hybrid heuristics for multimodal homecare scheduling. In Beldiceanu, N.; Jussien, N.; and Pinson, E., eds., *International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2012)*, volume 7298 of *Lectures Notes in Computer Science*, 339–355. Nantes, France: Springer Verlag.

Steeg, J., and Schröder, M. 2007. A hybrid approach to solve the periodic home health care problem. In *Operations Research Proceedings 2007 - Selected Papers of the Annual International Conference of the German Operations Research Society (GOR)*, 297–302.

Terekhov, D.; Beck, J. C.; and Brown, K. N. 2005. Solving a stochastic queueing design and control problem with constraint programming. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2005)*, 261–266.