Stochastic Decision Diagrams

J. N. Hooker(✉)

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

Abstract. We introduce stochastic decision diagrams (SDDs) as a generalization of deterministic decision diagrams, which in recent years have been used to solve a variety of discrete optimization and constraint satisfaction problems. SDDs allow one to extend the relaxation techniques of deterministic diagrams to stochastic dynamic programming problems in which optimal controls are state-dependent. In particular, we develop sufficient conditions under which node merger operations applied during top-down compilation of the SDD yield a valid relaxed SDD whose size can be limited as desired. The relaxed SDD provides bounds on the optimal value that can be used to evaluate the quality of solutions obtained heuristically or to accelerate the search for an optimal solution. This results in a general and completely novel method for obtaining optimization bounds for stochastic dynamic programming, and the only method that can be applied to the original state space. We report computational experience on stochastic maximum clique (equivalently, maximum independent set) problem instances.

Keywords: stochastic decision diagrams · stochastic dynamic programming · relaxed decision diagrams · maximum clique problem

1 Introduction

Decision diagrams have proved to be a useful tool for solving a variety of discrete optimization and constraint programming problems [1,5,8,9,14,16–20,23,28,31]. One of their distinctive features is that they are naturally suited for dynamic programming (DP) formulations and provide an alternative to traditional state space enumeration as a means to solving them. They do so by creating a novel framework for branch-and-bound search [8] and a discrete relaxation method for calculating bounds [7]. Yet many, if not most, useful DP models are stochastic [29]. This suggests that it may be beneficial to extend the concept of a decision diagram to the stochastic case by introducing probabilities. Such is the goal of the present paper.

Stochastic decision diagrams (SDDs) can represent a very general class of discrete stochastic DP models with state-dependent transitions and controls. This is because the state transition graph of the DP model can be interpreted as an SDD. Despite this, the concept of an SDD is very different from that of a state

P. Schaus (Ed.): CPAIOR 2022, LNCS 13292, pp. 138–154, 2022.
https://doi.org/10.1007/978-3-031-08011-1_11

transition graph, partly because the nodes of the SDD need not correspond to states. In addition, a DP model can sometimes be radically simplified by reconceiving the transition graph as a decision diagram (without state information) and applying well-known reduction techniques to the diagram [24]. Perhaps more importantly, an SDD-based perspective opens the door to relaxation techniques that have been developed for decision diagrams, such as node merger and node splitting. Decision-diagram-based relaxation is attractive in that it does not presuppose linearity or convexity, and it allows one to obtain bounds of any desired quality by controlling the size of the relaxed diagram. It has already been widely applied to deterministic models, as recounted in [9].

We therefore focus on how to build relaxed SDDs, using node merger in particular. We develop sufficient conditions under which nodes can be merged during top-down compilation of an SDD so as to yield a valid relaxed SDD. The relaxed SDD can then provide bounds on the optimal value of the original problem. A complicating element of this analysis is that an optimal solution of a stochastic DP is not simply an assignment of values to variables, but a *policy* for selecting an optimal control in any state one happens to reach. Nonetheless, the analysis yields a general and completely novel method for bounding stochastic dynamic programming models. Such bounds are essential for judging the quality of a heuristic solution, and they can be very helpful for accelerating the search for an exact solution by excluding unpromising regions of the search space.

We assess the quality of SDD-based bounds experimentally using a stochastic version of the maximum clique problem, which is equivalent to the maximum independent set (maximum stable set) problem defined on the complementary graph. We select this problem because decision diagrams have already been evaluated as a bounding mechanism for the deterministic case, using the well-known DIMACS instance set [7]. It was found that decision diagrams supply tighter bounds, in less time, than the full cutting plane resources of a commercial integer programming solver. This suggests that bounds from an SDD may likewise be useful for the stochastic case, even while they cannot be directly compared with an integer programming model because no such practical model exists. As a test bed, we use a variety of random and DIMACS instances. We find that SDD-based bounds degrade rather modestly as one reduces the size of the relaxed SDD and consequently the time invested in building it, even when the time is reduced to a few seconds. These results suggest that SDDs can provide useful bounds of continuously adjustable quality.

2 Related Work

Decision diagrams were introduced as an optimization method in [21,22]. The idea of a relaxed diagram first appears in [1] as a technique for enhancing propagation in constraint programming, by means of both node splitting and node merger. Relaxed diagrams were first used to obtain optimization bounds in [7,10], and much subsequent work is described in [9]. Sufficient conditions under which node merger creates a relaxed diagram are presented in [25]. They are simpler

than the conditions developed below for SDDs because there is no need to accommodate solutions in the form of policies. Relaxed decision diagrams are combined with Lagrangian methods in [6,14,26].

Dynamic programming, both deterministic and stochastic, is credited to Bellman [3,4]. Various DP techniques are described in [12,13]. Due to the astronomical size of state spaces often encountered in stochastic DP, practitioners generally resort to approximate DP, which estimates the cost-to-go, rather than attempting to establish valid bounds [12,29]. Connections between decision diagrams and deterministic DP, including nonserial DP [11], are discussed in [24].

When DP bounds are desired, they are traditionally obtained by state space relaxation, which approximates the original state space with a smaller, computationally feasible space [2,15,27,30]. It differs in fundamental ways from relaxation based on decision diagrams. A relaxed decision diagram uses the same variables and state space as the original problem, rather than mapping the problem into a smaller space. This allows the relaxed diagram to provide a branching framework for an exact branch-and-bound method [8]. Furthermore, the relaxation is constructed dynamically during compilation rather than specified *a priori* and is thereby potentially more sensitive to problem structure. It can be tightened by filtering techniques and Lagrangian methods. Finally, the relaxed diagram can be sized to provide a bound of any desired quality.

3 Stochastic Decision Diagrams

A decision diagram can be defined as a directed, acyclic multigraph in which the nodes are partitioned into *layers*. Each arc of the graph is directed from a node in layer i to a node in layer $i+1$ for some $i \in \{1,\dots,n\}$. We let L_i represent the set of nodes in layer i. Layers 1 and $n+1$ contain a single node, namely the root r and the terminus t, respectively. Each layer i is associated with a variable x_i with finite domain $\mathcal{X}_i$. The arcs leaving any node in layer i have *labels* in $\mathcal{X}_i$, representing possible values of x_i (*controls*) at that node.

Conventional decision diagrams are *deterministic*, meaning that the control at a given node determines a transition to particular node on the next layer. A path from r to t defines a sequence of controls $\boldsymbol{x} = (x_1,\dots,x_n)$ as indicated by the arc labels on the path. A decision diagram is *weighted* if there is a length (cost) associated with each arc.

Any discrete optimization problem of the form $\min_{\boldsymbol{x}\in\boldsymbol{X}}\{\sum_{i=1}^{n} c_i(x_i)\}$ with finite-domain variables $x_1,\dots,x_n$ and feasible set $\boldsymbol{X}$ can be represented by a weighted decision diagram.[1] The diagram is constructed so that its r–t paths correspond to feasible solutions $\boldsymbol{x} \in \mathcal{X}$, and the length of any r–t path is the cost of the corresponding solution. The optimal value of the problem is the length of a shortest r–t path.

A *stochastic decision diagram* (SDD) associates probabilities as well as labels and costs with the arcs. An SDD is significantly different from a deterministic diagram in that a given control x_i at a node u does not, in general, determine

[1] Decision diagrams can also represent problems with nonseparable cost functions $c(\boldsymbol{x})$ as described in [24]. However, this generalization is not relevant here.

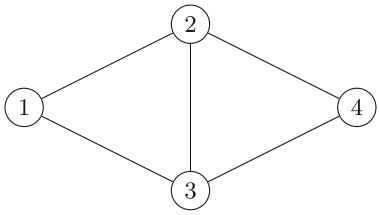

Fig. 1. Graph for a small maximum clique problem.

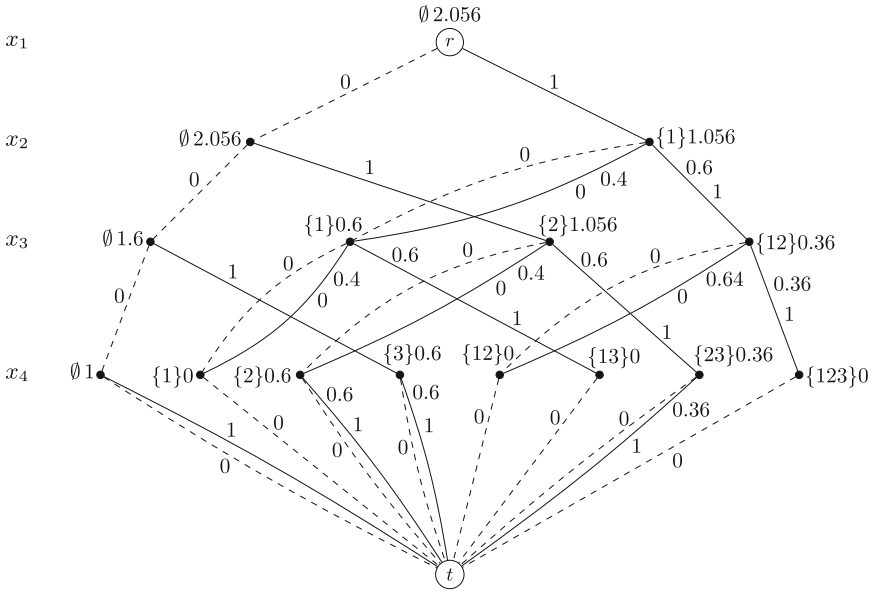

Fig. 2. SDD for a small stochastic maximum clique problem. Transition probabilities less than 1 are shown on the arcs. Zero-cost solid arcs from layer 4 to the terminus are not shown. States and maximum expected costs-to-go are indicated at nodes.

a transition to a particular node in the next layer. Rather, control x_i can lead to any of several possible *outcomes* ω, each with a probability $p_{i\omega}(u, x_i)$ and cost $c_{i\omega}(u, x_i)$. We suppose, without loss of generality, that the same set Ω_i of outcomes is available at each node in layer i, although many of the outcomes may have probability zero. Each outcome ω at node u with positive probability gives rise to an arc (u, u') that leads to a node $u' = \phi_{i\omega}(u, x_i)$.

Due to the probabilistic nature of transitions, a sequence of controls $\boldsymbol{x} = (x_1, \ldots, x_n)$ does not correspond to a single r–t path. Furthermore, the control exercised in layer i may depend on which node is reached in that layer. We therefore define a *policy* $\boldsymbol{\pi}$ rather than a vector $\boldsymbol{x}$ of controls for SDDs, where $\pi_i(u)$ is the specified control at node $u \in L_i$. The policy is *feasible* if $\pi_i(u) \in \mathcal{X}_i(u)$ for all $u \in L_i$ and all i, where $\mathcal{X}_i(u)$ is the set of permissible controls at node u.

As an example, consider a stochastic maximum clique problem associated with the graph of Fig. 1. Each edge of the graph appears with probability 0.6. The objective is to find a clique of maximum expected size, without knowing in advance which edges are present. The problem is represented by the SDD in Fig. 2. There are two controls x_i in each layer i, with $x_i = 1$ indicating a decision to include vertex i in the clique (solid arcs), and $x_i = 0$ a decision to exclude vertex i (dashed arcs). There are two solid arcs leaving most nodes, because there are two possible outcomes when $x_i = 1$: vertex i can be added to the clique ($\omega = 1$), or it cannot be added ($\omega = 0$). The former occurs when all the edges from vertex i to vertices currently in the clique exist. This outcome has probability 0.6^m, where m is the size of the clique, while outcome $\omega = 0$ has probability $1 - 0.6^m$. The resulting probabilities are indicated next to the arcs in Fig. 2 when they are less than 1. For readability, zero-cost solid arcs from layer 4 to the terminus (those corresponding to $\omega = 0$) are not shown in the figure.

Arc costs are also indicated in Fig. 2, where arcs with cost 1 correspond to outcome $\omega = 1$ and those with cost 0 to outcome $\omega = 0$. We seek a policy that maximizes expected cost; i.e., maximizes the expected size of the resulting clique. A policy $\boldsymbol{\pi}$ consists of a decision $\pi_i(u)$ at each node $u \in L_i$ as to whether vertex i should included in the clique. Thus one can consider the outcome of previous controls when deciding whether to include a given vertex.

The expected cost $c(D, \boldsymbol{\pi})$ of a policy $\boldsymbol{\pi}$ on a stochastic diagram D can be computed with the recursion

$$h_i(u, \boldsymbol{\pi}) = \sum_{\omega \in \Omega_i} p_{i\omega}\big(u, \pi_i(u)\big)\big[c_{i\omega}\big(u, \pi_i(u)\big) + h_{i+1}\big(\phi_{i\omega}(u, \pi_i(u)), \boldsymbol{\pi}\big)\big]$$

for $i = n, n-1, \ldots, 1$, where $h_{n+1}(t, \boldsymbol{\pi}) = 0$ and $c(D, \boldsymbol{\pi}) = h_1(r, \boldsymbol{\pi})$. A policy $\boldsymbol{\pi}^*$ is optimal if it minimizes $c(D, \boldsymbol{\pi})$ over all feasible policies $\boldsymbol{\pi}$. The minimum expected cost can be computed with the recursion

$$h_i(u) = \min_{x_i \in \mathcal{X}_i(u)} \Big\{ \sum_{\omega \in \Omega_i} p_{i\omega}(u, x_i)\big[c_{i\omega}(u, x_i) + h_{i+1}\big(\phi_{i\omega}(u, x_i)\big)\big] \Big\} \tag{1}$$

for $i = n, n-1, \ldots, 1$, where $h_{n+1}(t) = 0$. Here, $h_i(u)$ is the optimal expected *cost-to-go* at node u in layer i. The overall optimal cost is $c(D) = h_1(r)$. An optimal policy $\boldsymbol{\pi}^*$ is obtained by setting $\pi_i^*(u)$ to an optimizing value of x_i in (1) for each i and each $u \in L_i$.

Optimal expected costs-to-go for the example are shown at the nodes in Fig. 2 (along with the corresponding states, to be discussed in the next section). They are computed by maximizing rather than minimizing in (1), since we seek a maximum clique. The maximum expected cost for this problem instance is 2.056. This means that if an optimal choice is made at every node, the resulting clique will have size 2.056 on the average. Note that at the root node, either choice is optimal. At both nodes in layer 2, the optimal choice is to add vertex 2, and so forth.

4 Stochastic Dynamic Programming

Stochastic decision diagrams can represent a very general class of stochastic dynamic programming (DP) problems in which costs, probabilities and controls are state-dependent. Let $p_{i\omega}(\boldsymbol{S}_i, x_i)$ be the probability of outcome $\omega \in \Omega_i$ in state $\boldsymbol{S}_i$ under control x_i, and $c_{i\omega}(\boldsymbol{S}_i, x_i)$ the cost of that outcome.[2] Outcome ω effects a transition from state $\boldsymbol{S}_i$ to state $\phi_{i\omega}(\boldsymbol{S}_i, x_i)$. A solution of the problem is a policy $\boldsymbol{\pi} = (\pi_1, \dots \pi_n)$ that maps each possible state $\boldsymbol{S}_i$ to a control $\pi_i(\boldsymbol{S}_i)$. A policy $\boldsymbol{\pi}$ is feasible if $\pi_i(\boldsymbol{S}_i)$ is an available control for every state $\boldsymbol{S}_i$; that is, $\pi_i(\boldsymbol{S}_i) \in \mathcal{X}_i(\boldsymbol{S}_i)$. The expected cost of policy $\boldsymbol{\pi}$ is $h_1(\boldsymbol{S}_1, \boldsymbol{\pi})$, where

$$h_i(\boldsymbol{S}_i, \boldsymbol{\pi}) = \sum_{\omega \in \Omega_i} p_{i\omega}\big(\boldsymbol{S}_i, \pi_i(\boldsymbol{S}_i)\big)\big[c_{i\omega}\big(\boldsymbol{S}_i, \pi_i(\boldsymbol{S}_i)\big) + h_{i+1}\big(\phi_{i\omega}(\boldsymbol{S}_i, \pi_i(\boldsymbol{S}_i)), \boldsymbol{\pi}\big)\big] \quad (2)$$

for $i = n, n-1, \dots, 1$. There is a single initial state $\boldsymbol{S}_1$ and a single terminal state $\boldsymbol{S}_{n+1}$, with $h_{n+1}(\boldsymbol{S}_{n+1}, \boldsymbol{\pi}) = 0$. We refer to $h_i(\boldsymbol{S}_i, \boldsymbol{\pi})$ as the *expected cost-to-go* of state $\boldsymbol{S}_i$ under policy $\boldsymbol{\pi}$. A policy $\boldsymbol{\pi}^*$ is optimal if it minimizes $h_1(\boldsymbol{S}_1, \boldsymbol{\pi})$ over all feasible policies $\boldsymbol{\pi}$. An optimal solution can be found with the recursion

$$h_i(\boldsymbol{S}_i) = \min_{x_i \in \mathcal{X}_i(\boldsymbol{S}_i)} \Big\{ \sum_{\omega \in \Omega_i} p_{i\omega}(\boldsymbol{S}_i, x_i)\big[c_{i\omega}(\boldsymbol{S}_i, x_i) + h_{i+1}\big(\phi_{i\omega}(\boldsymbol{S}_i, x_i)\big)\big] \Big\} \quad (3)$$

with $h_{n+1}(\boldsymbol{S}_{n+1}) = 0$ and $h_1(\boldsymbol{S}_1)$ equal to the optimal expected cost. An optimal policy can be obtained by setting $\pi_i(\boldsymbol{S}_i)$ to an optimizing value of x_i in (3) for each state $\boldsymbol{S}_i$.

An SDD corresponding to model (3) contains a node in layer i for every state $\boldsymbol{S}_i$ that can be reached with positive probability under some policy. We let $\boldsymbol{S}_i(u)$ refer to the state associated with node $u \in L_i$. For each node $u \in L_i$ and each control $x_i \in \mathcal{X}(\boldsymbol{S}_i(u))$, there is an outgoing arc to a node associated with state $\phi_{i\omega}(\boldsymbol{S}_i(u), x_i)$ for every outcome ω with positive probability in state S_i. Each of these arcs has label x_i, probability $p_{i\omega}(\boldsymbol{S}_i, x_i)$, and cost $c_{i\omega}(\boldsymbol{S}_i, x_i)$. Several arcs may connect the same two nodes, but they must have different labels. An optimal policy can be computed in the SDD using (1).

In the DP model (3) for the stochastic maximum clique problem, the min operator is max, and each state $\boldsymbol{S}_i$ is the set S_i of vertices currently in the clique. The state corresponding to each node of the SDD is shown in Fig. 2. The transitions and probabilities are as follows:

$$p_{i\omega}(S_i, x_i) = \begin{cases} p_i(S_i), & \text{if } (x_i, \omega) = (1, 1) \\ 1 - p_i(S_i), & \text{if } (x_i, \omega) = (1, 0) \\ \omega, & \text{if } x_i = 0 \end{cases} \quad (4)$$

$$\phi_{i\omega}(S_i, x_i) = \begin{cases} S_i \cup \{i\}, & \text{if } (x_i, \omega) = (1, 1) \\ S_i, & \text{if } \omega = 0 \end{cases} \quad (5)$$

$$c_{i\omega}(S_i, x_i) = \omega \text{ for } x_i = 0, 1 \text{ and all } S_i \quad (6)$$

where $p_i(S_i) = \prod_{j \in S_i} p_{ij}$ and p_{ij} is the probability of edge (i, j).

[2] Following convention in the DP literature, the subscript i in S_i does not index the state but indicates that control x_i is applied in state S_i.

As a second example, we consider a stochastic job sequencing problem with time windows. Each job j has release time r_j and due date d_j. Its processing time is a random variable, and each possible processing time $t_{j\omega}$ corresponds to an outcome $\omega \in \Omega_i$. The set Ω_i can in principle be infinite, but for the purpose of building an SDD, we suppose it is finite. We also suppose that the outcome probabilities are state-independent, so that the probability of a processing time $t_{j\omega}$ is $p_{j\omega}$ regardless of which jobs have been scheduled so far. The objective is to minimize total tardiness, where the tardiness of a job j that starts at time s_j is $(s_j + t_{j\omega} - d_j)^+$, and where we use the notation α^+ for $\max\{0, \alpha\}$. The control x_i in stage i is which job will be ith in the sequence.

In the DP model (3) for the problem, each state $\boldsymbol{S}_i$ is a tuple (S_i, f_i), where S_i is the set of jobs scheduled so far, and f_i is the finish time of the last job scheduled. The recursion is defined by

$$p_{i\omega}\big((S_i, f_i), x_i\big) = p_{x_i\omega} \tag{7}$$

$$\phi_{i\omega}\big((S_i, f_i), x_i\big) = \big(S_i \cup \{x_i\},\ \max\{r_{x_i}, f_i\} + t_{x_i\omega}\big) \tag{8}$$

$$c_{i\omega}\big((S_i, f_i), x_i\big) = \big(\max\{r_{x_i}, f_i\} + t_{x_i\omega} - d_{x_i}\big)^+ \tag{9}$$

The nodes of the an SDD representing the problem correspond to the states (S_i, f_i). Each arc with label x_i leaving state (S_i, f_i) is generated by an outcome ω with positive probability $p_{i_i\omega}$ and has cost $c_{i\omega}((S_i, f_i), x_i)$.

5 Relaxed SDDs

In the deterministic case, a decision diagram $\bar{D}$ relaxes a diagram D when every r–t path of D occurs in $\bar{D}$ with equal or smaller cost. Since policies rather than paths represent solutions in the stochastic case, it is convenient to define a relaxed SDD in terms of its optimal expected cost, rather than the cost of individual paths. This, of course, ensures that a relaxed SDD provides a valid bound on the optimal value of the original SDD. Thus, if two diagrams D and $\bar{D}$ have the same number of layers, and the same possible controls and outcomes, then $\bar{D}$ *relaxes* D if the minimum expected cost $c(D)$ of D is at least the minimum expected cost $c(\bar{D})$ of $\bar{D}$.

Under suitable conditions, a relaxed SDD can be obtained by node merger during top-down compilation, as in the deterministic case. That is, when a given layer i is created from the previous layer, some of the nodes in layer i are merged, so as to reduce the size of the diagram. A larger number of mergers yield a smaller diagram but, in general, a weaker relaxation. When two nodes $u, u' \in L_i$ are merged, the state associated with the resulting node is obtained by applying a merger operation $\boldsymbol{S}_i(u) \oplus \boldsymbol{S}_i(u')$ to the states associated with the merged nodes. Layer $i+1$ is then created on the basis of the resulting states in layer i.

In the maximum clique problem, we can use the merger operation $S_i \oplus S'_i = S_i \cap S'_i$. We will see in the next section that this yields a valid relaxed SDD. Figure 3 displays the relaxed SDD that results from merging two particular nodes in the maximum clique SDD of Fig. 2, namely the nodes in layer 3 with states

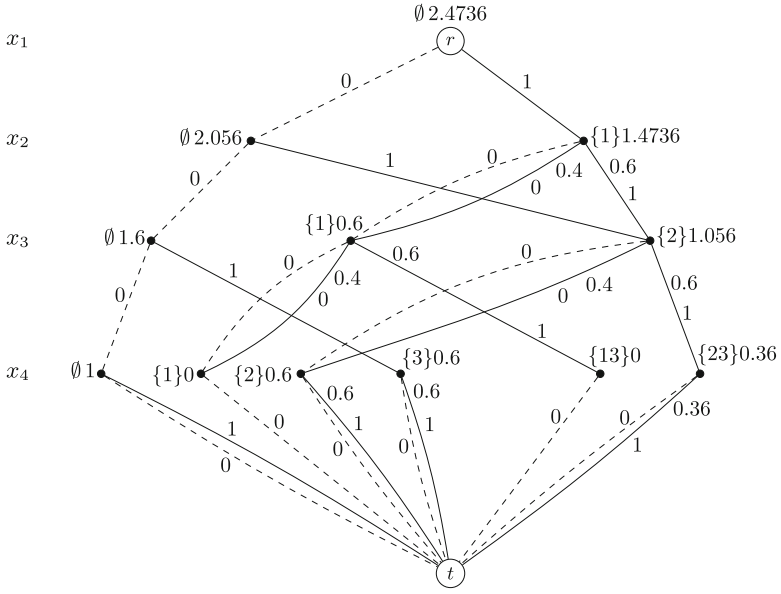

Fig. 3. Relaxed SDD that results from the merger of two nodes in the SDD of Fig 2.

$\{2\}$ and $\{1, 2\}$. The maximum expected cost of the relaxed SDD is 2.4736, which is an upper bound on the optimal expected cost of 2.056 of the original problem.

In the job sequencing problem, we can use the merger operation

$$(S_i, f_i) \oplus (S'_i, f'_i) = \big(S_i \cap S'_i, \min\{f_i, f'_i\}\big) \tag{10}$$

We likewise show in the next section that this creates a valid relaxed SDD.

6 Conditions for Node Merger

We now develop sufficient conditions under which a merger operation yields a relaxed SDD. We again suppose that a concept of relaxation is defined for states. We assume that relaxation has the property that state $\bar{\boldsymbol{S}}_i$ relaxes state $\boldsymbol{S}_i$ only if

$$p_{i\omega}(\boldsymbol{S}_i, x_i)c_{i\omega}(\boldsymbol{S}_i, x_i) \geq p_{i\omega}(\bar{\boldsymbol{S}}_i, x_i)c_{i\omega}(\bar{\boldsymbol{S}}_i, x_i) \tag{11}$$

for any control $x_i \in \mathcal{X}_i$ and any outcome $\omega \in \Omega_i$, and for $i = 1, \ldots, n$. Let the expected immediate cost of a state S_i under control x_i be

$$c_i(\boldsymbol{S}_i, x_i) = \sum_{\omega \in \Omega_i} p_{i\omega}(\boldsymbol{S}_i, x_i)c_{i\omega}(\boldsymbol{S}_i, x_i)$$

Thus, we have from (11)

Lemma 1. *Relaxing a state does not increase its expected immediate cost.*

We will show that node merger yields a valid relaxed SDD if the merger operation $\oplus$ and transition function ϕ satisfy the following conditions for $i = 1, \dots, n$:

(C1) For any two states $\boldsymbol{S}_i$ and $\boldsymbol{S}'_i$, state $\boldsymbol{S}_i \oplus \boldsymbol{S}'_i$ relaxes both $\boldsymbol{S}_i$ and $\boldsymbol{S}'_i$.
(C2) If state $\bar{\boldsymbol{S}}_i$ relaxes state $\boldsymbol{S}_i$, then $\phi_{i\omega}(\bar{\boldsymbol{S}}_i, x_i)$ relaxes $\phi_{i\omega}(\boldsymbol{S}_i, x_i)$ for any $\omega \in \Omega_i$ and any $x_i \in \mathcal{X}_i$.
(C3) If state $\bar{\boldsymbol{S}}_i$ relaxes state $\boldsymbol{S}_i$, then given any control $x_i \in \mathcal{X}_i$ and any set of $\{\eta_\omega \mid \omega \in \Omega_i\}$ of numbers, there is a control $\bar{x}_i \in \mathcal{X}_i$ such that

$$\sum_{\omega\in\Omega_i} p_{i\omega}(\boldsymbol{S}_i, x_i)\big(c_{i\omega}(\boldsymbol{S}_i, x_i) + \eta_\omega\big) \geq \sum_{\omega\in\Omega_i} p_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i)\big(c_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i) + \eta_\omega\big) \quad (12)$$

The argument to follow is simplified if we suppose, without loss of generality, that a given diagram and its relaxation are *fully articulated*. This means that the diagrams contain nodes representing all possible states, even those that cannot be reached with positive probability. This device will allow us to merge nodes by changing only the arc probabilities, with no change in the structure of the diagram. Thus each layer L_{i+1} of a fully articulated diagram contains all states $\boldsymbol{S}_{i+1}$ such that $\boldsymbol{S}_{i+1} = \phi_{i\omega}(\boldsymbol{S}_i(u), x_i)$ for some node $u \in L_i$, outcome $\omega \in \Omega_i$, and control $x_i \in \mathcal{X}_i$, even if the probability $p_{i\omega}(\boldsymbol{S}_i, x_i)$ of reaching $\boldsymbol{S}_{i+1}$ from u is zero for all ω, x_i.

When two nodes $u, u' \in L_{i+1}$ are merged to form node $\hat{u}$, we leave all nodes in place but change only the probabilities on arcs from nodes in layer L_i to u, u', and $\hat{u}$. That is, we transfer the probability on any arc (v, u) to an arc $(v, \hat{u})$ with the same label, and similarly for any arc (v, u'). Thus, if $u = \phi_{i\omega}(v, x_i)$ and $\hat{u} = \phi_{i\hat{\omega}}(v, x_i)$, we set $p_{i\hat{\omega}}(v, x_i) = p_{i\omega}(v, x_i)$ and then set $p_{i\omega}(v, x_i)$ to zero, and similarly for any arc (v, u'). The following lemma is key.

Lemma 2. *If conditions (C2) and (C3) are satisfied, and state $\bar{\boldsymbol{S}}_k$ relaxes state $\boldsymbol{S}_k$, then the optimal cost to go of $\boldsymbol{S}_k$ is at least the optimal cost to go of $\bar{\boldsymbol{S}}_k$.*

Proof. It suffices to show by induction that

$$h_i(\boldsymbol{S}_i) \geq h_i(\bar{\boldsymbol{S}}_i) \quad (13)$$

for $i = n, n-1, \dots, k$. Claim (13) is true for $i = n$ by virtue of Lemma 1, because $h_n(\boldsymbol{S}_n, x_n) = c_n(\boldsymbol{S}_n, x_n)$ for any control $x_n \in \mathcal{X}_n$. We now suppose (13) is true for $i+1$ and show it is true for i. It suffices to show that for any control $x_i \in \mathcal{X}_i$, there is a control $\bar{x}_i \in \mathcal{X}_i$ for which

$$\begin{aligned}\sum_{\omega\in\Omega_i} p_{i\omega}(\boldsymbol{S}_i, x_i)\big[c_{ij}(\boldsymbol{S}_i, x_i) + h_{i+1}(\phi_{i\omega}(\boldsymbol{S}_i, x_i))\big] \\ \geq \sum_{\omega\in\Omega_i} p_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i)\big[c_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i) + h_{i+1}(\phi_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i))\big]\end{aligned} \quad (14)$$

We have from condition (C2) that $\phi_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i)$ relaxes $\phi_{i\omega}(\boldsymbol{S}_i, x_i)$ for all $\omega \in \Omega_i$. This and the induction hypothesis imply $h_{i+1}(\phi_{i\omega}(\boldsymbol{S}_i, x_i)) \geq h_{i+1}(\phi_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i))$ for $\omega \in \Omega_i$. Thus to show (14) it suffices to show

$$\sum_{\omega\in\Omega_i} p_{i\omega}(\boldsymbol{S}_i, x_i)\big[c_{i\omega}(\boldsymbol{S}_i, x_i) + h_{i+1}(\phi_{i\omega}(\boldsymbol{S}_i, x_i))\big] \geq \sum_{\omega\in\Omega_i} p_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i)\big[c_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i) + h_{i+1}(\phi_{i\omega}(\boldsymbol{S}_i, x_i))\big] \tag{15}$$

But (15) follows from condition (C3) by setting $\eta_\omega = h_{i+1}(\phi_{i\omega}(\boldsymbol{S}_i, x_i))$ for each $\omega \in \Omega_i$. □

Theorem 1. *If conditions* (C1)–(C3) *are satisfied, the merger of nodes during compilation of diagram D results in a relaxation of D.*

Proof. Suppose without loss of generality that D is fully articulated, and let $\bar{D}$ be the fully articulated diagram that results from top-down compilation with node merger. Also let $\bar{D}_k$ be the compiled diagram that results when nodes are merged only in levels $1, \ldots, k$. It suffices to show the following inductively for $k = 1, \ldots, n$:

$$c(D) \geq c(\bar{D}_k) \tag{16}$$

This is trivially true for $k = 1$ because $D_1 = \bar{D}_1$. We therefore suppose (16) is true for k and show it is true for $k + 1$. Due to the induction hypothesis, it suffices to show

$$c(\bar{D}_k) \geq c(\bar{D}_{k+1}) \tag{17}$$

Let $\boldsymbol{\pi}^k$ be an optimal policy for $\bar{D}_k$, so that $c(\bar{D}_k) = h_1(\boldsymbol{S}_1, \boldsymbol{\pi}^k)$. We will suppose that only two nodes u, u' in layer $k+1$ of $\bar{D}_k$ are merged to obtain a node $\hat{u}$, since the argument can be repeated for additional mergers. The merger transfers the transition probabilities on arcs to nodes u and u' to the arcs to $\hat{u}$. In addition, by condition (C1), state $\boldsymbol{S}_{k+1}(\hat{u})$ is a relaxation of both $\boldsymbol{S}_{k+1}(u)$ and $\boldsymbol{S}_{k+1}(u')$. Thus, by Lemma 2, the optimal cost to go at $\hat{u}$ is no larger than the optimal cost to go at u and u'. That is,

$$\begin{aligned} h_{k+1}\big(\boldsymbol{S}_{k+1}(\hat{u})\big) &\leq h_{k+1}\big(\boldsymbol{S}_{k+1}(u)\big) \\ h_{k+1}\big(\boldsymbol{S}_{k+1}(\hat{u})\big) &\leq h_{k+1}\big(\boldsymbol{S}_{k+1}(u')\big) \end{aligned} \tag{18}$$

Now due to (18) and the redistribution of transition probabilities into level $k+1$ of $\bar{D}_{k+1}$, the cost-to-go $h_k(v, \boldsymbol{\pi}^k)$ at any node v in layer k under policy $\boldsymbol{\pi}^k$ is no less in $\bar{D}_k$ than in $\bar{D}_{k+1}$. This means that the optimal cost $h(\bar{D}_k)$ of $\bar{D}_k$ is no less than the expected cost $h_1(\boldsymbol{S}_1, \boldsymbol{\pi})$ of $\bar{D}_{k+1}$ under policy $\boldsymbol{\pi}^k$, and is therefore no less than the optimal cost of $\bar{D}_{k+1}$. This establishes (17), as desired. □

We can now verify that the merger operations used earlier in the maximum claim and job sequencing problems yield valid relaxed SDDs.

Corollary 1. *The merger operation $S_i \oplus S'_i = S_i \cap S'_i$ results in a relaxed SDD for the maximum clique problem.*

Proof. In this problem, a state $\boldsymbol{S}_i$ is the set S_i of vertices already selected. We will say that $\bar{S}_i$ relaxes S_i when $\bar{S}_i \subseteq S_i$. We first observe that the definitional

requirement (11) for a relaxation is satisfied, by substituting the values given in (4)–(6) into (11) for the four cases $(x_i, \omega) = (1,1), (1,0), (0.1), (0.0)$.

Now, due to Theorem 1, it suffices to show that conditions (C1)–(C3) are satisfied. Condition (C1) is obviously satisfied, since $S_i \cap S'_i \subseteq S_i$ and $S_i \cap S'_i \subseteq S'_i$. To show (C2), we suppose that $\bar{S}_i \subseteq \bar{S}_i$. Then $\phi_{i0}(\bar{S}_i) = \bar{S}_i \subseteq S_i = \phi_{i0}(S_i)$ and $\phi_{i1}(\bar{S}_i) = \bar{S}_i \cup \{i\} \subseteq S_i \cup \{i\} = \phi_{i1}(S_i)$. Thus, $\phi_{i\omega}(\bar{S}_i)$ relaxes $\phi_{i\omega}(S_i)$ for $\omega = 0, 1$.

We now show (C3) with the sense of the inequality reversed, since we are maximizing rather than minimizing. We note that

$$\sum_{\omega \in \Omega_i} p_{i\omega}(\boldsymbol{S}_i, x_i)\big(c_{i\omega}(\boldsymbol{S}_i, x_i) + \eta_\omega\big) = \begin{cases} \eta_0, & \text{if } x_i = 0 \\ \big(1 - p_i(S_i)\big)\eta_0 + p_i(S_i)(1 + \eta_1), & \text{if } x_i = 1 \end{cases}$$

$$\sum_{\omega \in \Omega_i} p_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i)\big(c_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i) + \eta_j\big) = \begin{cases} \eta_0, & \text{if } \bar{x}_i = 0 \\ \big(1 - p_i(\bar{S}_i)\big)\eta_0 + p_i(\bar{S}_i)(1 + \eta_1), & \text{if } \bar{x}_i = 1 \end{cases}$$

If $x_i = 0$, we select the control $\bar{x}_i = 0$, and (12) becomes $\eta_0 \geq \eta_0$, which is obviously satisfied. If $x_i = 1$, we consider two cases. We first suppose that $\eta_0 \geq 1 + \eta_1$, in which case we select $\bar{x}_i = 0$. Then (12) becomes

$$\big(1 - p_i(S_i)\big)\eta_0 + p_i(S_i)(1 + \eta_1) \leq \eta_0$$

which simplifies to $\eta_0 \leq 1 + \eta_1$ or $\eta_0 \leq \eta_0$, and (12) is therefore satisfied. We now suppose $\eta_0 < 1 + \eta_1$, in which case we select $\bar{x}_i = 1$. Then (12) becomes

$$\big(1 - p_i(S_i)\big)\eta_0 + p_i(S_i)(1 + \eta_1) \leq \big(1 - p_i(\bar{S}_i)\big)\eta_0 + p_i(\bar{S}_i)(1 + \eta_1)$$

Since $p_i(\bar{S}_i) \geq p_i(S_i)$, this simplifies to $\eta_0 \leq 1 + \eta_1$ or $0 \leq 0$, and (12) is again satisfied. □

The conditions for a valid relaxation simplify when transition probabilities are state-independent. In this case, it suffices to satisfy (C1) and (C2).

Corollary 2. *If conditions* (C1) *and* (C2) *are satisfied, and transition probabilities are state independent, then merger of nodes during compilation results in a relaxed SDD.*

Proof. Due to Theorem 1, it suffices to show that condition (C3) is satisfied as well as (C1) and (C2). In fact, we can show that for any control $x_i \in \mathcal{X}_i$ and any set $\{\eta_\omega \mid \omega \in \Omega_i\}$, (12) holds for $\bar{x}_i = x_i$. Since transition probabilities are state independent, we have $p_{i\omega}(\boldsymbol{S}_i, x_i) = p_{i\omega}(\bar{\boldsymbol{S}}_i, x_i)$, and (12) becomes

$$\sum_{\omega \in \Omega_i} p_{i\omega}(\boldsymbol{S}_i, x_i) c_{i\omega}(\boldsymbol{S}_i, x_i) + p_{i\omega}(\boldsymbol{S}_i, x_i)\eta_\omega \geq \sum_{\omega \in \Omega_i} p_{i\omega}(\bar{\boldsymbol{S}}_i, x_i) c_{i\omega}(\bar{\boldsymbol{S}}_i, x_i) + p_{i\omega}(\boldsymbol{S}_i, x_i)\eta_\omega$$

This follows immediately from (11), and (C3) is therefore satisfied. □

Corollary 3. *The merger operation* (10) *results in a relaxed SDD for the stochastic job sequencing problem.*

Proof. In this problem, we say that a state $(\bar{S}_i, \bar{f}_i)$ relaxes a state (S_i, f_i) when $\bar{S}_i \subseteq S_i$ and $\bar{f}_i \leq f_i$. To show that this relaxation satisfies (11), it suffices to show

$$c_{i\omega}\big((S_i, f_i), x_i\big) \geq c_{i\omega}\big((\bar{S}_i, \bar{f}_i), x_i\big)$$

for any x_i and ω, because the transition probabilities are state-independent. But due to (9), this becomes

$$\big(\max\{r_{x_i}, f_i\} + t_{x_i\omega} - d_{x_i}\big)^+ \geq \big(\max\{r_{x_i}, \bar{f}_i\} + t_{x_i\omega} - d_{x_i}\big)^+$$

which holds because $f_i \geq \bar{f}_i$. Conditions (C1) and (C2) can be checked in a similar manner, and the claim follows due to Corollary 2. □

7 Computational Evaluation

The primary challenge in computational evaluation of SDD bounds is finding nontrivial instances that can be solved to optimality, so that the tightness of bounds can be assessed. Unfortunately, the DIMACS instances of the maximum clique problem are much harder to solve as a stochastic problem, and nearly all are intractable. We were able to solve only five instances, three of which are too trivial for evaluation of bounds, and one of which ran for almost 24 h. We therefore generated a number of random instances that are calibrated in size to be both tractable and nontrivial. Since state-space enumeration is the only currently available method for optimal solution, we solved the instances to optimality by generating an exact SDD, which is essentially state-space enumeration with an intelligent variable ordering heuristic. We performed tests on both random and DIMACS instances, the latter mostly without comparison with optimal values.

We generated graphs for random instances by selecting each possible edge with a specified probability (which determines the average density of the resulting graphs). We then randomized the graphs by assigning each edge (i, j) probability $p_{ij} = 0.5 + 17ij \bmod(50)/100$. Thus, all probabilities were drawn from the interval $[0.5, 1]$. Probabilities were assigned to edges in DIMACS instances in the same manner.

The size of the relaxed SDDs is controlled by placing an upper bound on the width (the maximum number of node in a layer). The SDDs are compiled using the same heuristics as in [7]. In particular, the least attractive nodes on a given layer are merged until the desired width is obtained, where attractiveness is measured by the length of the longest path from the root in the deterministic problem. The rationale for this is that unattractive nodes are less likely to be part of an optimal solution and can therefore be merged without affecting the value of that solution. We do not check whether the state resulting from a merger already occurs at a node in the layer, because such a check is quite time consuming, although this can result in slightly weaker bounds.

Figure 4 displays the results for the random instances. Each plot shows the bounds that result from three random instances of different sizes. The bounds are plotted against the time invested in compiling relaxed SDDs of various widths

Table 1. DIMACS Instances Tested

Instance	Vertices	Density	Instance	Vertices	Density
brock200_1	200	0.7417	hamming6-2	64	0.8906
cfat200-1	200	0.0767	johnson8-4-4	70	0.7571
cfat500-1	500	0.0357	keller4	171	0.6453
c125.9	125	0.8913	p_hat300-1	300	0.2430
DSJC500_5	500	0.5010	san200_0.7_1	200	0.6965
gen200_p0.9_44	200	0.8955	sanr_0.7	200	0.6934

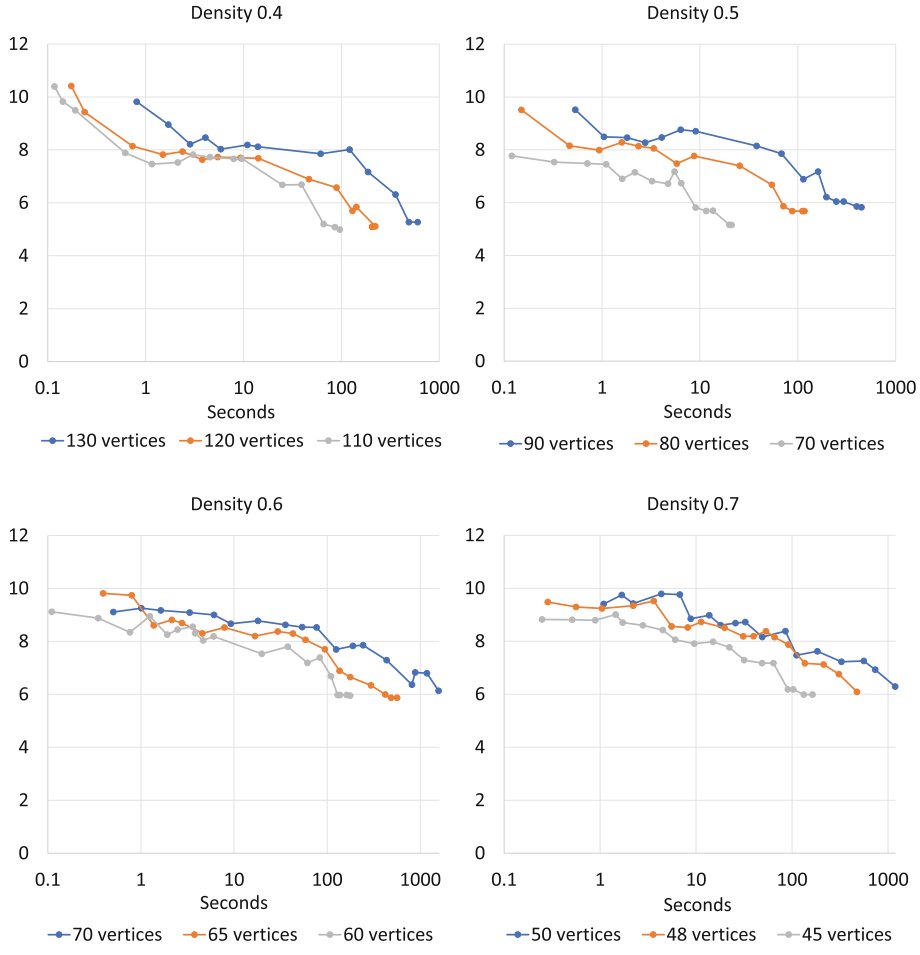

Fig. 4. Bound vs. time investment for random maximum clique instances. The smallest (rightmost) bound shown for each instance is the optimal value.

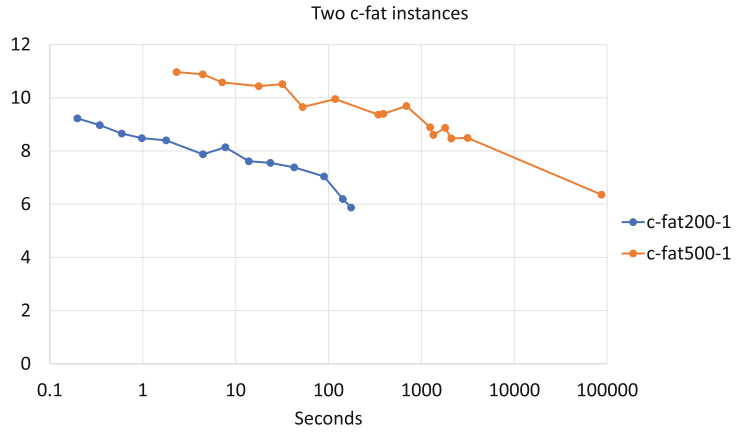

Fig. 5. Bound vs. time investment for two DIMACS instances solved to optimality.

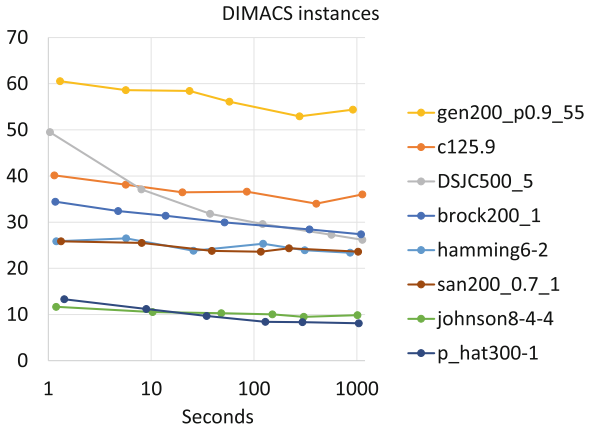

Fig. 6. Bound vs. time investment for various DIMACS instances, none of which are solved to optimality.

and computing the optimal expected clique size for each. Each point results from setting the maximum width to a certain value. The bound is plotted against the resulting computation time rather than the max width, because this better indicates the cost of obtaining the bound.[3] Nearly all of the computational cost is incurred by generating and merging nodes. The right-most data point for each instance represents an optimal solution. The plots indicate that the quality of the bound degrades rather modestly as the time investment ranges over about three orders of magnitude. The relationship is not entirely monotone, as would

[3] Detailed computational results, including specified max widths, are posted at public.tepper.cmu.edu/jnh/CPAIOR2022data.pdf.

be expected because the quality of the bound depends on the chance effects of the merger heuristic.

Tables 1 lists the DIMACS instances tested. Figure 5 shows bound-vs.-time plots for the two nontrivial DIMACS instances solved to optimality. The bound degradation is again gradual, and it is almost the same over a comparable time span for the large, difficult instance as for the small, easy instance. Figure 6 contains plots for various additional DIMACS instances, none of which could be solved to optimality. With one exception, they show a similar pattern of gradual and roughly logarithmic bound improvement as the time investment increases.

8 Conclusion

We introduced a concept of stochastic decision diagrams (SDDs) and showed how they can represent discrete stochastic dynamic programming (DP) problems. In particular, we indicated how state-dependent policies can be understood in a decision diagram context. Due to the difficulty and importance of deriving valid bounds for DP models, we focused on extending relaxation technology that has been developed for deterministic decision diagrams to the stochastic case, so that these bounds can be available for DP applications. We established sufficient conditions under which a node merger operation applied during top-down compilation yields valid relaxed SDDs. These provide optimization bounds whose quality can be adjusted at will by controlling the size of the relaxed SDD. As corollaries, we showed that simple node merger rules for stochastic maximum clique and job sequencing problems satisfy these conditions.

To test the performance of SDD bounding in practice, we computed bounds for random and benchmark instances of the stochastic maximum clique problem. We found that the bound quality degrades only gradually as the SDD size (and the time investment) is reduced. The time-quality relationship is roughly logarithmic in most cases, which allows one to estimate how much the bound would improve with further time investment after computing bounds for a few sample SDD widths.

A number of research directions can be pursued at this point, but two are particularly interesting. One is to extend to stochastic diagrams the decision-diagram-based branch-and-bound procedure introduced in [8]. This would provide a novel alternative to traditional state-space enumeration as an exact solution method for stochastic DP problems of moderate size.

A second direction is to investigate SDD-derived bounds on costs-to-go in approximate stochastic DP. The cost-to-go at any state in any stage of the DP recursion can be bounded by building an SDD that begins with that state at its root node. The solution of the DP problem would still be calculated on the basis of estimated costs-to-go, as in traditional approximate DP. Then one would compute the expected cost of this same policy using SDD-based bounds on the cost-to-go rather than estimates of the cost-to-go. This would provide a bound on the quality of the policy obtained by approximate DP.

References

1. Andersen, H.R., Hadzic, T., Hooker, J.N., Tiedemann, P.: A constraint store based on multivalued decision diagrams. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 118–132. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74970-7_11
2. Baldacci, R., Mingozzi, A., Roberti, R.: New state-space relaxations for solving the traveling salesman problem with time windows. INFORMS J. Comput. **24**(3), 356–371 (2012)
3. Bellman, R.: The theory of dynamic programming. Bull. Am. Math. Soc. **60**, 503–516 (1954)
4. Bellman, R.: Dynamic Programming. Priceton University Press, Princeton, NJ (1957)
5. Bergman, D., Ciré, A.A.: Discrete nonlinear optimization by state-space decompositions. Manag. Sci. **64**, 4700–4720 (2018)
6. Bergman, D., Cire, A.A., van Hoeve, W.-J.: Improved constraint propagation via lagrangian decomposition. In: Pesant, G. (ed.) CP 2015. LNCS, vol. 9255, pp. 30–38. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23219-5_3
7. Bergman, D., Ciré, A.A., van Hoeve, W.J., Hooker, J.N.: Optimization bounds from binary decision diagrams. INFORMS J. Comput. **26**, 253–268 (2013)
8. Bergman, D., Ciré, A.A., van Hoeve, W.J., Hooker, J.N.: Discrete optimization with binary decision diagrams. INFORMS J. Comput. **28**, 47–66 (2014)
9. Bergman, D., Cire, A.A., van Hoeve, W.-J., Hooker, J.: MDD-based constraint programming. In: Decision Diagrams for Optimization. AIFTA, pp. 157–181. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42849-9_9
10. Bergman, D., van Hoeve, W.-J., Hooker, J.N.: Manipulating MDD relaxations for combinatorial optimization. In: Achterberg, T., Beck, J.C. (eds.) CPAIOR 2011. LNCS, vol. 6697, pp. 20–35. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21311-3_5
11. Bertele, U., Brioschi, F.: Nonserial Dynamic Programming. Academic Press, New York (1972)
12. Bertsekas, D.P.: Dynamic programming and optimal control: approximate dynamic programming, vol. 2, 4th edn. Athena Scientific, Nashua, NH (2012)
13. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 1, 4th edn. Athena Scientific, Nashua, NH (2017)
14. Castro, M.P., Ciré, A.A., Beck, J.C.: An MDD-based Lagrangian approach to the multicommodity pickup-and-delivery TSP. INFORMS J. Comput. **32**, 263–278 (2020)
15. Christofides, N., Mingozzi, A., Toth, P.: State-space relaxation procedures for the computation of bounds to routing problems. Networks **11**(2), 145–164 (1981)
16. Ciré, A.A., van Hoeve, W.J.: Multivalued decision diagrams for sequencing problems. Oper. Res. **61**, 1411–1428 (2013)
17. Ciré, A.A., van Hoeve, W.J.: MDD propagation for disjunctive scheduling. In: Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS) (2012). AAAI Press
18. Gentzel, R., Michel, L., van Hoeve, W.-J.: HADDOCK: A language and architecture for decision diagram compilation. In: Simonis, H. (ed.) CP 2020. LNCS, vol. 12333, pp. 531–547. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58475-7_31

19. González, J.E., Ciré, A.A., Lodi, A., Rousseau, L.M.: BDD-based optimization for the quadratic stable set problem. Discrete Optim. **44**, 100610 (2020)
20. González, J.E., Cire, A.A., Lodi, A., Rousseau, L.-M.: Integrated integer programming and decision diagram search tree with an application to the maximum independent set problem. Constraints **25**(1), 23–46 (2020). https://doi.org/10.1007/s10601-019-09306-w
21. Hadžić, T., Hooker, J.N.: Discrete global optimization with binary decision diagrams. In: GICOLAG 2006. Vienna, Austria, December 2006
22. Hadžić, T., Hooker, J.N.: Cost-bounded binary decision diagrams for 0-1 programming. In: Van Hentenryck, P., Wolsey, L. (eds.) CPAIOR 2007. LNCS, vol. 4510, pp. 84–98. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72397-4_7
23. Hoda, S., van Hoeve, W.-J., Hooker, J.N.: A systematic approach to MDD-based constraint programming. In: Cohen, D. (ed.) CP 2010. LNCS, vol. 6308, pp. 266–280. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15396-9_23
24. Hooker, J.N.: Decision diagrams and dynamic programming. In: Gomes, C., Sellmann, M. (eds.) CPAIOR 2013. LNCS, vol. 7874, pp. 94–110. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38171-3_7
25. Hooker, J.N.: Job sequencing bounds from decision diagrams. In: Beck, J.C. (ed.) CP 2017. LNCS, vol. 10416, pp. 565–578. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66158-2_36
26. Hooker, J.N.: Improved job sequencing bounds from decision diagrams. In: Schiex, T., de Givry, S. (eds.) CP 2019. LNCS, vol. 11802, pp. 268–283. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30048-7_16
27. Mingozzi, A.: State space relaxation and search strategies in dynamic programming. In: Koenig, S., Holte, R.C. (eds.) SARA 2002. LNCS (LNAI), vol. 2371, pp. 51–51. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45622-8_4
28. O'Neil, R.J., Hoffman, K.: Decision diagrams for solving traveling salesman problems with pickup and delivery in real time. Oper. Res. Lett. **47**, 197–201 (2019)
29. Powell, W.B.: Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd edn. Wiley-Interscience, Hoboken (2011)
30. Righini, G., Salani, M.: New dynamic programming algorithms for the resource constrained shortest path problem. Networks **51**, 155–170 (2008)
31. Tjandraatmadja, C., van Hoeve, W.-J.: Incorporating bounds from decision diagrams into integer programming. Math. Program. Comput. **13**(2), 225–256 (2020). https://doi.org/10.1007/s12532-020-00191-6