

Bucknell University

Bucknell Digital Commons

Faculty Conference Papers and Presentations

Faculty Scholarship

Summer 2020

Enumerative Branching with Less Repetition

Thiago Serra

Bucknell University, tsa005@bucknell.edu

Follow this and additional works at: https://digitalcommons.bucknell.edu/fac_conf



Part of the [Theory and Algorithms Commons](#)

Recommended Citation

Serra, Thiago, "Enumerative Branching with Less Repetition" (2020). *Faculty Conference Papers and Presentations*. 58.

https://digitalcommons.bucknell.edu/fac_conf/58

This Conference Paper is brought to you for free and open access by the Faculty Scholarship at Bucknell Digital Commons. It has been accepted for inclusion in Faculty Conference Papers and Presentations by an authorized administrator of Bucknell Digital Commons. For more information, please contact dcadmin@bucknell.edu.

Enumerative Branching with Less Repetition

Thiago Serra

Bucknell University, USA
thiago.serra@bucknell.edu

Abstract. We can compactly represent large sets of solutions for problems with discrete decision variables by using decision diagrams. With them, we can efficiently identify optimal solutions for different objective functions. In fact, a decision diagram naturally arises from the branch-and-bound tree that we could use to enumerate these solutions if we merge nodes from which the same solutions are obtained on the remaining variables. However, we would like to avoid the repetitive work of finding the same solutions from branching on different nodes at the same level of that tree. Instead, we would like to explore just one of these equivalent nodes and then infer that the same solutions would have been found if we explored other nodes. In this work, we show how to identify such equivalences — and thus directly construct a reduced decision diagram — in integer programs where the left-hand sides of all constraints consist of additively separable functions. First, we extend an existing result regarding problems with a single linear constraint and integer coefficients. Second, we show necessary conditions with which we can isolate a single explored node as the only candidate to be equivalent to each unexplored node in problems with multiple constraints. Third, we present a sufficient condition that confirms if such a pair of nodes is indeed equivalent, and we demonstrate how to induce that condition through preprocessing. Finally, we report computational results on integer linear programming problems from the MIPLIB benchmark. Our approach often constructs smaller decision diagrams faster and with less branching.

Keywords: Branch-and-Bound · Decision Diagrams · Depth-First Search · Integer Programming · Solution Enumeration.

1 Introduction

The enumeration of near-optimal solutions is a feature that is present in commercial MILP solvers such as CPLEX [33] and Gurobi [26] as well as algebraic modelling environments such as GAMS [25]. This feature is important because some users need to qualitatively compare the solutions of a mathematical model. However, those solutions are often a small set collected along the way towards solving for the optimal solution, as in Gurobi [26]. While the option for complete enumeration exists in CPLEX, it comes with the following observation [33]:

Beware, however, that, even for small models, the number of possible solutions is likely to be huge. Consequently, enumerating all of them will take time and consume a large quantity of memory.

In fact, the problem of enumerating integer solutions is #P-complete [49, 50].

In practice, the enumeration of solutions has been an extension of the same methods used for optimization. When searching for an optimal solution of a problem with linear constraints and integer variables, the first step is often to solve a relaxation of this problem: a linear program in which we ignore that the variables should have integer values [18]. If the resulting solution has fractional values for some of the variables, then we may resort to *branching*: exploring two or more subproblems in which we fix or restrict the domain of these variables to exclude their corresponding fractional values. We can ignore some of the resulting subproblems if they are provably suboptimal, for which reason this process is known as *branch-and-bound* [36]. However, if we are interested in enumerating some or all the solutions, then we may need to keep branching even if no value is fractional. In such a case, we continue while the domains of the variables have multiple values, the relaxation remains feasible, and the objective function value is within a desired optimality gap. That is the case of the one-tree approach [19], which has been used to populate the solution pool of the CPLEX solver [32].

The branch-and-bound process is often represented by a directed tree, the *branch-and-bound tree*, in which a root node r corresponds to the problem of interest and the children of each node are the subproblems defined by branching. If we branch by assigning the value of one decision variable at a time to each possible value, then a path from the root to a leaf defining a feasible subproblem corresponds to a set of assignments leading to a unique solution. Such explicit representation of the solution set emerging from branch-and-bound, which may reach an exponential number of nodes with respect to the number of variables, could be naturally transformed into a *decision diagram* by merging the leaves of the tree as a single terminal node t [44]. Likewise, we can merge any other nodes from which the same solutions are obtained on the remaining variables while preserving a correspondence between solutions of the problem and r - t paths in the diagram. That results in a representation that is substantially more compact and, in some cases, can have less nodes than the number of solutions represented. If every path assigns a value to every variable, then we can find optimal solutions for varied linear objective functions by assigning the corresponding weights to the arcs and computing a minimum weight r - t path, which is usually faster than resolving the problem with different objective functions [44].

Figure 1 illustrates these different graphic representations for the solutions of inequality $2x_1 - 2x_2 - 3x_3 \leq -1$ on a vector of binary variables $x \in \{0, 1\}^3$: (a) is a tree in which every path from the root to a different leaf assigns a distinct set of values to the variables; (b) is the decision diagram produced by merging the leaves of the tree; and (c) is a decision diagram in which we merge the three nodes in (b) from which the only arc towards t corresponds to $x_3 = 1$. We use thin arcs for assignments of value 0 and bold arcs for assignments of value 1. In the example, we always assign a value to variable x_1 first, then to variable x_2 , and finally to variable x_3 . When the same order of variables is used in every r - t path, then nodes at the same arc-distance from r define a *layer* with the same *selector variable* for the next assignment, and we say that we have an *ordered*

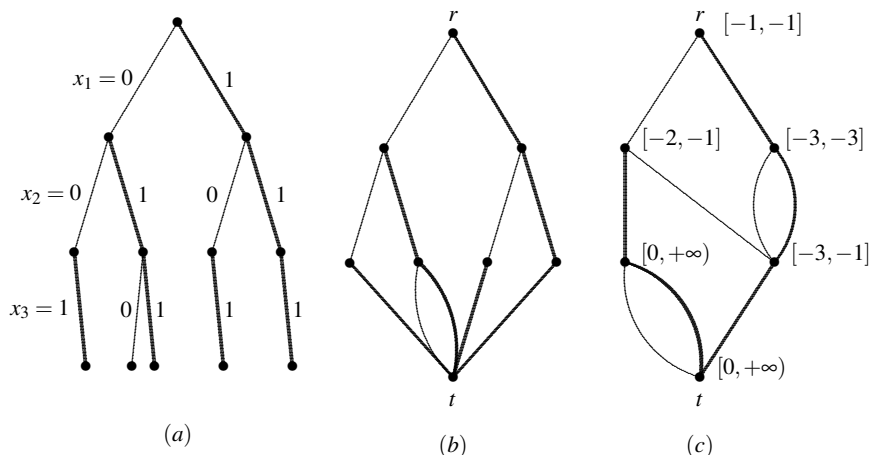


Fig. 1. (a) Tree representing the solutions of $\{x \in \{0, 1\}^3 : 2x_1 - 2x_2 - 3x_3 \leq -1\}$. (b) Decision diagram from merging the leaf nodes. (c) Reduced decision diagram with interval of equivalent right-hand side values for the inequality on each node.

decision diagram. In fact, Figure 1 (c) has the smallest ordered decision diagram for that sequence of variable selectors, and we call it a *reduced* decision diagram.

In this paper, we discuss how to compare the formulation of subproblems involving one or more inequalities to determine the equivalence of branch-and-bound nodes, and hence directly construct reduced decision diagrams.

1.1 Contribution

We generalize prior work on identifying equivalent subproblems with a single inequality [6, 1, 2] and introduce a variant for the case of multiple inequalities. First, we show how to compute the interval of equivalent Right-Hand Side (RHS) values for any inequality on finite domains with additively separable Left-Hand Side (LHS) and fractional RHS. Second, we discuss why the same idea cannot be directly applied to problems with multiple inequalities. In that case, we show how to eliminate candidates for equivalence among the explored nodes in such a way that we are left with at most one potentially equivalent node. Finally, we present a sufficient condition achievable by bottom-up preprocessing to determine if such a remaining node is indeed equivalent.

2 Decision Diagrams

A *decision diagram* is a directed acyclic graph with root node r . Arcs leaving each node denote assignments to a variable, which we denote its *selector variable*. If the variables can only take two possible values, the diagram is *binary*. If the variables can take more values, the diagram is *multi-valued*. When representing

only feasible solutions, there is one terminal node t and each solution is mapped to an r - t path. Otherwise, there are two terminal nodes \mathcal{T} and \mathcal{F} , with feasible solutions corresponding to r - \mathcal{T} paths and infeasible solutions to r - \mathcal{F} paths. We say that nodes at the same distance (in arcs) from the root node r are in the same *layer* of the diagram. The sets of paths from a given node u toward t (or \mathcal{T}) define the *solution set* of that node. Nodes have the same *state* if those sets coincide. A diagram is *reduced* if no two nodes have the same state. A diagram is *ordered* if all nodes in each layer have the same selector variable.

Bryant [14] has shown that we can efficiently reduce a decision diagram through a single bottom-up pass by identifying and merging nodes with equivalent sets of assignments on the remaining variables.

Two factors may help us obtain smaller decision diagrams. First, equivalences are intuitively more frequent in ordered decision diagrams. In such a case, all nodes in each layer have solution sets defined on the same set of variables, which is a necessary condition to merge such nodes. Second, it is easier to identify equivalences if the problem is defined by inequalities in which the *Left-Hand Side* (LHS) is an *additively separable function*. A function $f(x)$ on an n -dimensional vector of variables x is additively separable if we can decompose it as $f(x) = \sum_{i=1}^n f_i(x_i)$, i.e., a sum of univariate functions on each of those variables. In such a case, all the nodes in a given layer define subproblems with inequalities having the same LHS. For example, if we have a problem $\{x : \sum_{i=1}^n f_i(x_i) \leq \rho\}$, then each node obtained by assigning a distinct value \bar{x}_1 to variable x_1 defines a subproblem $\{x : \sum_{i=2}^n f_i(x_i) \leq \rho - f_1(\bar{x}_1)\}$ on the same $n - 1$ variables and with the same LHS for the inequality.

Following those conditions, we can anticipate that two nodes have the same solutions if their subproblems have inequalities with the same *Right-Hand Side* (RHS) values. In such a case, we can save time if we just branch on one of those nodes to enumerate its solutions and then merge it with the second node. For example, in the problem of Figure 1 we obtain the same subproblem $\{x_3 \in \{0, 1\} : -3x_3 \leq -1\}$ with assignments $x_1 = 0$ and $x_2 = 0$ or $x_1 = 1$ and $x_2 = 1$.

More generally, we say that two subproblems have *equivalent formulations* if they have the same solutions, even if the formulations are different. For example, in Figure 1 we obtain subproblem $\{x_3 \in \{0, 1\} : -3x_3 \leq -3\}$ by assigning $x_1 = 1$ and $x_2 = 0$, which has the same solution set as the previously mentioned assignments leading to $\{x_3 \in \{0, 1\} : -3x_3 \leq -1\}$. A method to compute such equivalent RHS values for nodes in the same layer of ordered decision diagrams representing a linear inequality with integer coefficients has been independently proposed twice on binary domains [1, 6] and later extended to multi-valued domains [2]. This method is based on search for all the solutions for some nodes and then inferring if a new node would have the same solutions as one of those nodes. We will use the following definition for those previous nodes.

Definition 1 (Explored node). *A node is said to be explored if all the solutions for the subproblem rooted at that node are known.*

Once a node has been explored, we can compute the minimum RHS value that would produce the same solutions and the minimum RHS value, if any, that

would produce more solutions. In Figure 1 (c), the interval of equivalent integer values for the RHS of the inequalities on the remaining variables is shown next to each node. For example, in the penultimate layer we have the intervals of RHS values for $-3x_3$ as LHS, which includes $[-3, -1]$ for the node that is reached by the three assignments to x_1 and x_2 that we discussed previously.

However, we cannot directly apply the same method to problems with multiple inequalities and expect to find all the nodes that are equivalent. For example, in Figure 2 we have three equivalent formulations for the solution set $\{(0, 0), (0, 1), (1, 0)\}$. The inequality with LHS $5x_1 + 4x_2$ has equivalent RHS values in $[5, 8]$. The inequality with LHS $6x_1 + 10x_2$ has equivalent RHS values in $[10, 15]$. But there are equivalent RHS values beyond $[5, 8] \times [10, 15]$: if the RHS of only one inequality is made larger, such as in the first two examples of Figure 2, the other inequality prevents the inclusion of solution $(1, 1)$. In other words, not every inequality needs to separate every infeasible solution.

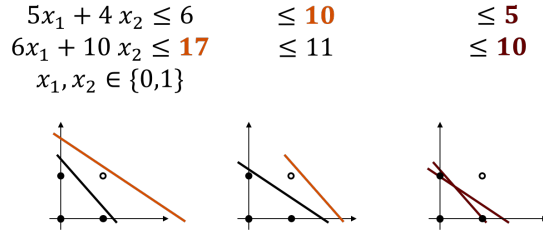


Fig. 2. Equivalent formulations in which the inequalities have the same functions in the Left-Hand Side (LHS) but different values in the Right-Hand Side (RHS).

3 Related Work

Put in context, our work may also improve exhaustive search. Even in cases where the goal is not to construct a decision diagram for the set of solutions, the characterization of a *state* for the nodes of the corresponding decision diagram based on their solution sets can be exploited to avoid redundant work during the branch-and-bound process.

Beyond the case in which all RHS values are the same, prior work on identifying equivalent branch-and-bound nodes has focused on detecting inequalities that are always satisfied. One example is the detection of *unrestricted subtrees*, in which any assignment to the remaining variables is feasible [3]. The same principle was later used to ignore all unrestricted inequalities and only compare the RHS values of inequalities that separate at least one solution [44]. That can be particularly helpful with set covering constraints, in which it suffices to have one variable assigned to 1. This line of work relies on computing the minimum RHS value after which each inequality does not separate any solution, and

therefore any RHS value larger than that is deemed equivalent. For example, in Figure 1 (c) the first node in the penultimate layer allows both possible values to x_3 , which for the LHS $-3x_3$ implies a RHS of 0 or more. In contrast to [44], the present paper aims to identify all nodes that can be merged during the top-down construction of a decision diagram corresponding to an ILP formulation. Hence, we also compare different RHS values that exclude at least one solution.

While the search effort may decrease, it can still be substantial and depend on other factors. First, the size of a reduced decision diagram for a single constraint can be exponential on the number of variables for any order of variable selectors [30]. Second, finding a better order of variable selectors for binary decision diagrams is NP-complete [13], hence implying that it is NP-hard to find an order of variable selections that minimizes the size of the decision diagram [22, 6]. Nevertheless, some ordering heuristics have been found to work well in practice [6, 7], while strong bounds for the size of the decision diagrams according to the ordering of variable selectors have been shown for certain classes of problems [28]. In other cases, a convenient order may be deduced from the context of the problem. For example, by identifying a temporal relationship among the decision variables, such as in sequencing and scheduling problems [17, 42].

A related approach consists of analyzing *dominance relations* among branch-and-bound nodes [35, 31]. When exploring a node u , we may wonder if there is another node v that can be reached by fixing the same variables with different values such that (i) the assignments leading to v are preferable according to the objective function; and (ii) v defines a relaxation of the subproblem defined by u , in which case all solutions that can be found by exploring u can also be found by exploring v [24, 23, 16]. If such a node v exists and we only want to find an optimal solution, then we can safely prune the subtree rooted at u and only explore the subtree rooted at v . However, such an approach is not fully compatible with enumerating solutions because the subproblems on u and v may not have the same set of solutions. Furthermore, a change in the objective function could make the discarded node u preferable to node v in applications of reoptimization. Finally, finding such a node v may entail solving an optimization problem on the variables that are fixed to reach node u , and thus it only pays off closer to the root node because not many variables have been assigned yet.

In contrast, our approach pays off closer to the bottom of the decision diagram representation. For a node at distance k from r , there are at most 2^k states if all variables have binary domains, which are the distinct assignments to the first k variables. For a node at distance k from t , there are at most $2^{2^k} - 1$ states, which are the non-empty sets of solutions for the last k variables. Hence, we should generally expect that equivalences occur more often when the number of top-down states exceeds the number of bottom-up states.

In the context of integer linear programming, a small set of solutions is often obtained through the application of inequalities separating previous solutions [5, 15]. Recent work has also focused on the generation of a diverse set of solutions [20, 48, 41], and there are also methods to obtain exact upper bounds [34] and probabilistic lower bounds [46] on the size of the solution set.

Decision diagrams and some extensions have also been widely used to solve combinatorial optimization problems [4, 43, 29, 40, 8, 9, 38, 53, 51, 52, 47], and more recently stochastic [27, 37, 45] and nonlinear optimization problems [10, 21].

4 Prior Result

We begin our analysis with a prior result about the direct construction of reduced decision diagrams, which concerns the decision diagram rooted at a node v and defined by an inequality of the form $a_1x_1 + \dots + a_nx_n \leq a_0$ with integer coefficients and each decision variable x_i having a discrete domain of the form $\{0, 1, \dots, d_i\}$. For such a node v and each of the other nodes in the decision diagram, we want to compute a corresponding interval $[\beta, \gamma]$ or $[\beta, \infty)$, where β is the smallest integer RHS value that could replace a_0 and still yield the same solutions. Similarly, γ is the largest integer RHS value that would still yield the same solutions, and γ only exists if at least one solution is infeasible. We recursively compute such intervals after exploring a node and all of its descendants.

Theorem 1 (Abío & Stuckey¹ [2]). *Let \mathcal{M} be the multi-valued decision diagram of a linear integer constraint $a_1x_1 + \dots + a_nx_n \leq a_0$. Then, the following holds:*

- *The interval of the true node \mathcal{T} is $[0, \infty)$.*
- *The interval of the false node \mathcal{F} is $(-\infty, -1]$.*
- *Let v be a node with selector variable x_i and children $\{v_0, v_1, \dots, v_{d_i}\}$. Let $[\beta_j, \gamma_j]$ be the interval of v_j . Then, the interval of v is $[\beta, \gamma]$, with $\beta = \max\{\beta_s + sa_i \mid s \in \{0, \dots, d_i\}\}$, $\gamma = \min\{\gamma_s + sa_i \mid s \in \{0, \dots, d_i\}\}$.*

The interval of the explored nodes can then be compared with the RHS of the inequalities defining each of the unexplored nodes in the same layer to identify equivalences. In order to fully avoid redundant work, the branch-and-bound algorithm should perform a *depth-first search* (DFS), where the unexplored node with fewer unfixed variables is explored next. In such a case, we do not risk branching on two nodes that would later be found to have the same state.

In this paper, we consider decision diagrams with a single terminal node t , hence disregarding infeasible solutions and nodes in which roughly $\beta = -\infty$.

5 The Case of One Inequality

In this section, we discuss how Theorem 1 can be further generalized when applied to a single inequality. Our contribution is evidencing that the sequence of work culminating in the result by Abío and Stuckey [2] can be extended to account for the case of linear inequalities with fractional coefficients, and to the slightly more general case of inequalities involving additively separable functions.

¹ Following the recommendation of one of the anonymous reviewers, there is small correction in comparison to [2]: we use $s \in \{0, \dots, d_i\}$ instead of $0 \leq s \leq d_i$.

We begin with an intuitive argument for generalizing that result for the case of fractional RHS values. In the interval $[\beta, \gamma]$ used in Theorem 1, both β and γ are integer values corresponding to the smallest and largest integer RHS values defining the same solutions for the inequality. If the LHS coefficients and the decision variables are integer, then it follows that any right-hand side value larger than γ but smaller than $\gamma + 1$ would also define the same solutions. Hence, $[\beta, \gamma + 1)$ is the maximal interval of equivalent right-hand side values if we allow a fractional right-hand side. Now the upper end value $\gamma + 1$ is the smallest RHS value yielding a proper superset of solutions. Furthermore, with such half-closed intervals, both ends may also become fractional if the LHS coefficients are fractional and assigning a variable changes the RHS by a fractional amount.

In addition, the only reason to represent infeasible solutions is to compute those upper ends of the RHS intervals. But for each node that has missing arcs for some values of its selector variable, which if represented would only reach the terminal \mathcal{F} , the corresponding upper end is the sum of the impact of that assignment with the smallest RHS associated with a solution on the remaining variables. But as we will see next, that value can be calculated by inspection.

In summary, we can ignore infeasible solutions by using a single terminal node t , lift the integrality of all coefficients, and consequently allow the LHS to be additively separable. That leads us to the following result:

Theorem 2. *Let \mathcal{D} be a decision diagram with variable ordering x_1, x_2, \dots, x_n of $f(x) = f_1(x_1) + \dots + f_n(x_n) \leq \rho$ on finite domains D_1 to D_n . Then, the following holds:*

- The interval of the terminal node t is $[0, \infty)$.
- Let v be a node with selector variable x_i and non-empty multi-set² of descendants $\{v_j \mid j \in D^v\}$, $D^v \subseteq D_i$, where v_j is reached by an arc denoting $x_i = j$. Let $[\beta_j, \gamma_j)$ be the interval of v_j . The interval of v is $[\beta, \gamma)$, with

$$\beta = \max\{\beta_j + f_i(j) \mid j \in D^v\}$$

and

$$\gamma = \min \left\{ \min\{\gamma_j + f_i(j) \mid j \in D^v\}, \min\{\xi_i + f_i(j) \mid j \in D_i \setminus D^v\} \right\},$$

where

$$\xi_i = \sum_{l=i+1}^n \min\{f_l(d) \mid d \in D_l\}.$$

Since we can prove a stronger result regarding the lower end β , which also applies to the case of multiple inequalities that will be discussed in Section 6, we split the proof of Theorem 2 into two lemmas.

² We use a multi-set because two nodes might be connected through multiple arcs for different variable assignments.

Lemma 1. *Let \mathcal{D} be a decision diagram with variable ordering x_1, x_2, \dots, x_n of $\{(x_1, x_2, \dots, x_n) \in D_1 \times D_2 \times \dots \times D_n \mid f^k(x) = f_1^k(x_1) + \dots + f_n^k(x_n) \leq \rho^k \ \forall k = 1, \dots, m\}$. Then, computing β^k for each inequality $f^k(x) \leq \rho^k$ as in Theorem 2 yields the smallest RHS not affecting the set of solutions satisfying all inequalities on each node.*

Proof. We proceed by induction on the layers, starting from the bottom. The base case holds since $0 \leq \rho^k$ is only valid if $\rho^k \geq \beta^k = 0$ for each inequality k at the terminal node t . Now suppose, for contradiction, that Lemma 1 holds for the $n - i$ bottom layers of \mathcal{D} and not for the one above. Hence, there would be a node v with selector variable x_i such that v has the same solutions if we replace the RHS of the k -th inequality with some $\delta < \beta^k \leq \rho^k$, thereby obtaining $\sum_{l=i}^n f_l^k(x_l) \leq \delta$. Let $j \in \arg \max\{\beta_j^k + f_i^k(j) \mid j \in D^v\}$, i.e., v_j would be one of the children maximizing the expression with which we calculate β^k in the statement of Lemma 1 and β_j^k is the smallest RHS value not affecting the set of solutions of v_j with respect to the k -th inequality. Consequently, node v_j would have the same solutions if the k -th inequality for v_j becomes $\sum_{l=i+1}^n f_l^k(x_l) \leq \delta - f_i^k(j)$. However, $\beta_j^k = \beta^k - f_i^k(j) > \delta - f_i^k(j)$, and we have a contradiction since by induction hypothesis any value smaller than β_j^k would yield proper subset of the solution set of v_j . \square

Lemma 2. *Computing γ for one inequality as in Theorem 2 yields the smallest RHS value that would yield a proper superset of the solutions of node v .*

Proof. We proceed by induction on the layers, starting from the bottom. The base case holds since the terminal node t has a set with one empty solution and we denote $\gamma = +\infty$.

By induction hypothesis, we assume that Lemma 2 holds for the $n - i$ lower layers of \mathcal{D} , and we show next that it consequently holds for the i -th layer of \mathcal{D} .

For any node v in the i -th layer with interval $[\beta, \gamma)$ for a finite γ such that $\sum_{l=i}^n f_l(x_l) \leq \gamma$, let $j \in \arg \min\left\{\min\{\gamma_j + f_i(j) \mid j \in D^v\}, \min\{\xi_i + f_i(j) \mid j \in D_i \setminus D^v\}\right\}$. If (I) $j \in \arg \min\{\gamma_j + f_i(j) \mid j \in D^v\}$, then there is a child node v_j that minimizes the expression with which we calculate γ in the statement of Lemma 2. By induction hypothesis, that implies that there is a solution that is not feasible for node v_j and such that $\sum_{l=i+1}^n f_l(\bar{x}_j) = \gamma_j$. Consequently, solution $(\bar{x}_i = j, \bar{x}_{i+1}, \dots, \bar{x}_n)$ is not feasible for node v and $\sum_{l=i}^n f_l(\bar{x}_j) = \gamma$. If (II) $j \in \arg \min\{\xi_i + f_i(j) \mid j \in D_i \setminus D^v\}$, then $\gamma = f_i(j) + \xi_i$ and node v does not have any solution in which $x_i = j$. Consequently, there is an infeasible solution $(\bar{x}_i = j, \bar{x}_{i+1}, \dots, \bar{x}_n)$ such that $\sum_{l=i+1}^n f_l(\bar{x}_j) = \xi_i$ and $\sum_{l=i+1}^n f_l(\bar{x}_j) = \gamma$. In either case, a RHS of γ yields a proper superset of the solutions of v . Furthermore, a smaller RHS value yielding a proper superset of the solutions of v would either contradict the choice of j in (I) if there is at least one solution of v such that $x_i = j$ or in (II) if there is no solution of v such that $x_i = j$. \square

We are now able to prove the main result of this section.

Proof (Theorem 2). Lemma 1 implies that β is the smallest RHS value yielding the same solution set as v and Lemma 2 implies that γ is the smallest RHS value yielding at least one more solution than node v . Since there is a single inequality, then a RHS of γ yields a proper superset of the solutions of v . \square

6 The Case of Multiple Inequalities

For variables x_1 to x_n with finite domains D_1 to D_n , we now consider constructing a decision diagram for a problem defined by m inequalities in the following form:

$$f^i(x) = f_1^i(x_1) + \dots + f_n^i(x_n) \leq \rho^i \quad \forall i = 1, \dots, m$$

We will exploit the fact that we still can apply Lemma 1 to a problem with multiple inequalities. Theorem 2 is not as helpful because the equivalent upper ends for one inequality may depend on the RHS values of other inequalities. That prevents us from immediately identifying if two nodes are equivalent by comparing the intervals of the explored node with the RHS values of the unexplored node. Nevertheless, we can characterize and distinguish nodes having different solution sets by their lower ends as in Lemma 1. We show that such comparison is enough to exclude all but one of the explored nodes as potentially equivalent, to which we describe a sufficient condition to guarantee equivalence.

6.1 Necessary Conditions

With multiple inequalities, we cannot simply use the intervals of RHS values associated with each of the inequalities independently. We have previously observed that with the example in Figure 2, which we now revisit with half-closed intervals. For the inequality with LHS $5x_1 + 4x_2$, the solution set $\{(0, 0), (0, 1), (1, 0)\}$ corresponds to RHS values in $[5, 9)$. For the inequality with LHS $6x_1 + 10x_2$, that same solution set corresponds to RHS values in $[10, 16)$. Hence, $[5, 9) \times [10, 16)$ defines a valid combination of RHS values yielding the same solutions. However, we can relax one inequality if the other still separates the remaining infeasible solution $(1, 1)$. Consequently, the solution set is actually characterized by following combination of RHS values: $[5, 9) \times [10, +\infty) \cup [5, +\infty) \times [10, 16)$.

Note that the *lower ends* in β are nevertheless the same. In fact, Lemma 1 implies that we can characterize node states by the smallest RHS value of each inequality that would yield the same solutions. The key difference is that pushing any RHS lower than β restricts the solution set, whereas increasing some RHS to γ or above may not enlarge the solution set if another inequality separates the solutions that would otherwise be included. Hence, we ignore the upper ends in what follows and focus on the consequences of Lemma 1.

In what follows, let v be an unexplored node with RHS values ρ^1 to ρ^m and let \bar{v} and $\bar{\bar{v}}$ be explored nodes with lower ends $\bar{\beta}^1$ to $\bar{\beta}^m$ and $\bar{\bar{\beta}}^1$ to $\bar{\bar{\beta}}^m$, respectively.

Corollary 1 (Main necessary condition). *Node v is equivalent to node \bar{v} only if $\rho^k \geq \bar{\beta}^k$ for $k = 1, \dots, m$.*

Proof. Lemma 1 implies that $\rho^k < \bar{\beta}^k$ for any inequality k would make a solution of \bar{v} infeasible for v . Conversely, if $\rho^k \geq \bar{\beta}^k$ for $k = 1, \dots, m$, then v has at least the same solutions as \bar{v} , a necessary condition for equivalence. \square

Corollary 2 (Dominance elimination). *Node v is equivalent to an explored node \bar{v} only if no other explored node \bar{v} for which v satisfies the necessary condition has a strictly larger lower end for any of the inequalities, i.e., there is no such \bar{v} for which $\bar{\beta}^k > \beta^k$ for any $k = 1, \dots, m$.*

Proof. If two nodes \bar{v} and \bar{v} have different lower ends for inequality k and they are such that $\bar{\beta}^k > \bar{\beta}^k$, then \bar{v} has a solution requiring a larger RHS value on inequality k to be feasible. Thus, \bar{v} does not have such a solution. However, if both \bar{v} and \bar{v} have a subset of the solutions of node v , then v also has that solution and thus cannot be equivalent to \bar{v} . \square

Corollary 3 (One or none). *Node v has at most one explored node satisfying both the necessary condition and the dominance elimination in a reduced decision diagram.*

Proof. Since the lower ends characterize the state of a node and no two nodes have the same state in a reduced decision diagram, then any pair of explored nodes will differ in at least one lower end value. Consequently, no more than one explored node can satisfy both conditions for node v . \square

When multiple nodes meet the necessary condition of having at least as many solutions as node v , we can eliminate some by dominance. If they differ in the lower end of some inequality, that implies that node v and one of them have a solution that the other does not have, hence allowing us to discard the latter. In fact, explored nodes can eliminate one another in different inequalities. Since no two nodes have all lower ends matching in a reduced decision diagram, no more than one node is left as candidate, but possibly none is.

6.2 A Sufficient Condition

From Corollary 3, we are left with at most one explored node \bar{v} that could be equivalent to a given unexplored node v . If there is no such node, then the solution set of v is distinct from all the solution sets of explored nodes in the layer. Otherwise, the solution set of such node \bar{v} is different from that of v only if the solutions of node v are a proper superset of the solutions of \bar{v} . If the layer has explored nodes corresponding to every possible solution set, then nodes v and \bar{v} would be equivalent. However, having all such nodes would be prohibitive.

Alternatively, we can individually consider each of the solutions that could be missing from \bar{v} . If a given layer has explored nodes containing each one of them, then node \bar{v} is always equivalent. We use the following definition for sufficiency:

Definition 2 (Populated layer). *A layer is said to be populated if it has explored nodes corresponding to the minimal solution set containing each of the solutions on the remaining variables.*

For a solution x in a populated layer, there is a node v_x with lower ends $\beta(x)$ corresponding to the tightest RHS values for which x is feasible. In other words, all the inequalities are active for x when the RHS is replaced with $\beta(x)$. Node v_x may also have any other solution y such that $\beta(y) \leq \beta(x)$, in which case solution x is only feasible when y is. Note that we only need $O(2^k)$ nodes with distinct states to populate a layer instead of $O(2^{2^k})$ to cover all possible states. Figure 3 shows the tightest RHS values associated with each solution for the inequalities of the problem illustrated in Figure 2. The next result formalizes the condition.

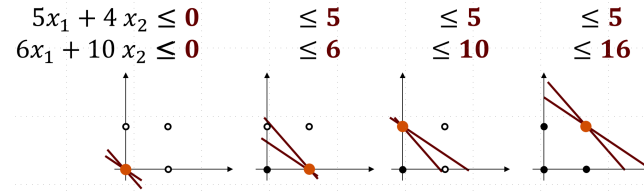


Fig. 3. Smallest (right-hand side) RHS values for which each solution in $\{0,1\}^2$ is feasible for the inequalities with left-hand side (LHS) $5x_1 + 4x_2$ and $6x_1 + 10x_2$.

Theorem 3. *Let v belong to a populated layer. If there is a node \bar{v} satisfying the necessary condition and the dominance elimination, then node \bar{v} is equivalent to v . If no node satisfies the necessary condition, then v has no solution.*

Proof. Let us suppose, for contradiction, that there is an explored node \bar{v} that satisfies both conditions but is not equivalent to v . In such a case, node v has a solution x that \bar{v} does not have. However, a populated layer containing node v would also contain a node v_x corresponding to the lower ends in $\beta(x)$. Node v_x satisfies the necessary condition since v contains x and hence any solution that is always feasible when x is feasible. Therefore, either v_x is the node left after both conditions or else \bar{v} contains all solutions of v_x , thereby contradicting that v has a solution that \bar{v} does not have.

Now let us suppose, for contradiction, that node v has a solution x but no explored node satisfies the necessary condition. That would imply that the layer does not contain a node v_x corresponding to $\beta(x)$, a contradiction. \square

One way to guarantee that a layer is populated is through bottom-up generation of nodes corresponding to the smallest RHS values yielding each solution.

7 Computational Experiments

We evaluated the impact of identifying equivalent search nodes when constructing reduced decision diagrams for integer linear programming problems. The construction of these diagrams mimics the branch-and-bound tree that emerges

from a mathematical optimization solver as it enumerates the solutions of a problem, which we captured through callback functions when a solution is found or a branching decision is about to be made. We use pure 0–1 problems that are small enough to have a reasonable runtime, since enumerating all solutions takes much longer than solving the problem to optimality [19].

For each problem, we defined a gap with respect to the optimal value to limit the enumeration. The gap was chosen as large as possible to either enumerate all solutions or to avoid a considerable solver slowdown, for example from storing search nodes in disk. Nevertheless, we tried to push the gap to the largest possible value since more equivalences can be identified as the solution set gets denser. All problems are either directly obtained or adapted from MIPLIB [11, 12, 44]. When enumerating near-optimal solutions, we add an inequality to limit the objective function value. The order of the selector variables follows the indexing of the decision variables for the corresponding problem. We did not consider the possibility of changing the order of the selector variables, since any such change could have an unrelated effect on the number of branches and runtime.

For each problem, we constructed decision diagrams with preprocessing in the bottom k layers above the terminal node, where $k \in \{3, 6, 9\}$. We use $k = 0$ as the baseline, which is the case in which we cannot always determine if two nodes are equivalent by inspecting the RHS. For $k > 0$, we generated the smallest RHS values for each feasible solution on the remaining k variables. We created marked nodes corresponding to each of such RHS vectors, which are explored when first matched with an unexplored node. Finally, we kept track of the solutions found before fixing all variables to avoid recounting them at the bottom of the diagram.

However, the number of equivalences that can be identified decays significantly as we move further away from the bottom of the decision diagram. Since the number of possible states for the last k levels is $O(2^{2^k})$, it is rather expected that these equivalences will only be identified closer to the bottom — even in the cases that we can significantly reduce the size of the decision diagram. In practice, we did not observe significant gains with a value of k larger than 10.

In our experiments, the code is written in C++ (gcc 4.8.4) using CPLEX Studio 12.8 as a solver and ran in Ubuntu 14.04.4 on a machine with 40 Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz processors and 132 GB of RAM.

Table 1 reports the number of branches, runtime, and solutions found for each problem. We do not report results for `stein9` with $k = 9$ because that problem has exactly 9 variables. Hence, we exclude the results for `stein9` from the geometric mean in all cases, although we note that the number of branches and the runtime for `stein9` reduced as k increased up to 6. For the rest of the problems, we found a reduction of over 27.7% in number of branches and 7.7% in runtimes when comparing the geometric mean of the baseline with the geometric mean for $k = 9$. We observed a consistent reduction in the number of branches for most cases, which is often not offset by the number of corresponding bottom-up states generated: 15 for $k = 3$, 127 for $k = 6$, and 1,023 for $k = 9$. While generating these additional nodes in advance is cheaper than branching, the extra time to check equivalence might explain the lesser impact on runtime.

Table 1. Number of branches and runtime to construct a decision diagram that enumerates near-optimal solutions using the baseline approach and bottom-up preprocessing in the bottom k layers for $k \in \{3, 6, 9\}$ to identify equivalent nodes.

Problem	Variables	Gap	Solutions	Branches					Runtime (s)				
				Baseline	$k = 3$	$k = 6$	$k = 9$	Baseline	$k = 3$	$k = 6$	$k = 9$		
air01	771	5,400	117,997	9,402,967	9,403,025	9,398,697	9,393,475	3,142.1	3,178.8	3,188	3,184.4		
bm23	27	60	2,168	17,931	17,873	17,573	16,054	4.2	4	4.1	4.1		
enigma	100	1	4	3,772	3,772	3,772	3,772	0.4	0.5	0.5	0.5		
ll52lav	1,989	30	37,741	7,576,394	7,576,394	7,576,394	7,576,394	7,976.5	9,274.3	9,360.4	8,036.5		
lp4l	1,086	100	91,672	6,641,504	6,641,504	6,641,504	6,641,504	3,631.6	3,534.3	3,518.3	3,555		
lseu	89	340	659,207	12,934,695	12,935,416	12,887,431	12,662,594	14,223	14,455.3	14,367.8	14,133.8		
misc01b	82	250	600	4,302	4,264	3,963	3,760	0.9	0.9	0.7	0.8		
misc02b	58	1,500	1,168	3,998	3,936	3,564	3,185	0.8	0.7	0.7	0.6		
misc03b	159	1,600	4,320	75,730	75,651	74,309	72,019	14.8	14.6	14.1	13.9		
misc07b	259	400	16,272	805,012	804,545	801,219	798,062	248.6	238.6	234.6	228.5		
p0033	33	2,200	10,746	17,174	16,368	11,783	7,498	2.6	2.5	2	1.5		
p0040	40	7,200	519,216	851,356	645,703	328,415	144,066	138	84.5	42.6	19.1		
p0201	201	600	864,128	27,675,260	27,661,111	27,501,029	27,140,363	9,017.5	8,564.6	8,865	8,418.1		
p0291	291	2	214	685,472	685,472	685,472	685,472	1,174.8	2,124.4	4,926	5,469		
pipex	48	200	12,266	47,420	47,420	47,420	47,420	9	9.1	9	9.1		
sentoy	60	270	67,820	6,587,574	6,504,322	5,919,396	5,219,636	8,052.8	7,743.3	7,054.3	6,732.1		
stein9	9	4	172	101	73	73	—	0.05	0.04	0.04	—		
stein15	15	6	2,809	1,739	1,233	842	841	0.5	0.3	0.3	0.4		
stein27	27	9	367,525	301,488	247,169	114,290	44,497	79.2	56.6	52.9	61.9		
stein45	45	2	795,064	3,970,424	3,955,623	3,833,862	3,575,399	896.9	882.4	1,014	1,973.9		
Geometric mean ³				2.9×10^5	2.8×10^5	2.4×10^5	2.1×10^5	97.0	92.8	91.2	89.5		

³ The geometric mean excludes results for stein9 in all columns.

8 Conclusion

This paper discussed the connection between redundant work in branch-and-bound and the direct construction of reduced decision diagrams. In both cases, such redundancy may manifest as nodes defining equivalent subproblems that are repetitively explored. That connection is particularly stronger if we want to generate a pool of feasible or near-optimal solutions of a discrete optimization problem, which requires substantially more branching than finding an optimal solution. The enumeration of solutions is a relatively unexplored topic, especially in integer linear programming. Nevertheless, alternate solutions are important in practice and generating them in smaller problems is now technically feasible due to the continuous advances in hardware and algorithms. Furthermore, decision diagrams provide a compact representation of solution sets, with which we can more efficiently manipulate to solve the same problem with different objectives.

Our first contribution is a simple but useful extension of prior work on identifying equivalent problems with a single inequality [6, 1, 2], which can be mainly useful for integer linear programs with fractional coefficients.

Our second contribution, which we believe is the most important, is the theoretical distinction between the case of equivalence involving one inequality and multiple inequalities. For problems defined by inequalities with additively separable LHS, we have seen that explored nodes can be uniquely identified by the smallest RHS values yielding the same solutions. When the nodes are explored in depth-first search, we showed that it is possible to isolate a single explored node as potentially equivalent to each unexplored node. This fact alone simplifies considerably the identification of potentially equivalent search nodes.

Our third contribution is a first sufficient condition to confirm the equivalence among such pairs of nodes, which consists of a bottom-up preprocessing technique based on fixing k among the n variables to branch last. Note that we can reasonably expect that equivalences will be more frequent if there are fewer variables left. For $k \ll n$, this would only marginally affect the search behavior and, in fact, our experiments showed a positive impact with small values of k .

For large solution sets, we found some instances in which our approach reduced branching and runtime. If the solution set of the decision diagram is not sufficiently dense, the effectiveness of our method would depend on identifying hidden structure in the problems. One potential example are problems with variables that produce the same effect when assigned, which has been previously exploited with orbital branching [39]: if we leave such variables for last in the decision diagram, we can anticipate that many nodes will have equivalent states.

We believe that there is further room to improve on runtime based on the reduction on the number of branches. Another topic for future work would be identifying simpler sufficient conditions for equivalence, which would allow us to directly construct decision diagrams for much larger problems.

Acknowledgements I would like to thank John Hooker for bringing this topic to my attention and the anonymous reviewers for their detailed feedback.

References

1. Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E., Mayer-Eichberger, V.: A new look at BDDs for pseudo-boolean constraints. *Journal of Artificial Intelligence Research* **45**, 443–480 (2012)
2. Abío, I., Stuckey, P.J.: Encoding linear constraints into SAT. In: *Principles and Practice of Constraint Programming (CP)*, pp. 75–91 (2014)
3. Achterberg, T., Heinz, S., Koch, T.: Counting solutions of integer programs using unrestricted subtree detection. In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pp. 278–282 (2008)
4. Andersen, H., Hadzic, T., Hooker, J., Tiedemann, P.: A constraint store based on multivalued decision diagrams. In: *Principles and Practice of Constraint Programming (CP)*, pp. 118–132 (2007)
5. Balas, E., Jeroslow, R.G.: Canonical cuts on the unit hypercube. *SIAM J. Appl. Math.* **23**, 61–69 (1972)
6. Behle, M.: *Binary Decision Diagrams and Integer Programming*. Ph.D. thesis, Universität des Saarlandes (2007)
7. Bergman, D., Cire, A.A., van Hove, W.J., Hooker, J.N.: Variable ordering for the application of bdds to the maximum independent set problem. In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*. pp. 34–49 (2012)
8. Bergman, D., Cire, A.A., van Hove, W.J., Hooker, J.N.: Discrete optimization with decision diagrams. *INFORMS Journal on Computing* **28**(1), 47–66 (2016)
9. Bergman, D., Cire, A., van Hove, W.J., Hooker, J.: *Decision Diagrams for Optimization*. Springer (2016)
10. Bergman, D., Cire, A.A.: Discrete nonlinear optimization by state-space decompositions. *Management Science* **64**(10), 4700–4720 (2018)
11. Bixby, R.E., Boyd, E.A., Indovina, R.R.: MIPLIB: A test set of mixed integer programming problems. *SIAM News* **25**, 16 (1992)
12. Bixby, R.E., Ceria, S., McZeal, C.M., Savelsbergh, M.W.P.: An updated mixed integer programming library: MIPLIB 3.0. *Optima* **58**, 12–15 (1998)
13. Bollig, B., Wegener, I.: Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers* **45**(9), 993–1002 (1996)
14. Bryant, R.: Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* **C-35**(8), 677–691 (1986)
15. Camm, J.D.: ASP, the art and science of practice: A (very) short course in suboptimization. *INFORMS Journal on Applied Analytics* **44**(4), 428–431 (2014)
16. Chu, G., de la Banda, M.G., Stuckey, P.J.: Exploiting subproblem dominance in constraint programming. *Constraints* **17**(1), 1–38 (2012)
17. Cire, A.A., van Hove, W.J.: Multivalued decision diagrams for sequencing problems. *Operations Research* **61**(6), 1411–1428 (2013)
18. Conforti, M., Cornuéjols, G., Zambelli, G.: *Integer Programming*. Springer (2014)
19. Danna, E., Fenelon, M., Gu, Z., Wunderling, R.: Generating multiple solutions for mixed integer programming problems. In: *Integer Programming and Combinatorial Optimization (IPCO)*, pp. 280–294 (2007)
20. Danna, E., Woodruff, D.L.: How to select a small set of diverse solutions to mixed integer programming problems. *Operations Research Letters* **37**, 255–260 (2009)
21. Davarnia, D., van Hove, W.J.: Outer approximation for integer nonlinear programs via decision diagrams (2018)

22. Ebdendt, R., Gunther, W., Drechsler, R.: An improved branch and bound algorithm for exact bdd minimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **22**(12), 1657–1663 (2003)
23. Fischetti, M., Salvagnin, D.: Pruning moves. *INFORMS Journal on Computing* **22**(1), 108–119 (2010)
24. Fischetti, M., Toth, P.: A new dominance procedure for combinatorial optimization problems. *Operations Research Letters* **7**(4), 181–187 (1988)
25. GAMS Software GmbH: Getting a list of best integer solutions of my MIP (2017), https://support.gams.com/solver:getting_a_list_of_best_integer_solutions_of_my_mip_model, accessed: 2019-11-29
26. Gurobi Optimization, LLC: Finding multiple solutions (2019), https://www.gurobi.com/documentation/8.1/refman/finding_multiple_solutions.html, accessed: 2019-11-29
27. Haus, U.U., Michini, C., Laumanns, M.: Scenario aggregation using binary decision diagrams for stochastic programs with endogenous uncertainty. *CoRR* **abs/1701.04055** (2017)
28. Haus, U.U., Michini, C.: Representations of all solutions of boolean programming problems. In: *International Symposium on Artificial Intelligence and Mathematics (ISAIM)* (2014)
29. Hooker, J.: Decision diagrams and dynamic programming. In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pp. 94–110 (2013)
30. Hosaka, K., Takenaga, Y., Kaneda, T., Yajima, S.: Size of ordered binary decision diagrams representing threshold functions. *Theor. Comput. Sci.* **180**, 47–60 (1997)
31. Ibaraki, T.: The power of dominance relations in branch-and-bound algorithms. *Journal of the Association for Computing Machinery* **24**(2), 264–279 (1977)
32. IBM Corp.: IBM ILOG CPLEX Optimization Studio Getting Started with CPLEX Version 12 Release 8 (2017)
33. IBM Corp.: How to enumerate all solutions (2019), https://www.ibm.com/support/knowledgecenter/SSSA5P_12.9.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/dscr_optim/soln_pool/18_howTo.html, accessed: 2019-11-29
34. Jain, S., Kadioglu, S., Sellmann, M.: Upper bounds on the number of solutions of binary integer programs. In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)* (2010)
35. Kohler, W., Steiglitz, K.: Characterization and theoretical comparison of branch-and-bound algorithms for permutation problems. *Journal of the Association for Computing Machinery* **21**(1), 140–156 (1974)
36. Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. *Econometrica* **28**(3), 497–520 (1960)
37. Lozano, L., Smith, J.C.: A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs. *Mathematical Programming* (2018)
38. Morrison, D., Sewell, E., Jacobson, S.: Solving the pricing problem in a branch-and-price algorithm for graph coloring using zero-suppressed binary decision diagrams. *INFORMS Journal on Computing* **28**(1), 67–82 (2016)
39. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. *Mathematical Programming* **126**, 147–178 (2011)
40. Perez, G., Régin, J.C.: Efficient operations on MDDs for building constraint programming models. In: *International Joint Conference on Artificial Intelligence (IJCAI)* (2015)
41. Petit, T., Trapp, A.C.: Enriching solutions to combinatorial problems via solution engineering. *INFORMS Journal on Computing* **31**(3), 429–444 (2019)

42. Raghunathan, A., Bergman, D., Hooker, J., Serra, T., Kobori, S.: Seamless multi-modal transportation scheduling. CoRR **abs/1807.09676** (2018)
43. Sanner, S., Uther, W., Delgado, K.V.: Approximate dynamic programming with affine ADDs. In: AAMAS (2010)
44. Serra, T., Hooker, J.: Compact representation of near-optimal integer programming solutions. *Mathematical Programming* (2019)
45. Serra, T., Raghunathan, A., Bergman, D., Hooker, J., Kobori, S.: Last-mile scheduling under uncertainty. In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pp. 519–528 (2019)
46. Serra, T., Ramalingam, S.: Empirical bounds on linear regions of deep rectifier networks. CoRR **abs/1810.03370** (2018)
47. Tjandraatmadja, C., van Hoes, W.J.: Target cuts from relaxed decision diagrams. *INFORMS Journal on Computing* **31**(2), 285–301 (2019)
48. Trapp, A.C., Konrad, R.A.: Finding diverse optima and near-optima to binary integer programs. *IIE Transactions* **47**, 1300–1312 (2015)
49. Valiant, L.G.: The complexity of computing the permanent. *Theoretical Computer Science* **8**, 189–201 (1979)
50. Valiant, L.G.: The complexity of enumeration and reliability problems. *SIAM Journal on Computing* **8**(3), 410–421 (1979)
51. Verhaeghe, H., Lecoutre, C., Schaus, P.: Compact-MDD: Efficiently filtering (s)MDD constraints with reversible sparse bit-sets. In: *IJCAI* (2018)
52. Verhaeghe, H., Lecoutre, C., Schaus, P.: Extending compact-diagram to basic smart multi-valued variable diagrams. In: *CPAIOR* (2019)
53. Ye, Z., Say, B., Sanner, S.: Symbolic bucket elimination for piecewise continuous constrained optimization. In: *CPAIOR* (2018)