Decision Diagrams and Dynamic Programming*

J. N. Hooker

Carnegie Mellon University, USA
jh38@andrew.cmu.edu

Abstract. Binary and multivalued decision diagrams are closely related to dynamic programming (DP) but differ in some important ways. This paper makes the relationship more precise by interpreting the DP state transition graph as a weighted decision diagram and incorporating the state-dependent costs of DP into the theory of decision diagrams. It generalizes a well-known uniqueness theorem by showing that, for a given optimization problem and variable ordering, there is a unique reduced weighted decision diagram with "canonical" edge costs. This can lead to simplification of DP models by transforming the costs to canonical costs and reducing the diagram, as illustrated by a standard inventory management problem. The paper then extends the relationship between decision diagrams and DP by introducing the concept of nonserial decision diagrams as a counterpart of nonserial dynamic programming.

1 Introduction

Binary and multivalued decision diagrams have long been used for circuit design and verification, but they are also relevant to optimization. A reduced decision diagram can be viewed as a search tree for an optimization problem in which isomorphic subtrees are superimposed, thus removing redundancy.

Dynamic programming (DP) is based on a similar idea. In fact, the state transition graph for a discrete DP can be viewed as a decision diagram, albeit perhaps one in which not all redundancy has been removed. Conversely, the reduced decision diagram for a given problem tends to be more compact when the problem is suitable for solution by DP. This indicates that there may be benefit in clarifying the connection between decision diagrams and DP. In particular, it may be possible to simplify a DP model by regarding its transition graph as a decision graph and reducing it to remove all redundancy.

However, decision diagrams differ from DP in significant ways. Nodes of the DP state transition graph are associated with state variables, whereas there are no explicit state variables in a decision diagram; only decision variables. Furthermore, arcs of a state transition graph are often labeled with costs, and this is not true of a decision diagram. A decision diagram can be given arc costs when the objective function is separable, but costs in a DP transition graph are more complicated because they are *state dependent*: they depend on state variables as well as decision variables.

* Partial support from NSF grant CMMI-1130012 and AFOSR grant FA-95501110180.

Nonetheless, we show that the elementary theory of decision diagrams can be extended to incorporate state-dependent costs and therefore establish a close parallel with DP. We define a *weighted* decision diagram to be a decision diagram with arbitrary arc costs. Unfortunately, differing arc costs can prevent reduction of a weighted diagram even when the unweighted diagram would reduce. However, we show that costs can often be rearranged on the diagram, without changing the objective function, so as to allow reduction. In fact, we define a unique *canonical* set of arc costs for a given objective function and generalize a well-known uniqueness result for reduced decision diagrams. We show that for a given optimization problem and variable ordering, there is a unique reduced weighted decision diagram with canonical costs that represents the problem.

This opens the possibility of simplifying a DP formulation by converting the transition costs to canonical costs and reducing the state transition diagram that results. In fact, we show this maneuver results in a substantial simplification even for a standard DP formulation of production and inventory management that has appeared in textbooks for decades.

We conclude by extending weighted decision diagram to *nonserial decision diagrams* by exploiting an analogy with nonserial DP.

2 Previous Work

Binary decision diagrams were introduced by [1, 19, 31]. In recent years they have been applied to optimization, initially for cut generation in integer programming [9, 11], post-optimality analysis [25, 26], and 0-1 vertex and facet enumeration [10]. Relaxed decision diagrams were introduced in [3] and further applied in [21, 27, 28] as an alternative to the domain store in constraint programming, and they were used in [14, 15] to obtain bounds for optimization problems. Introductions to decision diagrams can be found in [2, 18].

Dynamic programming is credited to Bellman [12, 13]. A good introductory text is [24], and a more advanced treatment [17]. Nonserial dynamic programming was introduced by [16]. Essentially the same idea has surfaced in a number of contexts, including Bayesian networks [30], belief logics [33, 34], pseudoboolean optimization [22], location theory [20], k-trees [4, 5], and bucket elimination [23].

The identification of equivalent subproblems is known as *caching* in the knowledge representation literature, where it has received a good deal of attention (e.g., [6–8, 29]). However, apparently none of this work deals with state-dependent costs, which are a unique and essential feature of DP, as it is understood in the operations research community.

3 Decision Diagrams

For our purposes, decision diagrams can be viewed as representing the feasible set S of an optimization problem

$$\min_x \{ f(x) \mid x \in S \} \tag{1}$$

Table 1. (a) A small set covering problem. The dots indicate which elements belong to each set i. (b) A nonseparable cost function for the problem. Values are shown only for feasible x.

(a)

	Set i			
	1	2	3	4
A	•	•		
B	•		•	•
C		•	•	
D		•		•

(b)

x	$f(x)$
(0,1,0,1)	6
(0,1,1,0)	7
(0,1,1,1)	8
(1,0,1,1)	5
(1,1,0,0)	6
(1,1,0,1)	8
(1,1,1,0)	7
(1,1,1,1)	9

where $f(x)$ is the objective function, and x a tuple $(x_1, \ldots, x_n)$ of discrete variables with finite domains $D_1, \ldots, D_n$, respectively.

An *ordered decision diagram* is a directed, acyclic graph $G = (N, A)$ whose node set N is partitioned into n *layers* $1, \ldots, n$ corresponding to the variables $x_1, \ldots, x_n$, plus a *terminal* layer (layer $n + 1$). Layer 1 contains only a root node r, and the terminal layer contains nodes 0 and 1. For each node u in layer $i \in \{1, \ldots, n\}$ and each value $d_i \in D_i$, there is a directed arc $a(u, d_i)$ in A from u to a node $u(d_i)$ in layer $i+1$, which represents setting $x_i = d_i$. Each path from r to 1 represents a feasible solution of S, and each path from r to 0 represents an infeasible solution. For our purposes, it is convenient to omit the paths to 0 and focus on the feasible solutions.

As an example, consider a set covering problem in which there are four sets as indicated in Table 1(a), collectively containing the elements A, B, C, D. The problem is to select a minimum-cost *cover*, which is a subcollection of sets whose union is $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$. Let binary variable x_i be 1 when set i is selected, and let $f(x)$ be the cost of subcollection x. The decision diagram in Fig. 1(a) represents the feasible set S. The 9 paths from r to 1 represent the 9 covers.

A decision diagram is a *binary decision diagram* if each domain D_i contains two values, as in the example of Fig. 1. It is a *multivalued decision diagram* if at least one D_i contains three or more values. Decision diagrams can be defined to contain *long arcs* that skip one or more layers, but to simplify notation, we suppose without loss of generality that G contains no long arcs.

A decision diagram is *reduced* when it is a minimal representation of S. To make this more precise, let $G_{uu'}$ be the subgraph of G induced by the set of nodes on paths from u to u'. Subgraphs $G_{uu'}$ and $G_{vv'}$ are *equivalent* when they are isomorphic, corresponding arcs have the same labels, and u, v belong to the same layer. A decision diagram is reduced if it contains no equivalent subgraphs. It is a standard result [19, 35] that there is a unique reduced diagram for any given S and variable order.

The reduced decision diagram can be obtained from a branching tree using a simple procedure. Supposing that the tree contains only the feasible leaf nodes,

we first superimpose all the leaf nodes to obtain terminal node 1, and then continue to superimpose equivalent subgraphs until none remain. For example, the branching tree for the set covering problem of Table 1(a) appears in Fig. 2 (ignore the arc labels at the bottom). The tree can be transformed in this manner to the reduced decision diagram of Fig. 1(a).

4 Weighted Decision Diagrams

Given an optimization problem (1), we would like to assign costs to the arcs of a decision diagram to represent the objective function. We will refer to a decision diagram with arc costs as a *weighted* decision diagram. Such a diagram represents (1) if the paths from r to 1 represent precisely the feasible solutions of (1), and length of each path x is $f(x)$. The optimal value of (1) is therefore the shortest path length from r to 1. We will say that two weighted decision diagrams are *isomorphic* if they yield the same unweighted diagram when arc costs are removed.

The assignment of costs to arcs is most straightforward when the objective function is separable. If $f(x) = \sum_i f_i(x_i)$, we simply assign cost $f_i(d_i)$ to each arc $a(u, d_i)$ leaving layer i in the reduced decision diagram representing S. For example, if $f(x) = 3x_1 + 5x_2 + 4x_3 + 6x_4$, the arc costs are as shown in Fig. 1(b), and the shortest path is $x = (0, 1, 0, 1)$ with cost 11.

Arc costs can also be assigned when the objection function is nonseparable. In fact, we will show that the problem is represented by a unique reduced weighted decision diagram, provided arc costs are assigned in a canonical way.

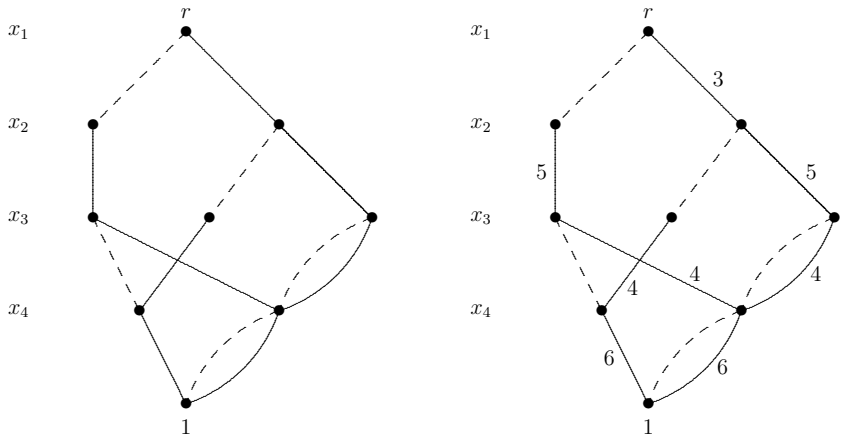


Fig. 1. (a) Decision diagram for the set covering problem in Table 1(a). Dashed arcs correspond to setting $x_i = 0$, and solid arcs to setting $x_i = 1$. (b) Decision diagram showing arc costs for a separable objective function. Unlabeled arcs have zero cost.

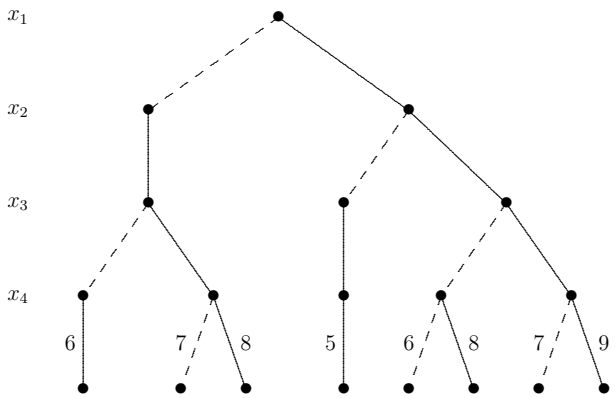


Fig. 2. Branching tree for the set covering problem in Table 1(a). Only feasible leaf nodes are shown.

Consider, for example, the nonseparable objective function of Table 1(b). There are many ways to capture the function by putting arc costs on the branching tree in Fig. 2. The most obvious is to assign the value corresponding to each leaf node to the incoming arc, as shown in the figure, and zero cost on all other arcs. However, this assignment tends to result in a large decision diagram.

To find a reduced diagram, we focus on *canonical* arc costs. An assignment of arc costs to a tree or a decision diagram is canonical if for every level $i \geq 2$, the smallest cost on arcs $a(u, d_i)$ leaving any given node u is some predefined value α_i. In the simplest case $\alpha_i = 0$ for all i, but it is convenient in applications to allow other values. We first show that canonical costs are unique.

Lemma 1. *For any given decision diagram or search tree representing a feasible set S, there is at most one canonical assignment of arc costs to G that represents the optimization problem (1).*

Proof. To simplify notation, we assume without loss of generality that each $\alpha_i = 0$. Given node u in layer i, let $c(u, d_i)$ be the cost assigned to arc $a(u, d_i)$, and let $L_u(\bar{d})$ be the length of a path $\bar{d} = (\bar{d}_i, \dots, \bar{d}_n)$ from u to 1 (in a decision diagram) or to a leaf node (in a tree). We show by induction on $i = n, n-1, \dots, 1$ that for any node u, $L_u(\bar{d})$ for any such path $\bar{d}$ is uniquely determined if the arc costs are canonical. It follows that all the arc costs are uniquely determined. First consider any node u on layer n. For any path $d = (d_1, \dots, d_{n-1})$ from r to u, we must have

$$f(d, d_n) - f(d, d'_n) = c(u, d_n) - c(u, d'_n)$$

for any pair of arcs $(u, u(d_n))$, $(u, u(d'_n))$. Because a canonical assignment satisfies $\min_{d_n}\{c(u, d_n)\} = 0$, each of the costs $c(u, d_n)$ is uniquely determined. Now for any node u in layer $i \in \{1, \dots, n-1\}$, suppose that $L_{u(d_i)}(\bar{d})$ is uniquely determined for any arc $a(u, d_i)$ and any path $\bar{d}$ from $u(d_i)$ to 1 or a leaf node.

```
For i = n, n − 1, . . . , 2:
    For each node u on layer i:
        Let c_min(u) = min_{u′ ∈ U_out} {c_{uu′}}
        For each u′ ∈ U_out:
            Let c_{uu′} ← c_{uu′} − c_min + α_i.
        For each u′ ∈ U_in:
            Let c_{u′u} ← c_{u′u} + c_min − α_i.
```

Fig. 3. Algorithm for converting arc costs to canonical arc costs. Here, U_{out} is the set of child nodes of node u, and U_{in} is the set of parent nodes of u.

Then for any path $d = (d_1, \ldots, d_{i-1})$ from r to u and for any pair $(d_i, \bar{d})$, $(d'_i, \bar{d}')$ of paths from u to 1 or a leaf node, we must have

$$\begin{aligned} f(d, d_i, \bar{d}) - f(d, d'_i, \bar{d}') &= L_u(d_i, \bar{d}) - L_u(d'_i, \bar{d}') \\ &= (c(u, d_i) + L_{u(d_i)}(\bar{d})) - (c(u, d'_i) + L_{u(d'_i)}(\bar{d}') \\ &= c(u, d_i) - c(u, d'_i) + \Delta \end{aligned}$$

where Δ is uniquely determined, by the induction hypothesis. This and the fact that $\min_{d_i}\{c(u, d_i)\} = 0$ imply that the arc costs $c(u, d_i)$ are uniquely determined. So $L_u(d_i, \bar{d})$ is uniquely determined for any $(d_i, \bar{d})$, as claimed. □

Canonical arc costs can be obtained in a search tree by moving nonzero costs upward in the tree. Beginning at the bottom, we do the following: if $c_{\min}(u)$ is the minimum cost on arcs leaving a given node u in layer i, then reduce the costs on these arcs by $c_{\min}(u) - \alpha_i$, and increase the costs on the arcs entering the node by $c_{\min}(u) - \alpha_i$. The algorithm appears in Fig. 3, and the result for the example appears in Fig. 4 (assuming each $\alpha_i = 0$). If the algorithm is applied to the search tree for the separable objective function $f(x) = 3x_1 + 5x_2 + 4x_3 + 6x_4$, the resulting canonical arc costs are those of Fig. 5(a).

A reduced weighted decision diagram can now be obtained from the branching tree of Fig. 4 much as in the unweighted case. Let subgraphs $G_{uu'}$ and $G_{vv'}$ be equivalent when they are isomorphic, corresponding arcs have the same labels *and costs*, and u, v belong to the same layer. A weighted decision diagram with canonical arc costs is *reduced* if it contains no equivalent subgraphs. A reduced weighted decision diagram can be obtained by superimposing all leaf nodes of the branching tree to obtain node 1, and then continuing to superimpose equivalent subgraphs until none remain. The branching tree of Fig. 4 reduces to the weighted decision diagram of Fig. 5(b). Note that the diagram is larger than the reduced diagram for the separable objective function, which appears in Fig. 5(a).

Theorem 1. *Any given discrete optimization problem (1) is represented by a unique reduced weighted decision diagram with canonical arc costs.*

Proof. We first construct a reduced weighted decision diagram G that represents (1) and has canonical costs, assuming all $\alpha_i = 0$ (the argument is similar for

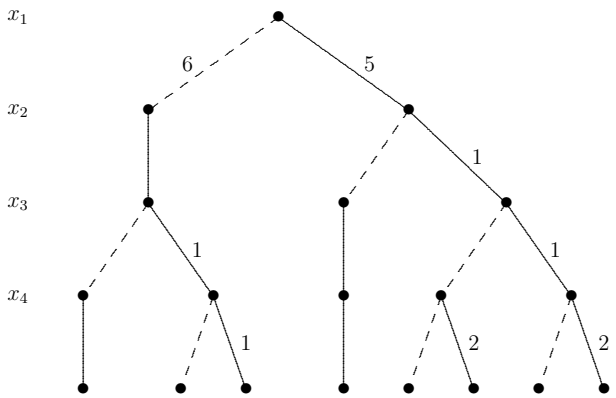


Fig. 4. Branching tree with canonical arc costs. Unlabeled arcs have zero cost.

arbitrary α_i). We will then show that G is reduced and unique. Define function g by $g(x) = f(x)$ when $x \in S$ and $g(x) = \infty$ when $x \notin S$. For each $i = 1, \ldots, n$ and each $d = (d_1, \ldots, d_{i-1}) \in D_1 \times \cdots \times D_{i-1}$ define the partial function g_{id} by

$$g_{id}(x_i, \ldots, x_n) = g(d_1, \ldots, d_{i-1}, x_i, \ldots, x_n)$$

Let partial function g_{id} be *finite* if $g_{id}(x_i, \ldots, x_n)$ is finite for some $x_i, \ldots, x_n$. By convention, $g_{n+1,d}() = 0$ for $d \in S$. We say that partial functions g_{id} and $g_{id'}$ are *equivalent* if both are finite and agree on relative values; that is,

$$g_{id}(x_i, \ldots, x_n) - g_{id}(x'_i, \ldots, x'_n) = g_{id'}(x_i, \ldots, x_n) - g_{id'}(x'_i, \ldots, x'_n),$$

for any pair $(x_i, \ldots, x_n)$, $(x'_i, \ldots, x'_n)$.

Now construct G as follows. In each layer $i \in \{1, \ldots, n+1\}$, create a node u in G for each equivalence class of finite partial functions g_{id}. Create outgoing arc $a(u, d_i)$ for each $d_i \in D_i$ such that $g_{i+1,(d,d_i)}$ is finite, where g_{id} is any function in the equivalence class for u. For $i \geq 2$ let arc $a(u, d_i)$ have cost

$$c(u, d_i) = g_{id}(d_i, p_u(d_i)) - c_{\min}(u) \tag{2}$$

where $p_u(d_i)$ is a shortest path from $u(d_i)$ to 1. Then the arc costs are defined recursively for $i = n, n-1, \ldots, 2$. Arcs $a(r, d_1)$ leaving the root node have cost $c(r, d_1) = g(d_1, p_r(d_1))$.

We will say that G_{u1} represents the finite partial function g_{id} if

$$g_{id}(x_i, \ldots, x_n) - g_{id}(x'_i, \ldots, x'_n) = L_u(x_i, \ldots, x_n) - L_u(x'_i, \ldots, x'_n) \tag{3}$$

for all pairs $(x_i, \ldots, x_n)$, $(x'_i, \ldots, x'_n)$. We will show by induction on i that for any node u in layer $i \geq 2$, (i) path $p_u(d_i)$ has length zero for any arc $a(u, d_i)$, and (ii) subgraph G_{u1} represents any function g_{id} in the equivalence class corresponding

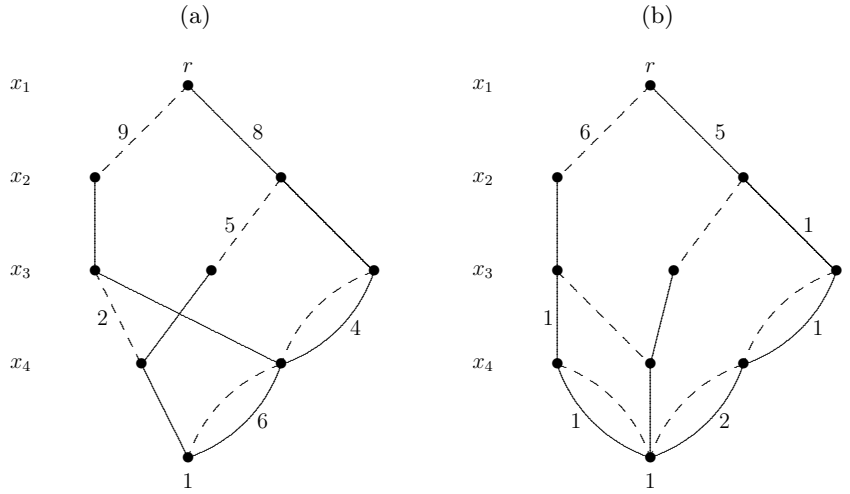


Fig. 5. (a) Weighted decision diagram with canonical arc costs for a separable objective function. (b) Canonical arc costs for a nonseparable objective function.

to u. Given this, for any feasible solution $x = (d_1, d)$, we have

$$\begin{aligned} g(d_1, d) &= g(d_1, p_r(d_1)) + g(d_1, d) - g(d_1, p_r(d_1)) && \text{(a)} \\ &= g(d_1, p_r(d_1)) + L_r(d_1, d) - L_r(d_1, p_r(d_1) && \text{(b)} \\ &= c(r, d_1) + L_r(d_1, d) - L_r(d_1, p_r(d_1)) && \text{(c)} \\ &= c(r, d_1) + L_r(d_1, d) - [c(r, d_1) + L_{r(d_1)}(p_r(d_1))] && \text{(d)} \\ &= c(r, d_1) + L_r(d_1, d) - c(r, d_1) = L_r(d_1, d) && \text{(e)} \end{aligned}$$

where (b) is due to (ii), (c) follows from the definition of $c(r, d_1)$, and (e) is due to (i). This means that for feasible x, $g(x)$ is the length of path x in G. Because the nodes of G represent only finite partial functions g_{id}, G contains no path for infeasible x. Thus, G represents (1).

For the inductive proof, we first observe that for any node u in layer n, path $p_u(d_n)$ is simply node 1 and therefore has length zero, which yields (i). Also for any pair x_n, x'_n, we have for any g_{id} in the equivalence class for u:

$$\begin{aligned} g_{nd}(x_n) - g_{nd}(x'_n) &= (g_{nd}(x_n) - c_{\min}(u)) - (g_{nd}(x'_n) - c_{\min}(u)) \\ &= c(u, x_u) - c(u, x'_u) = L_u(x_n) - L_u(x'_n) \end{aligned}$$

This means that G_{u1} represents g_{nd}, and we have (ii).

Supposing that (i) and (ii) hold for layer $i+1$, we now show that they hold for layer i. To show (i), we note that the length of $p_u(d_i)$ is

$$\min_{d_{i+1}} \{c(u(d_i), d_{i+1}) + p_u(d_{i+1})]\} = \min_{d_{i+1}} \{c(u(d_i), d_{i+1})\} = 0$$

where the first equation is due to the induction hypothesis and the second to the definition of $c(u(d_i), d_{i+1})$. To show (ii), it suffices to show (3), which can be

written

$$g_{id}(x_i, y) - g_{id}(x'_i, y') = L_u(x_i, y) - L_u(x'_i, y') \tag{4}$$

where $y = (x_{i+1}, \ldots, x_n)$. Note that

$$\begin{aligned}
&g_{id}(x_i, y) - g_{id}(x'_i, y') \\
&\quad = g_{id}(x_i, p_u(x_i)) - g_{id}(x'_i, p_u(x'_i)) \\
&\qquad\quad + g_{id}(x_i, y) - g_{id}(x_i, p_u(x_i)) - [g_{id}(x'_i, y') - g_{id}(x'_i, p_u(d'_i))] && \text{(a)} \\
&\quad = g_{id}(x_i, p_u(x_i)) - g_{id}(x'_i, p_u(x'_i)) \\
&\qquad\quad + L_{u(x_i)}(y) - L_{u(x_i)}(p_u(x_i)) - [L_{u(x'_i)}(y') - L_{u(x'_i)}(p_u(x'_i))] && \text{(b)} \\
&\quad = g_{id}(x_i, p_u(x_i)) - g_{id}(x'_i, p_u(x'_i)) + L_{u(x_i)}(y) - L_{u(x'_i)}(y') && \text{(c)} \\
&\quad = c(u, x_i) - c(u, x'_i) + L_{u(x_i)}(y) - L_{u(x'_i)}(y') && \text{(d)} \\
&\quad = c(u, x_i) + L_{u(x_i)}(y) - [c(u, x'_i) + L_{u(x'_i)}(y')] && \text{(e)} \\
&\quad = L_u(x_i, y) - L_u(x'_i, y') && \text{(f)}
\end{aligned}$$

where (b) is due to the induction hypothesis for (ii), (c) is due to the induction hypothesis for (i), and (d) is due to (2). This demonstrates (4).

We now show that G is minimal and unique. Suppose to the contrary that some weighted decision diagram $\bar{G}$ with canonical costs represents (1), is no larger than G, and is different from G. By construction, there is a one-to-one correspondence of nodes on layer i of G and equivalence classes of partial functions g_d. Thus for some node u on layer i if $\bar{G}$, there are two paths d, d' from r to u for which g_d and $g_{d'}$ belong to different equivalence classes. However, $\bar{G}$ represents g, which means that for any path $y = (y_i, \ldots, y_n)$ from u to 1,

$$\begin{aligned}
g_d(y) &= \bar{L}(d) + \bar{L}_u(y) \\
g_{d'}(y) &= \bar{L}(d') + \bar{L}_u(y)
\end{aligned} \tag{5}$$

where $\bar{L}(d)$ is the length of the path d from 1 to u in $\bar{G}$, and $\bar{L}_u(y)$ the length of the path y from u to 1 in $\bar{G}$. This implies that for any two paths y and y' from u to 1 in $\bar{G}$,

$$g_d(y) - g_d(y') = g_{d'}(y) - g_{d'}(y') = \bar{L}_u(y) - \bar{L}_u(y')$$

which contradicts the fact that g_d and $g_{d'}$ belong to different equivalence classes. □

5 Separable Decision Diagrams

A *separable* decision diagram is a weighted decision diagram whose arc costs are directly obtained from a separable objective function. For example, the diagram of Fig. 1(b) is separable. More precisely, a decision diagram is separable if on any layer i, $c(u, d_i) = c(u', d_i) = c_i(d_i)$ for any d_i and any two nodes u, u'.

Separable decision diagrams have the advantage that they can be reduced while ignoring arc costs. That is, the reduced diagram is obtained by removing

arc costs, reducing the diagram that results, and then putting cost $c_i(d_i)$ on each arc $a(u, d_i)$.

Converting costs to canonical costs can destroy separability. For example, the separable decision diagram of Fig. 1(b) becomes the nonseparable diagram of Fig. 5(a) when costs are converted to canonical costs. However, the *reduced* diagram remains the same. Thus Fig. 5(a) is a reduced weighted decision diagram.

This means that there is nothing lost by converting to canonical costs when the diagram is separable, and perhaps much to be gained when it is nonseparable, because in the latter case the diagram may reduce further.

Lemma 2. *A separable decision diagram that is reduced when costs are ignored is also reduced when costs are converted to canonical costs.*

Proof. Suppose that G is reduced when costs are ignored, but it is not reduced when costs are converted to canonical costs. Then some two weighted subgraphs G_{u1} and $G_{u'1}$ are equivalent. This means, in particular, that they are isomorphic. But this contradicts the assumption that G without costs is reduced. □

6 Dynamic Programming

A *dynamic programming problem* is one in which the variables x_i are regarded as *controls* that result in transitions from one state to the next. In particular, control x_i takes the system from the current state s_i to the next state

$$s_{i+1} = \phi_i(s_i, x_i), \quad i = 1, \ldots n \tag{6}$$

where the initial state s_1 is given. It is assumed that the objective function $f(x)$ is a separable function of *control/state pairs*, so that

$$f(x) = \sum_{i=1}^{n} c_i(x_i, s_i) \tag{7}$$

The optimization problem is to minimize $f(x)$ subject to (7) and $x_i \in X_i(s_i)$ for each i.

The attraction of a dynamic programming formulation is that it can be solved recursively:

$$g_i(x_i) = \min_{x_i \in X_i(s_i)} \{c_i(s_i, x_i) + g_{i+1}(\phi_i(s_i, x_i))\}, \quad i = 1, \ldots, n \tag{8}$$

where $g_{n+1}(s_{n+1}) = 0$ for all s_{n+1}. The optimal value is $g_1(s_1)$. To simplify discussion, we will suppose that $X_n(s_n)$ is defined so that there is only one final state s_{n+1}.

To make a connection with decision diagrams, we will assume that the control variables x_i are discrete. Then the recursion (8) describes a *state transition graph* in which each node corresponds to a state, and there is a directed arc

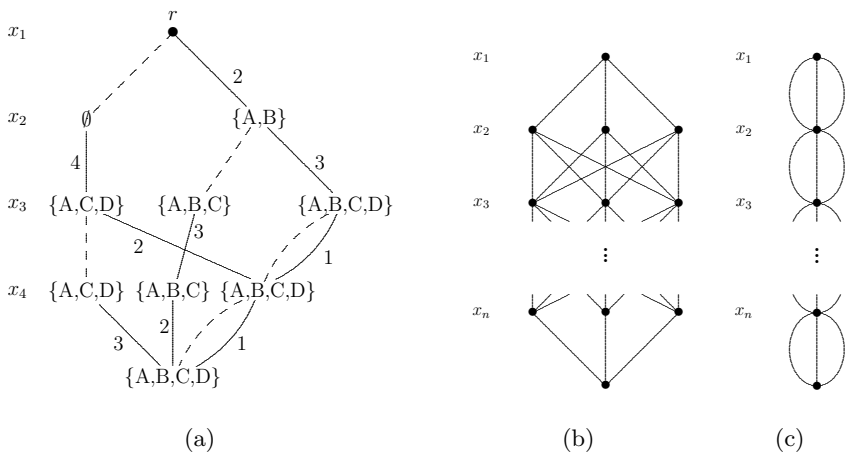


Fig. 6. (a) State transition graph for a set covering instance. (b) State transition graph for a production and inventory management problem. (c) Reduced state transition graph after converting costs to canonical costs.

$(s_i, \phi_i(s_i, x_i))$ with cost $c_i(s_i, x_i)$ for each control $x_i \in X_i$. The state transition graph is a binary decision diagram in which s_1 is the root node and s_{n+1} the terminal node.

In the set covering example, the state s_i can be defined as the set of elements that have been covered after variables $x_1, \ldots, x_{i-1}$ have been fixed. The resulting state transition graph appears in Fig. 6(a).

We see immediately that the state transition graph, when viewed as a decision diagram, may allow further reduction. Two of the nodes on level 4 of Fig. 6(a) can be merged even though they correspond to different states.

7 Reducing the State Transition Graph

It may be possible to reduce the size of a DP state transition graph by viewing it as a weighted decision diagram. Even when arc costs as given in the graph prevent reduction, conversion of the arc costs to canonical costs may allow significant reduction that simplifies the problem.

For example, this idea can be applied a textbook DP model that has remained essentially unchanged for decades. The objective is to adjust production quantities and inventory levels to meet demand over n periods while minimizing production and holding costs. We will suppose that h_i the unit holding cost in period i, and c_i is the unit production cost. Let x_i be the production quantity in period i, and let the state variable s_i be the stock on hand at the beginning of period i. Then the recursion is

$$g_i(s_i) = \min_{x_i \in X_i(s_i)} \{c_i x_i + h_i s_i + g_{i+1}(s_i + x_i - d_i)\}, \quad i = 1, \ldots, n \qquad (9)$$

where d_i is the demand in period i.

If we suppose that the warehouse has capacity m in each period, the state transition graph has the form shown in Fig. 6(b). Note that the set of arcs leaving any node is essentially identical to the set of arcs leaving any other node in the same stage. The controls x_i and the costs are different, but the controls can be equalized by a simple change of variable, and the costs can be equalized by transforming them to canonical costs.

To equalize the controls, let the control x'_i be the stock level at the beginning of the next stage, so that $x'_i = s_i + x_i - d_i$. Then the controls leaving any node are $x'_i = 0, \ldots, m$. The recursion (9) becomes

$$g_i(s_i) = \min_{x'_i \in \{0,\ldots,m\}} \{c_i(x'_i - s_i + d_i) + h_i s_i + g_{i+1}(x'_i)\}, \quad i = 1, \ldots, n \qquad (10)$$

To transform the costs to canonical costs, we subtract $h_i s_i + (m - s_i)c_i$ from the cost on each arc (s_i, s_{i+1}), and add this amount to each arc coming into s_i. Then for any period i, the arcs leaving any given node s_i have the same set of costs. Specifically, arc (s_i, s_{i+1}) has cost

$$\bar{c}_i(s_{i+1}) = (d_i + s_{i+1} - m)c_i + s_{i+1}h_{i+1} + (m - s_{i+1})c_{i+1}$$

and so depends only on the next state s_{i+1}. These costs are canonical for $\alpha_i = \min_{s_{i+1} \in \{0,\ldots,m\}} \{\bar{c}_i(s_{i+1})\}$.

In any given period i, the subgraphs G_{s_i} are now equivalent, and the decision diagram can be reduced as in Fig. 6(c). There is now one state in each period rather than m, and the recursion is

$$g_i = \min_{x'_i \in \{0,\ldots,n\}} \{c_i(d_i + x'_i - m) + h_{i+1}x'_i + c_{i+1}(m - x'_i) + g_{i+1}\} \qquad (11)$$

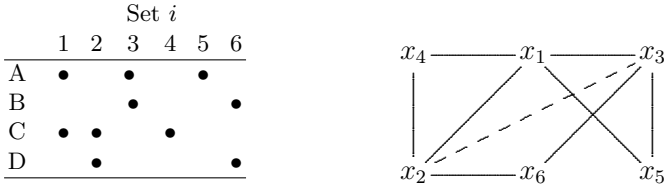
If $\bar{x}'_1, \ldots, \bar{x}'_n$ are the optimal controls in (11), the resulting stock levels are given by $s_{i+1} = \bar{x}'_i$ and the production levels by $x_i = \bar{x}'_i - s_i + d_i$.

There is no need for experiments to determine the effect on computation time. The reduction of the state space implies immediately that the time is reduced by a factor of m.

8 Nonserial Dynamic Programming

In serial dynamic programming, the state variables s_i are arranged linearly as a path. In *nonserial dynamic programming*, they are arranged in a tree. Because formal notation for nonserial DP is rather complex, the idea is best introduced by example. To simplify exposition, we will discuss only problems with separable objective functions.

Figure 7(a) shows a small set partitioning problem. The goal is to select a minimum subcollection of the 6 sets that partitions the set $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$. Thus there are 4 constraints, corresponding to the elements (A, B, C, D), each requiring that only one set containing that element be selected. The 3 feasible solutions are $(x_1, \ldots, x_6) = (0,0,0,1,1,1), (0,1,1,0,0,0), (1,0,0,0,0,1)$, where $x_i = 1$ indicates that set i is selected. The last two solutions are optimal.



(a) (b)

Fig. 7. (a) A small set partitioning problem. The dots indicate which elements belong to each set i. (b) Dependency graph for the problem. The dashed edge is an induced edge.

A nonserial recursion can be constructed by reference to the dependency graph for the problem, shown in Fig. 7(b). The graph connects two variables with an edge when they occur in a common constraint. We arbitrarily select a variable ordering $x_1, \ldots, x_6$ and remove vertices in reverse order, each time connecting vertices adjacent to the vertex removed. Edges added in this fashion are *induced edges*. As in serial DP, we let the state contain the information necessary to determine the feasible choices for x_i. Now, however, the feasible values of x_i depend on the set of variables to which x_i was adjacent when removed. We therefore let x_6 depend on (x_2, x_3), and similarly for the other control variables.

The problem can be solved recursively as follows. Let $C_{\mathrm{A}}(x_1, x_3, x_5)$, $C_{\mathrm{B}}(x_3, x_6)$, $C_{\mathrm{C}}(x_1, x_2, x_4)$, and $C_{\mathrm{D}}(x_2, x_6)$ be the constraints corresponding to elements A, B, C, and D, respectively. The recursion is

$$
\begin{aligned}
g_6(x_2, x_3) &= \min_{x_6 \in \{0,1\}} \{x_6 \mid C_{\mathrm{C}}(x_3, x_6) \wedge C_{\mathrm{D}}(x_2, x_6)\} \\
g_5(x_1, x_3) &= \min_{x_5 \in \{0,1\}} \{x_5 \mid C_{\mathrm{A}}(x_1, x_3, x_5)\} \\
g_4(x_1, x_2) &= \min_{x_4 \in \{0,1\}} \{x_4 \mid C_{\mathrm{C}}(x_1, x_2, x_4)\} \\
g_3(x_1, x_2) &= \min_{x_3 \in \{0,1\}} \{x_3 + g_6(x_2, x_3) + g_5(x_1, x_3)\} \\
g_2(x_1) &= \min_{x_2 \in \{0,1\}} \{x_2 + g_4(x_1, x_2) + g_3(x_1, x_2)\} \\
g_1() &= \min_{x_1 \in \{0,1\}} \{x_1 + g_2(s(1))\}
\end{aligned}
$$

The smallest partition has size $g_1()$, which is infinite if there is no feasible partition. The induced width (treewidth) of the dependency graph is the maximum number of state variables that appear as arguments of a $g_i(\cdot)$, in this case 2.

To write the general recursion for nonserial DP, let $x(J) = \{x_j \mid j \in J\}$. Let each constraint C_i contains the variables in $x(J_i)$, and let each $g_i(\cdot)$ be a function of $x(I_i)$. The recursion is

$$
g_i(x(I_i)) = \max_{x_i} \left\{ c_i(x_i) + \sum_{\substack{j > i \\ I_j \subset I_i}} g_j(x(I_j)) \;\middle|\; \bigwedge_{\substack{j \\ J_j \subset I_i}} C_j(x(J_j)) \right\}
$$

The complexity of the recursion is $\mathcal{O}(2^{W+1})$, where W is the induced width.

As in serial DP, we can introduce state variables s_i to encode the necessary information for selecting x_i. In the present case, we let s_i be the multiset of elements selected by the variables in $x(I_i)$, where elements selected by two variables are listed twice. For example, s_6 is the multiset of elements selected by x_2 and x_3. In general, we will let $s(I_i)$ denote the multiset of elements selected by the variables in $x(I_i)$, so that $s_i = s(I_i)$. The solution can now be calculated as shown in Table 2.

The state transition graph for the example appears in Fig. 8(a). Here each stage is labeled by the state variable s_i rather than the decision variable x_i. The initial state is associated with state variable s_1. Decision x_i is taken at each value of state variable s_i. Because state variables s_3 and s_4 are identical, they are superimposed in the graph. The choice of x_3 leads to states s_5 and s_6, with two outgoing arcs corresponding to each choice. The choice of x_4 leads to a terminal node.

Feasible solutions correspond to trees that are incident to the initial state and 3 terminal nodes. The tree shown in bold is one of the two optimal solutions. Note that its cost is 2 even though it contains 3 solid arcs, because two of the arcs correspond to the same choice $x_3 = 1$. States that are part of no feasible solution (tree) are omitted from the diagram.

As an illustration, consider state $s_3 = \{\mathrm{A}, \mathrm{C}\}$. The arcs for $x_3 = 0$ lead to the states $s_5 = \{\mathrm{A},\mathrm{C}\}$ and $s_6 = \emptyset$. Arcs for $x_3 = 1$ are not shown because they are not part of a feasible solution. One can check from Table 2 that when $x_3 = 1$,

$$x_3 + g_6(s_6) + g_5(s_5) = 1 + g_6(\{\mathrm{A},\mathrm{B}\}) + g_5(\{\mathrm{A},\mathrm{C},\mathrm{C},\mathrm{D}\}) = \infty$$

Now consider state $s_4 = \{\mathrm{A}, \mathrm{C}\}$, which corresponds to the same node. The arc for $x_4 = 0$ leads to a terminal node. The arc for $x_4 = 1$ is not shown because $\{\mathrm{A},\mathrm{C},\mathrm{C}\}$ violates constraint C_{C}.

9 Nonserial Decision Diagrams

The state transition graph for a nonserial DP can be regarded as a decision diagram, as in the case of serial DP. The diagram can also be reduced in a similar fashion. For example, the diagram of Fig. 7(a) reduces to that of Fig. 7(b). Note that two nodes are merged, resulting in a smaller diagram. Feasible solutions correspond to trees that are incident to the root node and the two terminal

Table 2. Recursive solution of the set partitioning example.

s_6	$g_6(s_6)$
$\emptyset$	1
{A,B}	∞
{C,D}	∞
{A,B,C,D}	0

s_5	$g_5(s_5)$
$\emptyset$	1
{A,C}	0
{A,B}	0
{A,A,B,C}	∞

s_4	$g_4(s_4)$
$\emptyset$	1
{A,C}	0
{C,D}	0
{A,C,C,D}	∞

s_3	$g_3(s_3)$
$\emptyset$	2
{A,C}	1
{C,D}	1
{A,C,C,D}	∞

s_2	$g_2(s_2)$
$\emptyset$	2
{A,B}	2

$g_1() = 2$

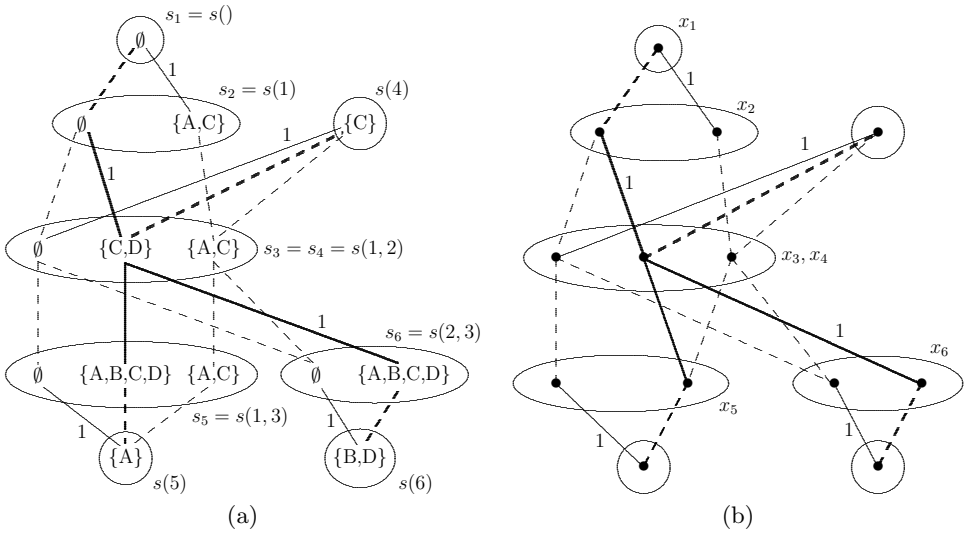


Fig. 8. (a) Nonserial state transition graph for a set partitioning problem. Only nodes and arcs that are part of feasible solutions are shown. Each feasible solution corresponds to a tree incident to the root and both terminal nodes. The boldface tree corresponds to optimal solution $(x_1, \dots, x_6) = (0, 1, 1, 0, 0, 0)$. (b) Reduced nonserial decision diagram for the same problem.

nodes. The reduction in size can be significant, as in the case of serial decision diagrams.

10 Conclusion

We showed how decision diagrams can be extended to weighted decision diagrams so as to establish a precise parallel with dynamic programming (DP). In particular, we proved that for a given optimization problem and variable ordering, there is a unique reduced decision diagram with canonical arc costs that represents the problem. We also showed how this perspective can allow one to simplify a discrete DP model by transforming arc costs on its state transition graph to canonical arc costs and reducing the diagram that results. Finally, we introduced nonserial decision diagrams as a counterpart to nonserial dynamic programming.

It remains to investigate other possible simplifications of DP models based on the decision diagram perspective, as well as to generalize the uniqueness result to nonserial decision diagrams. Another possible development is to merge relaxed decision diagrams, mentioned in Section 2, with approximate dynamic programming [32]. This may allow algorithms for relaxing a decision diagram to generate an efficient state space relaxation for approximate DP.

References

1. S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C-27:509–516, 1978.
2. H. R. Andersen. An introduction to binary decision diagrams. Lecture notes, available online, IT University of Copenhagen, 1997.
3. H. R. Andersen, T. Hadžić, J. N. Hooker, and P. Tiedemann. A constraint store based on multivalued decision diagrams. In C. Bessiere, editor, *Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *Lecture Notes in Computer Science*, pages 118–132. Springer, 2007.
4. S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Jorunal on Algebraic and Discrete Mathematics*, 8:277–284, 1987.
5. S. Arnborg and A. Proskurowski. Characterization and recognition of partial k-trees. *SIAM Jorunal on Algebraic and Discrete Mathematics*, 7:305–314, 1986.
6. F. Bacchus, S. Dalmao, and T. Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pages 340–351. 2003.
7. F. Bacchus, S. Dalmao, and T. Pitassi. Solving #SAT and Bayesian inference with backtracking search. *Journal of Artificial Intelligence Research*, 34:391–442, 2009.
8. P. Beame, R. Impagliazzo, T. Pitassi, and N. Segerlind. Memoization and DPLL: Formula caching proof systems. In *18th IEEE Annual Conference on Computational Complexity*, pages 248–259, 2003.
9. B. Becker, M. Behle, F. Eisenbrand, and R. Wimmer. BDDs in a branch and cut framework. In S. Nikoletseas, editor, *Experimental and Efficient Algorithms, Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA 05)*, volume 3503 of *Lecture Notes in Computer Science*, pages 452–463. Springer, 2005.
10. M. Behle and . Eisenbrand. 0/1 vertex and facet enumeration with BDDs. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments and the 4th Workshop on Analytic Algorithms and Combinatorics*, pages 158–165, 2007.
11. Markus Behle. *Binary Decision Diagrams and Integer Programming*. PhD thesis, Max Planck Institute for Computer Science, 2007.
12. R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60:503–516, 1954.
13. R. Bellman. *Dynamic programming*. Priceton University Press, Princeton, NJ, 1957.
14. D. Bergman and A. A. Ciré and W.-J. van Hoeve and J. N. Hooker. Optimization bounds from binary decision diagrams. technical report, Carnegie Mellon University, 2012.
15. D. Bergman, W.-J. van Hoeve, and J. N. Hooker. Manipulating MDD relaxations for combinatorial optimization. In T. Achterberg and J. S. Beck, editors, *Proceedings of the International Workshop on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combintaorial Optimization Problems (CPAIOR 2011)*, volume 6697 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2011.
16. U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, New York, 1972.
17. D. P. Bertsekas. *Dynamic Programming and Optimal Control, 3rd ed.*, volume 1 and 2. Athena Scientific, Nashua, NH, 2001.

18. B. Bollig, M. Sauerhoff, D. Sieling, and I. Wegener. Binary decision diagrams. In Y. Crama and P. L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 473–505. Cambridge University Press, Cambridge, UK, 2010.
19. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.
20. D. Chhajed and T. J. Lowe. Solving structured multifacility location problems efficiently. *Transportation Science*, 28:104–115, 1994.
21. A. A. Ciré and W.-J. van Hoeve. MDD propagation for disjunctive scheduling. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 11–19. AAAI Press, 2012.
22. Y. Crama, P. Hansen, and B. Jaumard. The basic algorithm for pseudoboolean programming revisited. *Discrete Applied Mathematics*, 29:171–185, 1990.
23. R. Dechter. Bucket elimination: A unifying framework for several probabilistic inference algorithms. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI 96)*, pages 211–219, Portland, OR, 1996.
24. E. V. Denardo. *Dynamic Programming: Models and Applications*. Dover Publications, Mineola, NY, 2003.
25. T. Hadžić and J. N. Hooker. Postoptimality analysis for integer programming using binary decision diagrams, presented at GICOLAG workshop (Global Optimization: Integrating Convexity, Optimization, Logic Programming, and Computational Algebraic Geometry), Vienna. Technical report, Carnegie Mellon University, 2006.
26. T. Hadžić and J. N. Hooker. Cost-bounded binary decision diagrams for 0-1 programming. Technical report, Carnegie Mellon University, 2007.
27. T. Hadžić, J. N. Hooker, B. O'Sullivan, and P. Tiedemann. Approximate compilation of constraints into multivalued decision diagrams. In P. J. Stuckey, editor, *Principles and Practice of Constraint Programming (CP 2008)*, volume 5202 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2008.
28. S. Hoda, W.-J. van Hoeve, and John N. Hooker. A systematic approach to MDD-based constraint programming. In *Proceedings of the 16th International Conference on Principles and Practices of Constraint Programming*, Lecture Notes in Computer Science. Springer, 2010.
29. J. Huang and A. Darwiche. DPLL with a trace: From SAT to knowledge compilation. In *International Joint Conference On Artificial Intelligence (IJCAI 2005)*, volume 19, pages 156–162. Lawrence Erlbaum Associates, 2005.
30. S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, 50:157–224, 1988.
31. C. Y. Lee. Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal*, 38:985–999, 1959.
32. W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Diumensionality, 2nd ed.* Wiley, 2011.
33. G. Shafer, P. P. Shenoy, and K. Mellouli. Propagating belief functions in qualitative markov trees. *International Journal of Approximate Reasoning*, 1:349–400, 1987.
34. P. P. Shenoy and G. Shafer. Propagating belief functions with local computation. *IEEE Expert*, 1:43–52, 1986.
35. D. Sieling and I. Wegener. *NC*-algorithms for operations on binary decision diagrams. *Parallel Processing Letters*, 3:3–12, 1993.