Constraints (2015) 20:346–361
DOI 10.1007/s10601-015-9193-y

Lagrangian bounds from decision diagrams

David Bergman[1] · Andre A. Cire[2] · Willem-Jan van Hoeve[3]

Published online: 21 April 2015

Abstract Relaxed decision diagrams have recently been used in constraint programming to improve constraint propagation and optimization reasoning. In most applications, however, a decision diagram is compiled with respect to a single combinatorial structure. We propose to expand this representation by incorporating additional constraints in the decision diagram via a Lagrangian relaxation. With this generic approach we can obtain stronger bounds from the same decision diagram, while the associated cost-based filtering allows for further refining the relaxation. Experimental results on the traveling salesman problem with time windows show that the improved Lagrangian bounds can drastically reduce solution times.

Keywords Decision diagrams · Lagrangian relaxation · Constraint propagation

✉ Willem-Jan van Hoeve
vanhoeve@andrew.cmu.edu

David Bergman
david.bergman@business.uconn.edu

Andre A. Cire
acire@utsc.utoronto.ca

[1] School of Business, University of Connecticut, One University Place, Stamford, CT 06901, USA

[2] Department of Management, University of Toronto Scarborough, 1265 Military Trail, Toronto, ON M1C-1A4, Canada

[3] Tepper School of Business, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

1 Introduction

Decision diagrams are compact graphical representations of Boolean functions, originally introduced for applications in circuit design, and widely studied and applied in computer science [1, 12, 21]. More recently, multivalued decision diagrams (MDDs) have been utilized to represent the solution set of discrete optimization problems [3, 4, 16]. In order to address the (typically) exponentially sized representation that the exact diagram yields, Andersen et al. [2] introduced *relaxed decision diagrams* of a given maximum size. Constraint propagation based on relaxed decision diagrams can be much stronger than conventional domain propagation and has been shown to lead to orders of magnitude search space (and time) reduction [7, 13, 18]. Relaxed MDDs have also been studied for obtaining optimization bounds, both for stand-alone discrete optimization problems [8, 9, 11] and in the context of constraint programming [13].

In the context of constraint programming, one typically associates a decision diagram with a specific global constraint that is defined on a subset of variables (its scope). The decision diagram is then (usually) compiled with respect to the combinatorial structure represented by that global constraint. Other constraints that operate on the same variables, or a subset of them, can then be used to subsequently refine and filter the decision diagram. Additionally, if the objective function is defined on the same set of variables, we can use the relaxed decision diagram to obtain an objective function bound, as well as apply cost-based filtering to further refine the decision diagram.

Even when MDD-based constraint propagation reduces the search space, the optimization bound may not be majorly impacted by the associated filtering due to other constraints. This paper aims towards enhancing the inference from other constraints directly into the objective function evaluation of the MDD via a *Lagrangian relaxation* method. This allows for obtaining stronger bounds than the original relaxation, which can in turn help to refine the MDD, and ultimately reduce the search space.

Lagrangian relaxations have previously been incorporated in the context of constraint propagation; Benoist et al. [5] and Khemmoudj et al. [20] provide one of the earliest examples. In almost all cases, however, a Lagrangian relaxation is introduced for a specific global constraint or for a specific problem. The aim of the present study is to provide a generic framework that utilizes the structure of a decision diagram.

The remainder of the paper is organized as follows. First, in Section 2, a brief background on Lagrangian relaxations is provided followed in Section 3 by necessary background on MDD relaxations. Section 4 describes how Lagrangian relaxations can be incorporated in MDD relaxations. An experimental evaluation is presented in Section 5, with a conclusion in Section 6.

2 Preliminaries

Notation. Let $x = (x_1, \ldots, x_n)$ be a set of n decision variables having finite domains $D(x_1), \ldots, D(x_n)$, respectively, with $D = D(x_1) \times D(x_2) \times \cdots \times D(x_n)$. Given an objective function $f : D \to \mathbb{R}$, a *discrete optimization problem* $\mathcal{P} = (f, S, x, D)$ is a problem of the form $\{\min \ f(x) : x \in S \cap D\}$. The goal is to find an assignment of values from D to the variables x within a *feasible region* S that minimizes the objective f. We assume throughout

this paper that there is always at least one feasible solution ($S \cap D \neq \emptyset$). The feasible region S is traditionally described implicitly by a set of constraints, as in Problem (1):

$$\begin{aligned} \min \quad & 5x_1 + 7x_2 + 10x_3 \\ \text{s.t.} \quad & \texttt{alldifferent}(x_1, x_2, x_3) \\ & 7x_1 + 5x_2 + 4x_3 \leq 51 \\ & x_1 + x_2 \leq 6 \\ & x_1, x_2, x_3 \in \{1, 3, 6\}. \end{aligned} \tag{1}$$

The keyword `alldifferent` is a high-level constraint which enforces variables x_1, x_2, x_3 to be pairwise distinct, and appears in many constraint programming models [24]. The problem above will be used as a running example throughout this paper. It has an optimal solution value of 82 attained by the feasible assignment (*optimal solution*) $x = (3, 1, 6)$.

Relaxations. Generic solution methods for discrete optimization problems, such as mathematical programming and constraint programming, typically rely on *relaxations*. A relaxation for $\mathcal{P}$ is an alternative optimization problem $\mathcal{P}_R = (f_R, S_R, x, D_R)$, usually easier to solve than $\mathcal{P}$, where $\forall j,\ D(x_j) \subseteq D_R(x_j)$, and any solution x that is feasible to $\mathcal{P}$ is also feasible to $\mathcal{P}_R$ (i.e. $S \subseteq S_R$) and satisfies $f_R(x) \leq f(x)$. Relaxations are useful for a variety of purposes in optimization models [19], most often for providing bounds on the optimal value. Such bounds can be used both for certifying the quality of a feasible solution, and for reducing the search space in enumerative solution procedures such as branch-and-bound and backtracking search. For instance, a relaxation to Problem (1) may be the linear program

$$\begin{aligned} \min \quad & 5x_1 + 7x_2 + 10x_3 \\ \text{s.t.} \quad & \sum_{i=1}^{3} y_{ij} = 1, \quad j = 1, 2, 3 \\ & \sum_{j=1}^{3} y_{ij} = 1, \quad i = 1, 2, 3 \\ & x_i = 1y_{i1} + 3y_{i2} + 6y_{i3}, \quad i = 1, 2, 3 \\ & 7x_1 + 5x_2 + 4x_3 \leq 51 \\ & x_1 + x_2 \leq 6 \\ & 1 \leq x_1, x_2, x_3 \leq 6 \\ & 0 \leq y_{ij} \leq 1, \quad i, j = 1, 2, 3 \end{aligned} \tag{2}$$

obtained by representing the convex hull of the `alldifferent` constraint in linear form [19]. This linear program can be solved much more efficiently than the original problem and has an optimal value of 77, which in turn is a lower bound on the optimal solution value of Problem (1).

Relaxations also have other uses in optimization besides providing optimization bounds. In particular, solution methods in constraint programming are fundamentally based on a relaxation of the original problem, the *constraint store*. A constraint store accumulates inference from each individual constraint processing, thereby encoding a global structure of the problem that can be shared among constraints. In practice the constraint store is the *domain store*, which is defined by the Cartesian product of variable domains (and

hence solutions in this relaxation are represented *explicitely*). Each constraint receives the current set of variable domains and inference (or *constraint propagation*) is carried out in the form of domain reductions. For instance, in Problem (1) the initial domain store is $D(x_1) \times D(x_2) \times D(x_3) = \{1, 3, 6\}^3$. Propagation on the constraint $x_1 + x_2 \leq 6$ results in the domain inference $x_1, x_2 \neq 6$, which yields a new set of tighter domains $D'(x_1) \times D'(x_2) \times D(x_3) = \{1, 3\}^2 \times \{1, 3, 6\}$. These domains are then passed as input to other constraints, which will process them in turn until a fixed point is reached. The lower bound on the objective function from the domain store relaxation at this point is 22, obtained by assigning 1 to all variables, although more filtering may be possible. If the `alldifferent` is then processed and eliminates 1 and 2 from the domain of x_3, the bound becomes 72 by assigning 1 to both x_1 and x_2, and assigning 6 to x_3.

Lagrangian relaxations. Another well-known relaxation for discrete optimization problems in operations research is the *Lagrangian relaxation*. Lagrangian relaxations were originally introduced for discrete optimization problems $\mathcal{P}$ with S described at least partially by inequality constraints, i.e. $\mathcal{P} = \{\min f(x) : g^i(x) \leq 0, i = 1, \ldots, m, x \in S' \cap D\}$. The Lagrangian relaxation of $\mathcal{P}$ results from moving the inequality constraints to the objective, associating each with a non-negative *penalty* or *Lagrange multiplier* λ_i. For any $\lambda = (\lambda_1, \ldots, \lambda_m) \geq 0$, the Lagrangian relaxation of $\mathcal{P}$ is

$$\mathcal{L}_{\mathcal{P}}(\lambda) = \left\{ \min f(x) + \sum_{i=1}^{m} \lambda_i g^i(x) : x \in S' \cap D \right\}$$

It follows that $\mathcal{L}_{\mathcal{P}}$ is a relaxation of $\mathcal{P}$: the feasible region of $\mathcal{L}_{\mathcal{P}}$ contains that of $\mathcal{P}$ (as $\mathcal{L}_{\mathcal{P}}$ has fewer constraints) and, for any feasible $x \in S$,

$$f(x) + \sum_i \lambda_i \cdot g^i(x) \leq f(x)$$

since $g^i(x) \leq 0$ and $\lambda_i \geq 0$ for all $i = 1, \ldots, m$.

Different values of λ yield distinct relaxations. The problem $\mathcal{L}^*$ of finding the tightest Lagrangian relaxation, i.e.

$$\mathcal{L}^* = \max_{\lambda \geq 0} \mathcal{L}_{\mathcal{P}}(\lambda),$$

is denoted by the *Lagrangian dual*. A number of methods exist for solving the Lagrangian dual, all of which exploit the fact that $\max_{\lambda \geq 0} \mathcal{L}_{\mathcal{P}}(\lambda)$ is a piecewise concave function on λ [14]. Examples include subgradient optimization, cutting-plane algorithms, and bundle methods (the interested reader is referred to a survey of techniques [22]). These methods are *iterative*, in that consecutive problems $\mathcal{L}_{\mathcal{P}}(\lambda_0), \mathcal{L}_{\mathcal{P}}(\lambda_1), \mathcal{L}_{\mathcal{P}}(\lambda_2), \ldots$ are solved until the sequence $\lambda_0, \lambda_1, \lambda_2, \ldots$ converges to a local or global optimal solution to the Lagrangian dual. It is often the case that many iterations may be necessary for convergence, and hence it is typically necessary that each $\mathcal{L}_{\mathcal{P}}(\lambda_i)$ can be solved in a computationally efficient way.

Lagrangian relaxations have frequently been used to decompose models for which some of the constraints are "easy" and others are "hard". As an illustration, a possible Lagrangian relaxation to Problem (1) is

$$\begin{array}{ll} \min & 5x_1 + 7x_2 + 10x_3 + \lambda_1(7x_1 + 5x_2 + 4x_3 - 51) + \lambda_2(x_1 + x_2 - 6) \\ \text{s.t.} & \texttt{alldifferent}(x_1, x_2, x_3) \\ & x_1, x_2, x_3 \in \{1, 3, 6\} \end{array} \tag{3}$$

which corresponds to the well-studied *matching problem* and can be solved efficiently by a number of methods [23]. The optimal bound obtained from Problem (3), after solving

the Lagrangian dual, is 77 for $\lambda_1 = 1$ and $\lambda_2 = 2$. This is the same value obtained from the linear programming relaxation (2), a result which is theoretically expected [14]: If the relaxed inequality constraints define a linear system $Ax \leq b$, then

$$\mathcal{L}^* = \{\min f(x) \ : \ Ax \leq b, \ x \in \text{conv}(S' \cap D)\} \tag{4}$$

where $\text{conv}(X)$ is the convex hull of a set X. In our example, the convex hull of the `alldifferent` constraint in Problem (3) is represented exactly in the linear program, and thus the Lagrangian bound matches that of a linear relaxation.

Recently, Lagrangian methods have been generalized to optimization models described by highly structured languages [15], such as in constraint programming and local search formulations. In that case, the semantics of each constraint is exploited in order to reveal the degree to which a solution satisfies or violates the constraint. For simplicity, in this paper we focus on problems that can be described, at least partially, by inequality constraints noting that many concepts presented here have a natural extension to the more general setting [15].

3 Relaxed multivalued decision diagrams.

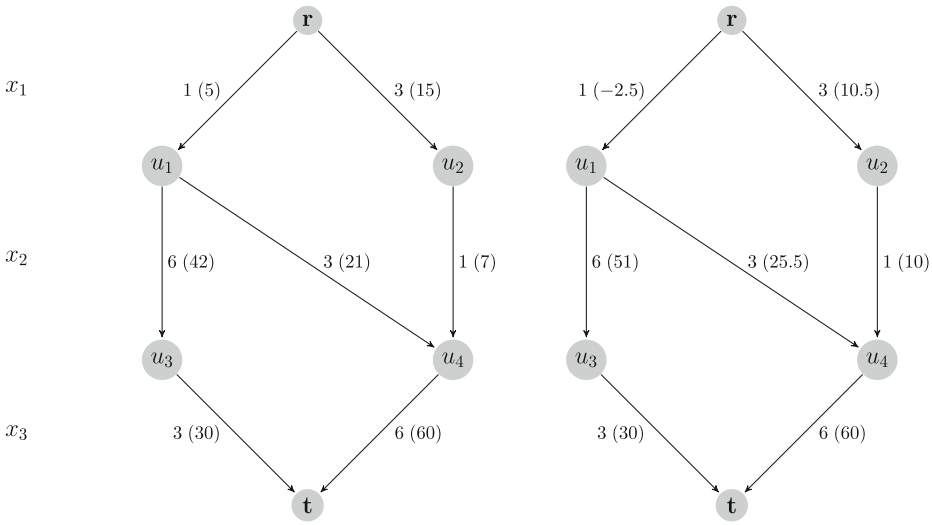
A multivalued decision diagram (MDD) $\mathcal{M}$ for a discrete optimization problem $\mathcal{P}$ is a directed acyclic multigraph whose paths encode a set of solutions to $\mathcal{P}$. Given the n variables of the problem with domains $D(x_1), \ldots, D(x_n)$, the nodes of $\mathcal{M}$ are partitioned into $n+1$ layers $L_1, \ldots, L_{n+1}$, where L_1 and L_{n+1} are defined by single nodes: the *root node* $\mathbf{r}$ and the *terminal* $\mathbf{t}$, respectively. Each arc a in $\mathcal{M}$ is directed from a node in a layer L_j to the consecutive layer L_{j+1} for some j which is specified by $layer(a)$, and has a label $val(a) \in D(x_j)$ that represents a value to be assigned to variable x_j. An arc-specified path $p = (a_1, a_2, \ldots, a_n)$ from $\mathbf{r}$ to $\mathbf{t}$ thereby encodes the solution $x^p = (x_1, \ldots, x_n) = (val(a_1), val(a_2), \ldots, val(a_n))$.

The set of solutions encoded by the paths of an MDD $\mathcal{M}$ is denoted by $Sol(\mathcal{M})$. In the context of optimization, each arc a of $\mathcal{M}$ can be associated with a *cost* $c(a)$. The cost of a path $p = (a_1, \ldots, a_n)$ is given by $c(p) = \sum_{i=1}^{n} c(a_i)$. We denote the function $c(\cdot)$ by the MDD *cost structure*.

The diagrams in Fig. 1 are examples of MDDs that encode solutions for Problem (1). Each arc a has two values associated with it - the first is $val(a)$ and the second, in parentheses, is $c(a)$.

An MDD $\mathcal{M}$ represents a relaxation for $\mathcal{P} = (f, S, x)$ if $\mathcal{M}$ *underapproximates* $\mathcal{P}$; that is, $S \cap D \subseteq Sol(\mathcal{M})$ and $c(p) \leq f(x^p)$ for any path p in $\mathcal{M}$ for which x^p is feasible to $\mathcal{P}$, i.e. $x^p \in S \cap D$. MDDs satisfying these conditions are called *relaxed MDDs*. From this definition, a lower bound on the optimal value of $\mathcal{P}$ is $\min_{p \in \mathcal{M}} c(p)$, which can be obtained in a straightforward way by computing a shortest path from $\mathbf{r}$ to $\mathbf{t}$ with respect to the costs $c(a)$ on arcs. Figure 1a depicts a relaxed MDD for Problem (1), where its shortest path value yields a lower bound of 77 (solution (1, 6, 3)). Notice that this bound is also the same as the one obtained from Problems (2) and (3), although this need not be the case in general.

A relaxed MDD for a problem $\mathcal{P}$ can be compiled either by a top-down algorithm [9] or by an incremental refinement method [17]. The MDD in Fig. 1a was constructed via an incremental refinement algorithm [13]. Compilation techniques rely on a *recursive formulation* of the problem (such as a dynamic programming model) which do not require a linear inequality-based description of the constraints. MDD relaxations are obtained by limiting the *width* of the diagram, i.e. the maximum number of nodes in any layer, according to some input parameter W. For example, in Fig. 1a we have $W = 2$. Larger values for W allow for



(a) Original MDD. (b) Lagrangian MDD.

Fig. 1 Relaxed MDD for Problem (1). For each arc a, the number immediately next to a represent the label $val(a)$ and the number in parenthesis, the cost $c(a)$

a more accurate representation of the problem and, in general, tighter optimization bounds, though at a higher computational construction cost, in terms of both time and memory. Optimization bounds from relaxed MDDs were studied in problems such as set covering [11], maximum independent set [9], and maximum cut [6].

Notice that MDDs, similar to domain store relaxations, are an *extensional* solution encoding of an optimization problem, in that solution values are represented explicitly as opposed to an implicit representation given by other models, for example IP models. Recent work in constraint programming has exploited this property by using relaxed MDDs either as a constraint store or to encode solutions for groups of highly-structured constraints, which may improve the global knowledge about the problem and lead to significant speed ups in search. Applications include systems of `alldifferent` constraints [2], scheduling, [13], and timetabling constraints [6].

4 Lagrangian bounds from MDDs

The underlying concept of the technique presented in this paper is to apply Lagrangian methods to strengthen relaxed MDDs. This strengthening will be reflected both in terms of the optimization bounds provided by the MDD and in the set of solutions encoded by the diagram. Penalties are associated with the constraints that are violated by the solutions of an MDD, which are in turn incorporated into the MDD's cost structure. The general properties of a relaxed MDD are maintained, readily allowing for the application of the techniques described in this paper with previous work that makes use of this data structure (e.g., [6, 9, 13]).

Let $\mathcal{P} = (f, S, x, D)$ be a discrete optimization problem and $\mathcal{M}$ a relaxed MDD for $\mathcal{P}$. Let $g^i(x) = \sum_{j=1}^{n} g^i_j(x_j), i = 1, \ldots, m$ be a set of *m additively separable functions*, a class

that includes, for example, linear functions. Assume that $g^i(x) \leq 0$, $i = 1, \ldots, m$, for any feasible $x \in S \cap D$, but some (or all) of the constraints above may be violated by solutions in $Sol(\mathcal{M})$ that are encoded in $\mathcal{M}$. Our results are based on the following Theorem.

Theorem 1 *For any arc a with label $val(a)$ originating from layer $\ell(a)$ and any $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m) \geq 0$, let*

$$c_\lambda(a) = c(a) + \sum_{i=1}^{m} \lambda_i g^i_{\ell(a)}(val(a)). \tag{5}$$

Then $\mathcal{M}$ with redefined costs c_λ is also a relaxed MDD:

$$\forall \lambda \geq 0, \forall p \in \mathcal{M} : c_\lambda(p) \leq f(x^p). \tag{6}$$

Additionally, the optimal Lagrangian dual on $\mathcal{M}$ never yields a bound worse than the one obtained from the original cost structure:

$$\min_{p \in \mathcal{M}} c(p) \leq \max_{\lambda \geq 0} \min_{p \in \mathcal{M}} c_\lambda(p) \leq \min_{x \in S \cap D} f(x). \tag{7}$$

Proof 1 To show (6), let $x \in S \cap D$ be a feasible solution to $\mathcal{P}$ and take any $\lambda \geq 0$. Since the graphical structure of the MDD $\mathcal{M}$ was not modified, there exists an arc-specified path $p = (a_1, \ldots, a_n)$ from the root $\mathbf{r}$ to the terminal $\mathbf{t}$ such that $x^p = (val(a_1), \ldots, val(a_n)) = x$. Then

$$\begin{aligned} c_\lambda(p) &= \sum_{j=1}^{n} c_\lambda(a_i) = \sum_{j=1}^{n} \left(c(a_i) + \sum_{i=1}^{m} \lambda_i g^i_j(val(a_j)) \right) \\ &= \sum_{i=1}^{n} c(a_i) + \sum_{i=1}^{m} \lambda_i \left(\sum_{j=1}^{n} g^i_j(val(a_i)) \right) \\ &= c(p) + \sum_{i=1}^{m} \lambda_i g^i(x) \\ &\leq f(x^p) \end{aligned} \tag{8}$$

since $c(p) \leq f(x^p)$ by the definition of $\mathcal{M}$ and $\sum_{i=1}^{m} \lambda_i g^i(x) \leq 0$.

To show (7), let $\lambda_0 = (0, \ldots, 0)$ and notice that

$$\min_{p \in \mathcal{M}} c(p) = \min_{p \in \mathcal{M}} c_{\lambda_0}(p) \leq \max_{\lambda \geq 0} \min_{p \in \mathcal{M}} c_\lambda(p).$$

Finally, let p^* be the path in $\mathcal{M}$ that encodes an optimal solution of $\mathcal{P}$ and λ^* the optimal Lagrange multipliers. By inequality (6),

$$\max_{\lambda \geq 0} \min_{p \in \mathcal{M}} c_\lambda(p) = \min_{p \in \mathcal{M}} c_{\lambda^*}(p) \leq c_{\lambda^*}(p^*) \leq f(x^{p^*}) = \min_{x \in S \cap D} f(x),$$

as desired, completing the proof of the inequalities in (7). □

According to Theorem 1, we can incorporate any $\lambda \geq 0$ into the cost structure of $\mathcal{M}$ following rule (5) and the resulting MDD will also be relaxed and may provide a stronger bound than the one obtained with the original costs. Any set of inequalities which are valid for $\mathcal{P}$ can be incorporated into a relaxed MDD $\mathcal{M}$ this way.

Example. Take Problem (1) again and the relaxed MDD $\mathcal{M}$ from Fig. 1a. Consider the following inequality constraints and their associated Lagrange multipliers λ_1 and λ_2:

$$7x_1 + 5x_2 + 4x_3 \leq 51 \quad (\lambda_1) \quad \text{and} \quad x_1 + x_2 \leq 6 \quad (\lambda_2)$$

Using relation (5) from Theorem 1, the arc costs in layer L_1 should take into account the penalties λ_1 and λ_2 over the constraint coefficients that have x_1 in their scope. In particular, for the first layer we will take into account the constants related to the right-hand sides of the inequalities. Thus, for each arc a emanating from L_1, the arc costs should be replaced by $val(a) \times (5+7\lambda_1+\lambda_2) - 51\lambda_1 - 6\lambda_2$. Analogously, arc costs in layers L_2 and L_3 should be replaced by $val(a) \times (7+5\lambda_1+\lambda_2)$ and $val(a) \times (10+4\lambda_1)$, respectively. For instance, $\lambda_1 = 0$ and $\lambda_2 = 1.5$ give rise to the relaxed MDD depicted in Fig. 1b. The shortest path value is now 78.5 (solution (1, 6, 3)), tighter than the bounds provided by (2) and (3). The optimal λ, obtained using a subgradient algorithm, is $\lambda^* = (0, 1.6667)$ which yields a lower bound of 78.6667. Notice it is stronger than the bound provided by the linear program (2), which was 77. Some insights about the quality of the bound are provided in Section 4.2.

4.1 Cost-based filtering with Lagrangian bounds

Given $\mathcal{P} = (f, S, x, D)$, let U^* be an upper bound on the optimal objective value which is identified through a primal heuristic or any other mechanism. *MDD cost-based filtering* consists of removing arcs encoding only suboptimal solutions, i.e. all arcs a for which $c(p) > U^*$ for any path p from $\mathbf{r}$ to $\mathbf{t}$ that traverses a, since then $f(x^p) \geq c(p) > U^*$ for such paths. All arcs satisfying this property can be identified in linear time in the size of the MDD by computing the shortest path from $\mathbf{r}$ to any node and from $\mathbf{t}$ to any node, which can be done within a single top-down and bottom-up pass, respectively. MDD cost-based filtering reduces the size of the MDD, which in turn could be further refined to strengthen the relaxation through incremental refinement algorithms [17]. Having a more accurate relaxed MDD is paramount in branch-and-bound procedures [6] and when enhancing inference in MDD-based constraint programming methods [7, 13].

The Lagrangian cost structure c_λ could be directly used for cost-based MDD filtering as well: If $c_\lambda(p) > U^*$ for all paths p crossing an arc a in $\mathcal{M}$, then the arc a can be removed since, by Theorem 1, $f(x^p) \geq c_\lambda(p) > U^*$. The advantage in this case is that $c_\lambda(p) > c(p)$ may hold for some of the paths p, thereby increasing the number of arcs filtered. For instance, notice in Fig. 1a that the single path going through arc $(u_3, \mathbf{t})$ has a value of 77, while in Fig. 1b that path has a value of 78.5.

We note that the penalties maximizing the Lagrangian dual over a relaxed MDD will still be optimal after cost-based filtering, since the shortest path will not be excluded. Nonetheless, it might be necessary to recompute the optimal Lagrangian dual if the shortest path is perturbed due to incremental refinement or branching, for example.

4.2 Strength of bounds for the linear case

If the objective function and the dualized constraints are all linear, we can obtain insights on the relative strength of the MDD Lagrangian bound in comparison to other linear and/or Lagrangian relaxations of the problem. In this case, the cost structure of a relaxed

MDD often directly represents the contribution of each variable assignment to the objective function (see for example [9, 10]). Namely, if the objective f is given by $f(x) = d_1x_1 + \cdots + d_nx_n$, where $d_1, \ldots, d_n$ are scalars, then the cost structure $c(a) = d_{\ell(a)} \times val(a)$ for all arcs a in $\mathcal{M}$ always yields a valid MDD relaxation, as long as $Sol(\mathcal{M})$ contains all feasible solutions to the problem. Note also that $c(p) = f(x^p)$ for any path p from $\mathbf{r}$ to $\mathbf{t}$ in the MDD.

To state our result, consider an MDD $\mathcal{M}$ and let $\texttt{shortpath\_poly}(\mathcal{M})$ be the *shortest path polytope* of $\mathcal{M}$ [4]. The shortest path polytope of $\mathcal{M}$ is a linear program that models a shortest path problem from $\mathbf{r}$ to $\mathbf{t}$ defined over the graphical structure of $\mathcal{M}$, considering the MDD arc costs directly as transition costs. It has been introduced for cut generation and for enhancing branching in the context of mathematical programming [4].

Given $\mathcal{M}$, the $\texttt{shortpath\_poly}(\mathcal{M})$ is formally described in (9). Let $\mathcal{A}$ and $\mathcal{U}$ be the set of arcs and nodes, respectively, in $\mathcal{M}$, with $\mathcal{A}_j$ denoting the set of arcs emanating from nodes in layer j. Additionally, let $\delta^+(u) = \{a : a = (u, v) \text{ for some } v \in L_{layer(u)+1}\}$ and $\delta^-(u) = \{a : a = (v, u) \text{ for some } v \in L_{layer(u)-1}\}$. The $\texttt{shortpath\_poly}(\mathcal{M})$ can be perceived as a network model that assigns a variable f_a to each $a \in \mathcal{A}$, ensures that the number of units entering the root and exiting the terminal both equals 1, requires that the number of units directed into any other node equals the number of units directed out of a node, and associates the value x_j as a linear combination of the units on the arcs in layer L_j, scaled by the associated $val(.)$ labels.

$$
\begin{aligned}
&\sum_{a \in \delta^+(\mathbf{r})} f_a = 1 \\
&\sum_{a \in \delta^-(\mathbf{t})} f_a = 1 \\
&\sum_{a \in \delta^+(u)} f_a - \sum_{a \in \delta^-(u)} f_a = 0, && u \in \bigcup_{j=2}^{n} L_j \\
&x_j = \sum_{u \in L_j} \sum_{a \in \delta^+(u)} val(a) f_a, && j = 1, \ldots, n \\
&0 \le f_a \le 1, && a \in \mathcal{A}
\end{aligned}
\tag{9}
$$

Theorem 2 *Let $\mathcal{P} = (f, S, x, D)$ be a discrete optimization problem with a linear objective function f and let $\mathcal{M}$ be a relaxed MDD with cost structure $c(\cdot)$ directly encoding f. Assume the linear system $Ax \le b$ must hold for all $x \in S \cap D$. If we associate penalties λ with the inequalities $Ax \le b$ and obtain c_λ according to (5), then*

$$
\max_{\lambda \ge 0} \min_{p \in \mathcal{M}} c_\lambda(p) = \left\{ \begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax \le b \\ & x \in \texttt{shortpath\_poly}(\mathcal{M}) \end{array} \right\}.
$$

and the optimal duals associated with the constraints $Ax \le b$ of the right-hand side linear program correspond to λ values that maximize the MDD Lagrangian dual.

Proof 2 We have:

$$\begin{aligned}
\max_{\lambda \geq 0} \min_{p \in \mathcal{M}} c_\lambda(p) &= \max_{\lambda \geq 0} \left\{ \begin{array}{ll} \min & c(p) + \lambda(Ax^p - b) \\ & p \in \mathcal{M} \end{array} \right\} && \text{[by (8)]} \\
&= \max_{\lambda \geq 0} \left\{ \begin{array}{ll} \min & f(x) + \lambda(Ax - b) \\ & x \in Sol(\mathcal{M}) \end{array} \right\} && \text{[since } c(p) = f(x^p)\text{]} \\
&= \max_{\lambda \geq 0} \left\{ \begin{array}{ll} \min & f(x) + \lambda(Ax - b) \\ & x \in \texttt{shortpath_poly}(\mathcal{M}) \end{array} \right\} && \text{[By Behle[4]]} \\
&= \left\{ \begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax \leq b \\ & x \in \texttt{shortpath_poly}(\mathcal{M}) \end{array} \right\},
\end{aligned}$$

where the last equality follows from strong Lagrangian duality for linear programs (see, e.g., Hooker [19]).

Also from strong Lagrangian duality, the duals associated with the constraints $Ax \leq b$ of the last linear program define an optimal solution to the previous program, $\max_{\lambda \geq 0}\{f(x) + \lambda(Ax - b) : x \in \texttt{shortpath\_poly}(\mathcal{M})\}$. By Behle [4], the extreme points of the projection of $\texttt{shortpath\_poly}(\mathcal{M})$ over the variables x (which are encoded by $\mathcal{M}$) correspond exactly to $Sol(\mathcal{M})$. Thus, for any $\lambda \geq 0$,

$$\left\{ \begin{array}{ll} \min & f(x) + \lambda(Ax - b) \\ & x \in \texttt{shortpath\_poly}(\mathcal{M}) \end{array} \right\} = \left\{ \begin{array}{ll} \min & f(x) + \lambda(Ax - b) \\ & x \in Sol(\mathcal{M}) \end{array} \right\}$$

Hence, since there is an one-to-one correspondence between the paths $p \in \mathcal{M}$ and the set $Sol(\mathcal{M})$, the optimal duals for $Ax \leq b$ in the last linear program also define an optimal set of λ for the MDD Lagrangian dual problem. □

Thus, the optimal Lagrangian bound can be equivalently obtained by "linearizing" the MDD through a minimum-cost network flow formulation, adding the dualized constraints back, and solving the linear program with the original objective function. The quality of the bound in comparison to other linear programming relaxations therefore depends on whether $\texttt{shortpath\_poly}(\mathcal{M})$ is tighter than the alternative linear formulation of the other constraints of the problem (when projecting onto the same variable space). Indeed, this was the case for relaxation (2) and the MDD bound in Section 4: The relaxed MDD from Fig. 1a was built considering the combinatorial structure of *all* constraints of $\mathcal{P}$. In particular, $\texttt{alldifferent}(x_1, x_2, x_3)$ and $7x_1 + 5x_2 + 4x_3 \leq 51$ are already satisfied by all paths in $\mathcal{M}$, which explains why the MDD Lagrangian bound was strictly better than the one from relaxation (2).

This property could be particularly exploited in the context of constraint programming. Research in relaxed MDDs has focused on encoding groups of highly structured constraints, such as a system of `alldifferent` constraints [2], clique constraints [9], and the `sequence` constraint [7]. The corresponding relaxed MDDs often provided a stronger bound than strengthened linear programming relaxations of these problems, due in part to the fact that the linear programming relaxations do not capture the combinatorial nature of the constraints as well as the relaxed MDDs do.

Some final observations are in order. We note that an equivalent lower bound of 78.6667 or better for Problem (1) could be obtained by incorporating the constraint $7x_1+5x_2+4x_3 \leq 51$ into the Lagrangian subproblem of the relaxation (3), though the resulting subproblem is not necessarily polynomially solvable (while the complexity of the MDD approach depends on the input maximum width). In addition, we observed in computational experiments that solving the Lagrangian dual to optimality was often much faster than solving the linear

program with shortpath_poly($\mathcal{M}$) and the added dualized constraints. This is due to the fact that optimizing a linear function over an MDD is equivalent to a shortest path computation in a directed acyclic graph, which takes time linear in the size of the MDD.

5 Computational experiments

We embedded the Lagrangian techniques Section 4 in the MDD-based sequencing constraint presented in Cire and van Hoeve [13]. That work introduced a propagation mechanism for disjunctive scheduling based on MDDs, which was incorporated into the state-of-the-art constraint-based scheduler ILOG CP Optimizer as a new disjunctive global constraint. The MDD is used both as a means for providing optimization bounds and for inference, in particular by deducing precedence relations between tasks. It was shown that MDD-based propagation for disjunctive scheduling can greatly improve the time to solve single-machine problems to optimality (by several orders of magnitude in many cases) [13].

The goal of this section is to provide a case study analyzing the performance of these techniques when the MDDs are augmented with the Lagrangian methods presented in Section 4. All tests were performed with ILOG CP Optimizer 12.6 on an Intel Xeon E5346 with 32GB of RAM. The CP Optimizer parameters were modified to consider a time-limit of 1 hour, one thread, depth-first search with default variable/value ordering, and extended filtering. The source code has been made available for download.[1]

As a test problem we selected the *traveling salesman problem with time windows* (TSPTW): Given n cities with travel times t_{ij} given between each pair of cities i, j, the problem asks for the tour through all cities in which each city i is visited within a time window $[r_i, d_i]$ and the total time is minimized. The TSPTW was formulated as the following constraint programming model in ILOG CP Optimizer (some specific solver constructs were simplified for clarity):

$$\begin{aligned}
\min\ & z \\
\text{s.t.}\ & \texttt{noOverlap}(\{s_1, \ldots, s_n\} \mid \{t_{ij} : \forall i, j\}) && (10)\\
& z = \sum_{i=1}^{n} t_{i,\texttt{next}(s_i)} && (11)\\
& s_i : \texttt{intervalVar}(r_i, d_i), \quad i = 1, \ldots, n && (12)\\
& \texttt{disjunctiveMDD}(\{s_1, \ldots, s_n\}, \{x_1, \ldots, x_n\}, z \mid \{t_{ij} : \forall i, j\}, W) && (13)\\
& \texttt{alldifferent}(x_1, \ldots, x_n) && (14)\\
& x_1, \ldots, x_n \in \{1, \ldots, n\} && (15)
\end{aligned}$$

In the model above, constraints (10) to (12) represent the typical CP Optimizer TSPTW formulation, while constraints (13) to (15) add the special MDD constraints. The model relies on *interval variables* s_i, each representing the time a city i is visited. The global constraint (10) enforces that cities are visited in sequence, and constraint (11) models the objective function using the sum of *element* constraints $t_{i,\texttt{next}(s_i)}$. In particular, $\texttt{next}(s_i)$ is a CP Optimizer construct that evaluates to the index of the city immediately succeeding city s_i in the sequence defined by noOverlap. The constraint (12) defines the interval variables and enforces cities to be visited within their specified time window. The disjunctive MDD

[1] http://www.andrew.cmu.edu/user/vanhoeve/mdd/

Cities

City	Time window ($[r_i, d_i]$)
1	[2,17]
2	[0,11]
3	[1,10]

Travel Times (t_{ij})

	1	2	3
1	-	6	5
2	7	-	4
3	6	4	-

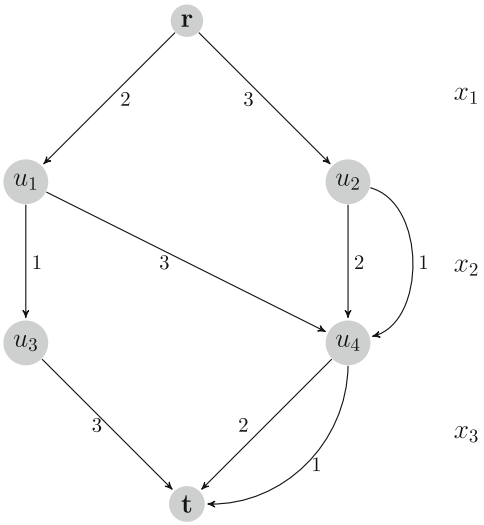


Fig. 2 Example of a relaxed MDD for a TSPTW instance

global constraint (13) is semantically equivalent to a `noOverlap`, but also receives as input the objective variable z, the maximum allowed width W, and variables x, where each x_i represents the i-th city to be visited in the sequence. Constraint (14) is redundant and added to enhance propagation and constraints (15) define the domain of variables x.

The `disjunctiveMDD` maintains a relaxed MDD of maximum width W defined over variables x. Each layer L_i is associated with variable x_i, $i = 1, \ldots, n$, and hence an arc at layer L_i assigns the i-th city to be visited. An example of a relaxed MDD for a 3-city TSPTW instance is presented in Fig. 2, with an optimal tour of value 10 given by the sequence (2, 3, 1). The MDD is compiled and filtered according to the same procedure in the previous study [13]. In particular, let $\delta^-(u)$ be the set of incoming arcs at a node u and let $src(a)$ be the source node of an arc a. The lower bound on the total travel time in the relaxed MDD is given by $\min_{a \in \delta^-(\mathbf{t})} V(a)$, where

$$V(a) = \begin{cases} 0, & \text{if } src(a) = \mathbf{r}, \\ \min_{a' \in \delta^-(src(a)), a' \neq a} (V(a') + t_{val(a'),val(a)}), & \text{otherwise.} \end{cases}$$

In Fig. 2, the lower bound on the total travel time is 8, given by the sequence (2, 3, 2). Bounds on the objective function and the projection of the arcs onto the domains of variables x are passed to CP Optimizer. In addition, the relaxed MDD is also used to deduce precedence relations that are communicated directly to the solver precedence graph. For instance, one can deduce from the relaxed MDD in Fig. 2 that city 1 can never be the first in the sequence.

Similar to the example above, often the shortest path in a relaxed MDD violates the constraint that each city must be visited exactly once. This happens since the width required to represent an `alldifferent` constraint over the x space is $O(2^n)$ [13]. In our MDD Lagrangian scheme we will penalize the violation of this constraint by associating a Lagrange multiplier λ_i to the condition $\sum_{j=1}^{n}(x_j = i) = 1$ for each $i = 1, \ldots, n$. Notice that λ is not restricted in sign in this case. In view of rule (5) from Theorem 1, the new lower

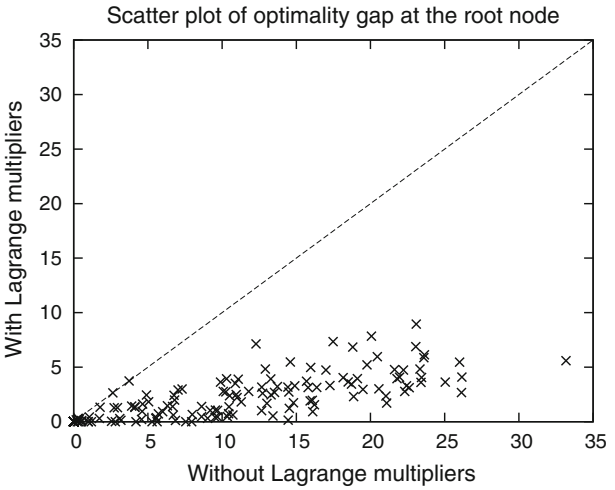


Fig. 3 Scatter plot comparing the optimality gaps with and without the Lagrangian

bound can be computed as $\min_{a \in \delta^-(\mathbf{t})} V_\lambda(a)$, where

$$V_\lambda(a) = \begin{cases} \lambda_{val(a)} - \sum_{i=1}^{n} \lambda_i, & \text{if } src(a) = \mathbf{r}, \\ \lambda_{val(a)} + \min_{a' \in \delta^-(src(a)), a' \neq a} (V(a') + t_{val(a'),val(a)}), & \text{otherwise.} \end{cases}$$

The case $src(a) = \mathbf{r}$ accounts for the constant 1 in the right-hand side of each equality. Also, since the dualized constraints are equalities, paths that do not violate the tour constraint will have the same cost as the original objective function. For example, in Fig. 2 the shortest path for $\lambda_1 = -3$, $\lambda_2 = 2$, and $\lambda_3 = 0$ is (2, 3, 1) with a value of 10, which proves its optimality.

The computational experiments reported here consider the same 230 TSPTW instances that were tested in Cire and van Hoeve [13] (the sets `Dumas` and `GendreauDumasExtended`). We evaluate two versions: the original code [13] and one where the cost structure of the relaxed MDD $\mathcal{M}$ was modified only once at the root node of the backtrack search tree, and re-used throughout the search. The optimal Lagrange multipliers were computed using the *Kelly-Cheney-Goldstein* method [22]. This is an iterative cutting-plane approach where existing multipliers are accumulated in a "bundle" and the next iterate is obtained from a linear program formed from the elements of this bundle (in this case solved using ILOG CPLEX 12.6). Although this technique typically has a slow convergence rate, the technique does not require any parameter settings which is ideal for computational evaluation. Finally, the width of the relaxed MDD was fixed to 128.

Figure 3 compares the optimality gap of the MDD relaxation with and without the Lagrangian at the root node of the backtracking tree for all tested instances. The optimality gap here is computed as $100 * (b - r)/b$, where r is the bound at the root node and b is the best solution found for that instance by either of the methods within the time limit. This figure shows that the optimality gaps can improve substantially when Lagrangian multipliers are incorporated for this problem class, which are up to 7 times better than the bounds obtained originally.

The difference in the obtained bounds has a significant impact on total solution times as well. Figure 4 depicts performance plots, comparing the number of solved instances versus time (in seconds in log-scale). Figure 5 shows a scatter plot, where for each instance the

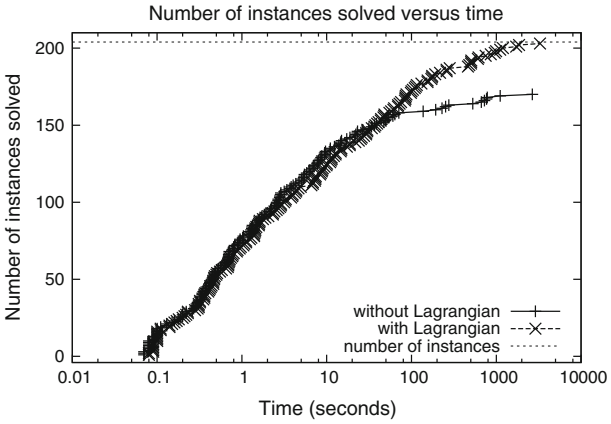


Fig. 4 Performance plot comparing the MDD relaxation with and without the Lagrangian. Times are on a log-log scale

time to solve with and without the Lagrangian method is reported (again in seconds in log-scale); instances below the diagonal are solved faster using the Lagrangian method. For the easier instances almost no performance gains are realized, but as the solution time for the instance grows the enhanced model utilizing Lagrange multipliers prevails. In particular, adding the Lagrangian relaxation allows for 203 of the instances to be solved as opposed to the previous number of 170, within the 1 hour time limit.

Figure 6 depicts, for each instance, how much time the Lagrangian scheme spent on solving the Lagrangian dual using the Kelly-Cheney-Goldstein method (black area) and the total solution time (gray area). Instances are sorted in descending order of total solution times, and points for which the total time was less than one second were discarded. In general, the impact on the Lagrangian dual on the total time varies, though in many instances

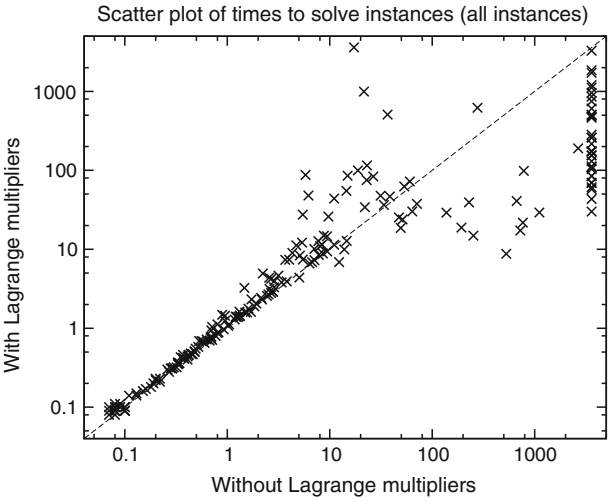


Fig. 5 Scatter plot comparing the MDD relaxation with and without the Lagrangian. Times are on a log-log scale

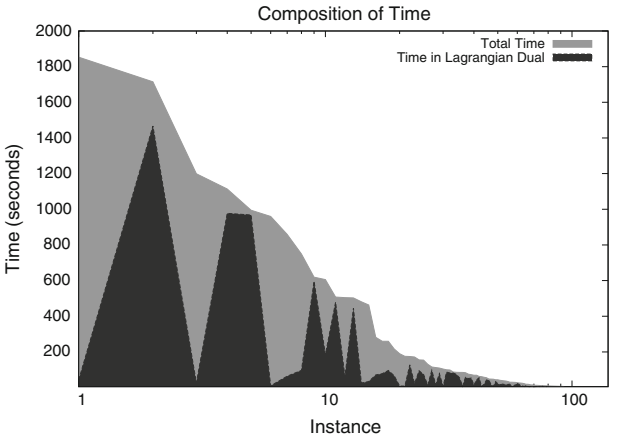


Fig. 6 Time composition of the Lagrangian method

with large times it dominated total solution time, as can be verified in Fig. 6. We note that in many cases for which the Lagrangian relaxation increased the solving time, the relatively long solving time of Kelly-Cheney-Goldstein method was the cause. We expect that a different (subgradient) optimization method may yield better results in those cases. However, even using the slow to converge Kelly-Cheney-Goldstein method can have a substantially positive effect on the overall solution time.

6 Conclusion

In this paper we introduced a generic approach for improving bounds from relaxed decision diagrams by representing the impact of side constraints via a Lagrangian relaxation. This allows for improving optimization reasoning in the context of constraint programming with decision diagrams. The experimental evaluation on a standard benchmark set of the TSP with time windows has demonstrated that incorporating Lagrangian methods to decision diagram-based optimization can lead to substantial savings in computation time.

References

1. Akers, S.B. (1978). Binary decision diagrams. *IEEE Transactions on Computers*, *C-27*, 509–516.
2. Andersen, H.R., Hadzic, T., Hooker, J.N., & Tiedemann, P. (2007). A constraint store based on multivalued decision diagrams. In *Proceedings of the 13th international conference on Principles and practice of constraint programming* (pp. 118–132). Springer-Verlag, Berlin, Heidelberg.
3. Becker, B., Behle, M., Eisenbrand, F., & 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). Lecture Notes in Computer Science* (Vol. 3503, pp. 452–463). Springer.
4. Behle, M. (2007). *Binary decision diagrams and integer programming*. Ph.D. thesis, Max Planck Institute for Computer Science.
5. Benoist, T., Laburthe, F., & Rottembourg, B. (2001). Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems. In *Proceedings of the 3rd international workshop on Integration of AI and OR techniques in constraint programming for combinatorial optimization problems (CPAIOR)*.

6. Bergman, D. (2013). *New techniques for discrete optimization*. Ph.D. thesis, Tepper School of Business, Carnegie mellon university.
7. Bergman, D., Cire, A.A., & van Hoeve, W.J. (2014). Mdd propagation for sequence constraints. *Journal of Artificial Intelligence Research (JAIR)*, *50*, 697–722.
8. Bergman, D., Cire, A.A., van Hoeve, W.J., & Hooker, J.N. (2012). Variable ordering for the application of bdds to the maximum independent set problem. In *Proceedings of the 9th international conference on integration of AI and OR techniques in Constraint programming for combinatorial optimization problems* (pp. 34–49). Springer-Verlag, Berlin, Heidelberg.
9. Bergman, D., Cire, A.A., van Hoeve, W.J., & Hooker, J.N. (2014). Optimization bounds from binary decision diagrams. *INFORMS Journal on Computing*, *26*(2), 253–268.
10. Bergman, D., Cire, A., van Hoeve, W.J., & Yunes, T. (2014). Bdd-based heuristics for binary optimization. *Journal of Heuristics*, *20*(2), 211–234.
11. Bergman, D., van Hoeve, W.J., & Hooker, J. (2011). Manipulating MDD relaxations for combinatorial optimization. In T. Achterberg & J. Beck (Eds.), *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems, Lecture notes in computer science* (Vol. 6697, pp. 20–35). Springer Berlin / Heidelberg.
12. Bryant, R.E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, *C-35*, 677–691.
13. Cire, A.A., & van Hoeve, W.J. (2013). Multivalued decision diagrams for sequencing problems. *Operations Research*, *61*(6), 1411–1428.
14. Fisher, M.L. (2004). The lagrangian relaxation method for solving integer programming problems. *Management Science*, *50*(12), 1861–1871. Supplement.
15. Fontaine, D., Michel, L., & Van Hentenryck, P. (2014). Constraint-based lagrangian relaxation. In B. O'Sullivan (Ed.), *Principles and practice of constraint programming, Lecture notes in computer science* (Vol. 8656, pp. 324–339). Springer International Publishing.
16. Hadzic, T., & Hooker, J. (2006). *Postoptimality analysis for integer programming using binary decision diagrams*. Tech. rep., Carnegie Mellon University.
17. Hadzic, T., Hooker, J.N., O'Sullivan, B., & Tiedemann, P. (2008). Approximate compilation of constraints into multivalued decision diagrams. In *Proceedings of the 14th international conference on principles and practice of constraint programming* (pp. 448–462). Springer-Verlag, Berlin.
18. Hoda, S., van Hoeve, W.J., & Hooker, J.N. (2010). A systematic approach to MDD-based constraint programming. In *Proceedings of constraint programming* (Vol. 6308, pp. 266–280). LNCS, Springer.
19. Hooker, J.N. (2012). *Integrated methods for optimization (International Series in Operations Research & Management Science)*, 2nd Edn. Inc., Secaucus, Springer-Verlag New York.
20. Khemmoudj, M., Bennaceur, H., & Nagih, A. (2005). Combining arc-consistency and dual lagrangean relaxation for filtering csps. In R. Barták & M. Milano (Eds.), *Integration of AI and OR techniques in Constraint programming for combinatorial optimization problems, Lecture notes in computer science* (Vol. 3524, pp. 258–272). Springer Berlin Heidelberg.
21. Lee, C.Y. (1959). Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal*, *38*, 985–999.
22. Lemaréchal, C. (2001). Lagrangian relaxation. In M. Jünger & D. Naddef (Eds.), *Computational combinatorial optimization, Lecture notes in computer science* (Vol. 2241, pp. 112–156). Springer Berlin Heidelberg.
23. Papadimitriou, C.H., & Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River.
24. Rossi, F., van Beek, P., & Walsh, T. (eds.) (2006). *Handbook of constraint programming*. Elsevier.