Towards Improving Merging Heuristics for Binary Decision Diagrams

Nikolaus Frohner(✉) and Günther R. Raidl

Institute of Logic and Computation, TU Wien, Vienna, Austria
{nfrohner,raidl}@ac.tuwien.ac.at

Abstract. Over the last years, binary decision diagrams (BDDs) have become a powerful tool in the field of combinatorial optimization. They are directed acyclic multigraphs and represent the solution space of binary optimization problems in a recursive way. During their construction, merging of nodes in this multigraph is applied to keep the size within polynomial bounds resulting in a discrete relaxation of the original problem. The longest path length through this diagram corresponds then to an upper bound of the optimal objective value. The algorithm deciding which nodes to merge is called a *merging heuristic*. A commonly used heuristic for layer-wise construction is *minimum longest path length* (minLP) which sorts the nodes in a layer descending by the currently longest path length to them and subsequently merges the worst ranked nodes to reduce the width of a layer. A shortcoming of this approach is that it neglects the (dis-)similarity between states it merges, which we assume to have negative impact on the quality of the finally obtained bound. By means of a simple tie breaking procedure, we show a way to incorporate the similarity of states into minLP using different distance functions to improve dual bounds for the maximum independent set problem (MISP) and the set cover problem (SCP), providing empirical evidence for our assumption. Furthermore, we extend this procedure by applying similarity-based node merging also to nodes with close but not necessarily identical longest path values. This turns out to be beneficial for weighted problems where ties are substantially less likely to occur. We evaluate the method on the weighted MISP and tune parameters that control as to when to apply similarity-based node merging.

Keywords: Binary decision diagrams · Top-down construction · Merging heuristic · State similarity · Tie breaking

1 Introduction

In the last decade, decision diagrams (DDs) have emerged as a new tool in the field of combinatorial optimization. Originally, they were conceived by Lee [10] in circuit design as a compact representation for binary functions. In the optimization context, they were introduced by Hadzic and Hooker [7] as a tool for post-optimality analysis. Since then, DDs have been used to obtain strong dual

N. F. Matsatsinis et al. (Eds.): LION 13 2019, LNCS 11968, pp. 30–45, 2020.
https://doi.org/10.1007/978-3-030-38629-0_3

bounds by means of a new form of discrete relaxation [6], as constraint stores for advanced constraint propagation in constraint programming [7], for obtaining promising heuristic solutions [4], and for a new branching scheme leading to a general purpose branch-and-bound framework [3]. For a comprehensive book on DDs for optimization, see [2].

A DD for a given problem is a directed acyclic graph $G = (V, A)$ with node set V and arc set A containing dedicated root and target nodes $r, t \in V$. An exact DD represents all feasible solutions of the underlying problem in the sense that there is a one-to-one correspondence between r–t paths and feasible solutions. Therefore exact DDs for hard problems typically have exponential size. In the layer-wise, top-down construction of relaxed DDs, one restricts the size by merging nodes whenever a layer would exceed a specified width. Merging is done in such a way that no feasible solution is lost, but new paths, corresponding to infeasible solutions, may emerge. Assuming maximization, a longest path from the root to the terminal node represents a solution that is usually infeasible for the original problem but yields a dual bound. The tightness of this bound is determined by the maximum width of the layers, the ordering of the decision variables [1] and the merging heuristic, i.e., the selection of the nodes that are merged. Algorithms building on a DD can strongly benefit from a stronger bound or a more compact DD that yields the same bound. The latter holds in particular when the once constructed DD is then traversed many times as in bound strengthening schemes like the value enumeration method [6] that incrementally strengthens integral bounds when there is no path with the current bound that corresponds to a feasible solution.

In this paper, we show how to improve the commonly used merging heuristic *minimum longest path* (minLP) for two benchmark problems, namely the maximum independent set (MISP) and the set cover problem (SCP). Section 2 reviews related work. In Sect. 3, we formally introduce binary decision diagrams (BDDs) based on dynamic programming formulations and provide the concrete modeling of the MISP and SCP. In Sect. 4, we introduce a state similarity-based tie breaking procedure for the minLP merging heuristic with the aim to improve the quality of obtained dual bounds. The approach is specifically instantiated for MISP and SCP. We then generalize the method by applying the similarity-based merging not just in case of ties but already when longest path values of nodes are sufficiently close. This turns out to be particularly meaningful in case of the weighted MISP, since there ties are substantially less likely to occur. More generally, for other problems we also provide suggestions on how to construct meaningful *merging distance functions*. In Sect. 5 we present our computational study, where the effectiveness of our tie breaking approach on compact BDDs with small widths for the MISP, weighted MISP, and SCP is demonstrated. We conclude in Sect. 6.

2 Related Work

Our work builds upon the classic top-down construction method of BDDs as described by Bergman et al. in [5] and [6], whose results we also use as a baseline

for the MISP and SCP in our computational study. In limited-width BDDs, nodes are merged to achieve a discrete relaxation of the solution space; the selection of which nodes to merge is called *merging heuristic*, see Sect. 4. The pairwise minLP merging heuristic was introduced in [6], in its bulk form in [5]. The size of a BDD is crucially determined by the order in which the decision variables are processed as elaborated on in [1]. The *minState* variable ordering heuristic selects in each layer dynamically the next decision variable for which the least successor nodes can be derived to aim for keeping the BDD small in a greedy way. Together, the minState variable ordering heuristic and the minLP merging heuristic provide strong bounds for the MISP on random and DIMACS graphs, as presented in [5]. The possible impact of state (dis-)similarity is already addressed and a minimum distance pairwise merging heuristic is suggested in [6], on which we focus in this paper in Sect. 4. In [9], a clustering algorithm is used to partition DD nodes into approximate equivalence classes for solving a multi-dimensional bin packing problem.

3 Binary Decision Diagrams (BDDs)

We consider a combinatorial optimization problem (COP) $\mathcal{C} = \langle S, f \rangle$, where S is the finite search space and $f\colon S \to \mathbb{R}$ the objective function to be maximized. Every element $x \in S$ is represented by an assignment of values to n binary decision variables $x_i \in \{0,1\},\ i = 1,\dots,n$. Hence, $S \subset \{0,1\}^n$ and $f\colon \{0,1\}^n \to \mathbb{R}$. The goal is to find an optimal solution x^*, i.e., for which the objective value $z^* = f(x^*) \geq f(x')\ \forall x' \in S$:

$$z^* = \max_{x \in S} f(x) \tag{1}$$

We restrict f to be a separable function of the decision variables $f(x) = \sum_{i=1}^{n} f_i(x_i)$ which allows us to state the COP in a recursive formulation. For a well-defined ordering in a recursion, a variable ordering $\pi\colon \{1,\dots,n\} \to \{1,\dots,n\}$, with π being bijective, is assumed. A partial assignment of the decision variables of length k, under the ordering π is then defined by an ordered tuple $(d_{\pi_1},\dots,d_{\pi_k}) \in \{0,1\}^k, k \in \{0,\dots,n\}$, where $k = 0$ corresponds to an empty assignment.

Definition 1 (State). *A state $s_i \in \mathcal{S}$ is a mapping from an i-partial assignment. It determines the subset $F_{s_i} \subseteq \{0,1\}^{n-i}$ of feasible decisions for remaining variables $x_{\pi(i+1)},\dots,x_{\pi(n)}$, the* ***feasible completions*** *of the current partial assignment. If two partial assignments have the same state, they have the same feasible completions.*

The representation of a state needs to be concretely defined for the problem at hand, for example by means of sets or reals. This admits a recursive enumeration of the state space $\mathcal{S}^{n+1} \ni (s_0,\dots,s_n)$ corresponding to the search space S by defining a state transition function:

$$\tau\colon \{0,1\} \times \mathcal{S} \to \mathcal{S} \tag{2}$$

$$(d, s_i) \mapsto \tau(d, s_i) = s_{i+1} \tag{3}$$

We can now formulate our maximization problem recursively over the states via Bellman equations $\forall i \in \{0, \dots, n-1\}$:

$$z^*(s_i) = \max_{d \in \{0,1\}} \{f_{\pi_i}(d) + z^*(\tau(d, s_i)) \mid d \, \exists c \in F_{s_i} \colon c = (d, \dots)\} \tag{4}$$

$$z^*(s_n) = 0 \tag{5}$$

If for a given s_i there exists a feasible completion $c \in F_{s_i}$ for which we can set the next decision variable $x_{\pi_{i+1}}$ to d, i.e., $\exists c \in F_{s_i} \colon c = (d, \dots)$, we say that the state *admits* a d-transition. The root state s_0 corresponds to an empty partial assignment, s_1 to when the first variable x_{π_1} has been assigned, and so forth. Clearly, $F_{s_0} = S$ and $F_{s_n} = \emptyset$.

A binary decision diagram (BDD) in our context is a directed acyclic layered multigraph with layers $L, |L| = n + 1$ and represents this state space enumeration graphically. Layer 0 contains only the root node r representing the root state $s(r) = s_0$ and layer n the terminal node t representing the terminal state $s(t) = s_n$. Each node u in layer l is thus associated with a state $s(u)$. If $s(u) = s(u')$ for nodes u, u' in a given layer, they admit by definition the same feasible completions and can therefore be superimposed to reduce the size of the BDD. Except for the terminal node, each node u has a d-labeled outgoing arc $a = (u, v)$ for each d admissible by $F_{s(u)}$, representing the possible decisions at state $s(u)$. $source(a) = u$ is called the source (node) of the arc and $target(a) = v$ the target (node) respectively. The arcs point downwards, the layer of the target must always be greater than the one of the source.

Every arc receives a label $d(a) \in \{0, 1\}$ to encode a binary decision. If a path starts at the root node and finally leads to some node v, which we denote by p_{rv}, this corresponds to a k-partial assignment $(d((r, u_1)), \dots, d((u_{k-1}, v)))$, where $d((u, v))$ is the aforementioned label of arc (u, v). Every arc is assigned a weight $f_{\pi_i}(d)$, contributing to the length of paths going trough the decision diagram, for instance $f_{\pi(i)}(d) = c_{\pi_i} d$ when we are given constant objective function contributions c_{π_i} for each decision variables $x_{\pi(i)}$ set to one. In exact BDDs, there is by construction a one-to-one mapping between paths p_{rt} and feasible solutions S. For maximization problems, paths of longest length correspond to its optimal solutions. In general, the exact decision diagrams grow exponential in size in the number of decision variables. The focus in this paper lies on limited-width, *relaxed* DDs, where layers have a maximum number β of nodes to keep the DD size bounded by $\beta|L|$ nodes. The contained paths represent a *superset* of the search space S and, thus, a discrete relaxation of the original problem. This is achieved by also superimposing nodes that have *different* states, which is called *merging*.

Definition 2 (Merging of nodes). *When nodes u, v are merged into a node w, all incoming arcs of u, v are redirected to the new node w and the states $s(u), s(v)$ are merged into $s(w)$ in a way that no feasible paths, i.e., solutions in the search space are lost. Therefore, $F_{s(w)} \supset F_{s(u)} \cup F_{s(v)}$.*

The length of a longest path in a relaxed BDD is an upper bound on the optimal objective value to the original problem. Our first specific problem we

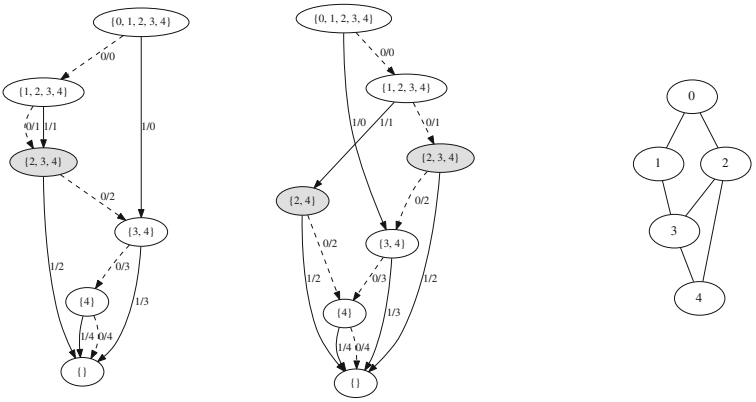

Fig. 1. Two relaxed BDDs for a simple graph instance on the right. Left with maximum width $\beta = 1$, in the center with $\beta = 2$, both having the same longest path length of 2 with optimal solutions of zero-indexed vertices $\{\{0, 3\}, \{0, 4\}, \{1, 2\}, \{1, 4\}\}$.

consider is the maximum independent set problem (MISP). It is defined on an undirected simple graph $G = (V, E)$ as finding a maximum subset of nodes $I \subset V$, s.t. no pair of nodes in I are adjacent. A proper state s_i is the subset of the vertices for which no decision has been yet made and for which no neighbor has been selected so far. The transition function is

$$\tau : \{0, 1\} \times 2^V \to 2^V \tag{6}$$

$$(0, s_i) \mapsto s_{i+1} = \tau(0, s_i) = s_i - \{\pi_i\} \tag{7}$$

$$(1, s_i) \mapsto s_{i+1} = \tau(1, s_i) = s_i - \{\pi_i\} - N(\pi_i) \tag{8}$$

where $N(\pi_i) \subset V$ is the neighborhood of the i-th considered vertex π_i. The root state s_0 is V, the terminal state $\emptyset$. A natural merging operator $\oplus$ of k states is given by the set union:

$$\oplus(\{u_1, \ldots, u_k\}) \mapsto w \colon s(w) = \bigcup_{j=1}^{k} s(u_j) \tag{9}$$

Two examplary BDDs for a simple MISP instance of width $\beta = 1$ and $\beta = 2$ are depicted in Fig. 1. For each arc the weight and the corresponding decision variable is shown. The label is indicated by a dotted arc for a 0-transition and a solid arc for a 1-transition. When reducing the maximum width from 2 to 1, we see that a merging is applied in the second layer of states $\{2, 4\}$ and $\{2, 3, 4\}$.

As a second fundamental problem, we consider the classical set cover problem (SCP). Given a universe $\mathcal{U}$ and a set of sets $\mathcal{S}$ with $\mathcal{S} \ni S \subset \mathcal{U}$ and $\bigcup_{S \in \mathcal{S}} S = \mathcal{U}$, we seek to find a $\mathcal{S}^* \subset \mathcal{S}$ with minimum cardinality so that $\bigcup_{S \in \mathcal{S}^*} S = \mathcal{U}$, i.e., a minimum set covering. A proper state s_i is the set of elements that still have to be covered. To ensure that all paths are feasible, a set j has to be selected (i.e.,

```
1  X ← {1, ..., n}, P ← {r};
2  for l ← 1 to n do
3  |  π_l ← next-decision-variable(l, P, X);
4  |  X ← X − π_l;
5  |  L_l ← P' ⊂ P for which x_{π_l} can bet set to 1;
6  |  while |L_l| > β do
7  |  |  L_l ← merge-nodes(L_l, l);
8  |  end
9  |  foreach u ∈ L_l do
10 |  |  foreach d ∈ {0, 1} do
11 |  |  |  if s(u) admits d-transition then
12 |  |  |  |  create v_d;
13 |  |  |  |  s(v_d) = τ(d, s(u));
14 |  |  |  |  create arc (u, v_d) with label d and weight f_{π(l)}(d);
15 |  |  |  end
16 |  |  end
17 |  end
18 end
```

Algorithm 1. Relaxed limited-width layered binary decision diagram construction algorithm, adapted from [3, p. 12].

its decision variable set to 1) if there exists an element in s_i that can only be covered by selecting j, since all other possible decision variables have been set to 0. A natural merging operator $\oplus$ of k states is given by the set intersection:

$$\oplus\left(\{u_1, \ldots, u_k\}\right) \mapsto w \colon s(w) = \bigcap_{j=1}^{k} s(u_j) \tag{10}$$

Throughout this paper, we focus on the top-down layer-wise construction algorithm [5] for relaxed binary decision diagrams with maximum width β as described in Algorithm 1.

It facilitates zero-suppressing long arcs, a dynamic variable ordering by the function next-decision-variable and merging of nodes by the function merge-nodes. As concrete variable ordering heuristic, we consider here minState [5] which selects as next decision variable the one that yields the least number of one-transitions from the current nodes for the next layer. A simple, yet effective and commonly used merging heuristic is minLP [5], which sorts the nodes u in a layer by the longest path lengths from the root node to them, denoted by $z^{\mathrm{lp}}(u)$, in decreasing order and merges the tail into one node so that the resulting layer is of maximum width β, see Algorithm 2. In the minLP approaches described in the literature so far, to the best of our knowledge, no tie breaking mechanism for the sorting is explicitly specified which gives rise to the next section.

```
Function merge-nodes(L_l, l)
    T ← sorted nodes of L_l in decreasing order of z^lp(u);
    T' ← T without first β − 1 nodes of T;
    L_l = L_l \ T';
    w ← ⊕T;
    if ∃w' ∈ L_l | s(w) = s(w') then
        w' ← w' ⊕ w;
    else
        L_l = L_l ∪ {w};
    end
```

Algorithm 2. minLP merging heuristic in the bulk variant where all nodes to be merged are merged within one step into one single node.

4 State Similarity

We consider two different merging heuristic patterns: *pairwise* merging and *bulk* merging. Both face a layer l with a set of nodes L_l where $|L_l|$ exceeds the maximum width β. Pairwise merging is a form of *iterative* merging where pairs of nodes are selected and merged until the desired layer width has been reached. In contrast, bulk merging selects and merges the necessary number of nodes in a single iteration. The bulk minLP merging heuristic as introduced in the last section in Algorithm 2 sorts the nodes in a layer according to the longest path length to them and merges the last $|L_l| - \beta + 1$ nodes into one. It generalizes to *rank* based merging, which sorts nodes in a layer according to some criterion and merges the required number of tail nodes. If the criterion can be calculated easily, a clear benefit is the $\mathcal{O}(|L_l| \log |L_l|)$ runtime complexity, whereas pairwise mergings needs in general at least $\mathcal{O}(|L_l|^2)$ time.

The rationale behind minLP is to consider nodes with smaller $z^{\text{lp}}(u)$ less promising to be part of an overall longest path in the completed DD and therefore less critical when merged in order to finally obtain a tight upper bound. This strategy is supported by the minState variable ordering heuristic, which keeps the size of the layers before merging as small as possible, therefore reducing the number of nodes that need to be merged.

A shortcoming of this approach is that it neglects information that could be obtained from the states of the nodes themselves, in particular the similarity between states. Intuitively, merging similar states will usually lead to less new paths corresponding to infeasible solutions than merging very different states. If two states are comparable, for instance by the subset relation for sets or the total order for reals, we denote $s(u_1) \succeq s(u_2)$ when $s(u_1)$ is greater than $s(u_2)$. If $s(u_1) \succeq s(u_2)$, then $F_{s(u_1)} \supseteq F_{s(u_2)}$. One way merging of nodes u, v introduces infeasible solutions is by increasing the size of feasible completions $F_{s(w)} \supset F_{s(u)} \cup F_{s(v)}$, which gives rise to a definition for a meaningful distance function:

Definition 3 (Merging distance between two nodes). *A merging distance between two nodes u, v is a non-negative function $d\colon L_l \times L_l \to \mathbb{R}_0^+$. For any*

```
1  Function merge-nodes(L_l, l)
2      T ← pairs of nodes (u, v), u, v ∈ L_l for which (z^lp(u), z^lp(v)) is minimal;
3      T' ← pairs of nodes (u, v) ∈ L_l for which d(u, v) is minimal;
4      select (u, v) ∈ T' randomly;
5      L_l = L_l \ {u, v};
6      w ← u ⊕ v;
7      if ∃w' ∈ L_l | s(w) = s(w') then
8          w' ← w' ⊕ w;
9      else
10         L_l = L_l ∪ {w};
11     end
```

Algorithm 3. Iterative minLP merging function with similarity-based tie breaking.

triple of nodes $u_1, u_2, v \in L_l$*, we demand that if* $F_{s(u_1 \oplus v)} \supset F_{s(u_2 \oplus v)}$*,* $d(u_1, v) \geq d(u_2, v)$ *should hold.*

The goal is to find a distance function *for a specific problem* such that greater distance means a higher probability of introducing new paths and thus new represented solutions in the decision diagram, even if the states of the nodes are uncomparable. To consider a merging distance in the current state-of-the-art merging heuristics, we first look at an iterative minLP variant, where we find the use of the state similarity as a straightforward extension in form of a tie breaking mechanism. This becomes relevant when there are two pairs of nodes $(u_1, v), (u_2, v), s(u_1) \neq s(u_2)$ for which $(z^{\text{lp}}(u_1), z^{\text{lp}}(v)) = (z^{\text{lp}}(u_2), z^{\text{lp}}(v))$—then we simply take the pair with minimal distance according to d, see Algorithm 3. In the case of bulk minLP, a tie breaking is necessary when multiple nodes with the same rank go through the *merging boundary*, see Fig. 2, which separates the nodes to be merged from those to be kept as they are. Since the iterative minLP merging always takes pairs of nodes with currently smallest ranks with respect to z^{lp}, an alternative implementation is to first do a bulk merge of the nodes that have rank less than the one causing the need for tie breaking and then switch to merging nodes pairwise:

1. For a given layer l with nodes L_l, sort the nodes according to their current longest path length $z^{\text{lp}}(u)$ in decreasing order.
2. If the rank r of the $\beta - 1$-th node equals the rank of the β-th node, then we select all nodes with that rank r into a tie breaking set $T \subset L_l$; otherwise we do a simple minLP bulk merging.
3. Let B be the set of nodes that have a rank $<r$. We merge them yielding a node w with state $s(w)$ that is either still at the end of the ordered list or is absorbed by another node, if there already exists a node w' with $s(w) = s(w')$.
4. Finally, we iteratively merge pairs of nodes out of $T \cup \{w\}$ (or T if w has been absorbed) until the desired width β is reached. In each iteration, we choose the pair u, v that currently has minimal distance $d(u, v)$.

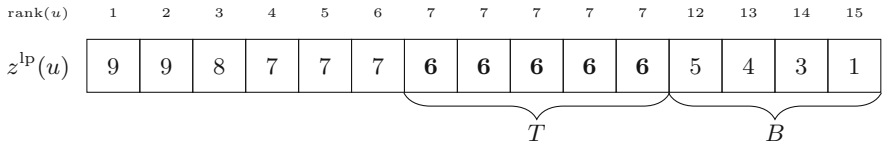

Fig. 2. Example layer with $|L_l| = 15$ nodes sorted by longest path length $z^{\text{lp}}(u)$, which is shown in the nodes. Let the maximum width be $\beta = 10$. All nodes with longest path value 6 (bold) are now subject to tie breaking.

When considering weighted problems, ties are in general substantially less likely to occur than in unweighted counterparts—still, we want to take the state similarity into account when differences in the longest path lengths are small. For that purpose, we introduce a parameterized hybrid merging algorithm, which is based on the minLP ordering but artificially introduces a region of nodes of similar longest path value around the merging boundary with which we deal as with the tie breaking region above. This region is determined by parameters δ_l, δ_r. To have meaningful parameters tunable between 0 and 1, regardless of the absolute values of the longest path lengths, we first normalize those according to the following transformation:

$$\tilde{z}^{\text{lp}}(u) = \frac{z^{\text{lp}}(u) - \min_{v \in L_l} z^{\text{lp}}(v)}{\max_{v \in L_l} z^{\text{lp}}(v) - \min_{v \in L_l} z^{\text{lp}}(v)} \tag{11}$$

The reference value is obtained by taking the normalized path value $\tilde{z}^{\text{lp}}_{\text{ref}}$ of the node immediately right to the merging boundary, i.e., the node with the largest value to be merged, if regular minLP would be applied. Now, two regions (contiguous sets of nodes in the ordered view of the layer) are defined:

1. bulk merging region $B := \{u \in L_l \mid \tilde{z}^{\text{lp}}(u) \in (\tilde{z}^{\text{lp}}_{\text{ref}} - \delta_r, 0.]\}$
2. pairwise merging region $T := \{u \in L_l \mid \tilde{z}^{\text{lp}}(u) \in [\tilde{z}^{\text{lp}}_{\text{ref}} + \delta_l, \tilde{z}^{\text{lp}}_{\text{ref}} - \delta_r]\}$

Let $w \leftarrow \oplus B$ be the node resulting from the bulk merging of B, and $L_l - T - B$ are the nodes that are kept as they are. The pairwise merging is now performed iteratively by always selecting a node pair with minimum distance d from $T \cup w$ and replacing the two nodes by the merged node until the desired layer width is reached. Setting $\delta_l = 0.0$, $\delta_r = 0.0$ yields the bulk-iterative hybrid as described before that only considers pairwise merging for real ties, whereas $\delta_l = 1.0$, $\delta_r = 1.0$ would completely ignore the longest path information and only focus on iteratively finding two minimum distance nodes to merge. For the choice of the pairwise merging region T in an example layer, see Fig. 3.

As mentioned before, it is crucial to conceive a meaningful distance function for a concrete problem. Notice that each node u in a layer has a maximum remaining path length $\max_{e \in F_{s_i = s(u)}} f(e)$, where $f(e = (d_{\pi_{i+1}}, \ldots, d_{\pi_n}))$ is the length of the feasible completion (see Definition 1), which is clearly not known to us during the construction of the DD at layer l. Still, a possible construction

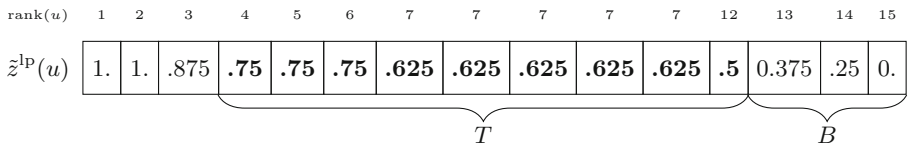

Fig. 3. Example layer with $|L_l| = 15$ nodes sorted by normalized longest path length $\tilde{z}^{\mathrm{lp}}(u)$, which is also shown in the nodes. Let the maximum width be $\beta = 10$, $\delta_l = \delta_r = .125$. All nodes with normalized longest path value $.625 \pm .125$ (bold) are now subject to pairwise merging.

Function *merge-nodes(L_l, l)*
  $\tilde{z}^{\mathrm{lp}}(u) \leftarrow \frac{z^{\mathrm{lp}}(u) - \min_{v \in L_l} z^{\mathrm{lp}}(v)}{\max_{v \in L_l} z^{\mathrm{lp}}(v) - \min_{v \in L_l} z^{\mathrm{lp}}(v)} \; \forall u \in L_l$;
  $B \leftarrow \{u \in L_l | \tilde{z}^{\mathrm{lp}}(u) \in (\tilde{z}^{\mathrm{lp}}_{\mathrm{ref}} - \delta_r, 0.]\}$;
  $T \leftarrow \{u \in L_l | \tilde{z}^{\mathrm{lp}}(u) \in [\tilde{z}^{\mathrm{lp}}_{\mathrm{ref}} + \delta_l, \tilde{z}^{\mathrm{lp}}_{\mathrm{ref}} - \delta_r]\}$;
  $L_l = L_l \backslash B$;
  $w \leftarrow \oplus B$;
  include-node-into-layer(w, L_l, l);
  while $|L_l| > \beta$ **do**
    $T' \leftarrow$ pairs of nodes $(u, v) \in T$ for which $d(u, v)$ is minimal;
    select $(u, v) \in T'$ for which $\max\{z^{\mathrm{lp}}(u), z^{\mathrm{lp}}(v)\}$ is minimal;
    $L_l = L_l \backslash \{u, v\}$;
    $T = T \backslash \{u, v\}$;
    $w \leftarrow u \oplus v$;
    include-node-into-layer(w, L_l, l);
  end

Function *include-node-into-layer(w, L_l, l)*
  if $\exists w' \in L_l | s(w) = s(w')$ **then**
    $w' \leftarrow w' \oplus w$;
  else
    $L_l = L_l \cup \{w\}$;
  end

Algorithm 4. Bulk-iterative minLP-state similarity-based hybrid merging algorithm with parameters δ_l, δ_r.

scheme to formulate a distance between u and v is to consider the maximum increase of the maximum remaining path lengths that u and v experience by being merged to $w = u \oplus v$:

$$d(u, v) = \max\{\max_{e \in F_{s(w)}} f(e) - \max_{e \in F_{s(u)}} f(e), \max_{e \in F_{s(w)}} f(e) - \max_{e \in F_{s(v)}} f(e)\} \tag{12}$$

This is can be made use of by approximating the maxima by an upper bound function $z^{\mathrm{ub}}(u)$:

$$d_{\mathrm{ub}}(u, v) = \max\{z^{\mathrm{ub}}(w) - z^{\mathrm{ub}}(u), z^{\mathrm{ub}}(w) - z^{\mathrm{ub}}(v)\} \tag{13}$$

For the MISP a first coarse upper bound to consider is given by the cardinality of the state $|s(u)|$, which is only reasonably tight for sparse graphs, but might still be meaningful since we are only interested in the maximum increase:

$$d^{\text{MISP}}_{\text{coarse}}(u,v) = \max\{|s(u) \cup s(v)| - |s(u)|, |s(u) \cup s(v)| - |s(v)|\} \tag{14}$$

In the weighted MISP (MWISP) case, we sum over the vertex weights of the remaining vertices defined by the state, $z^{\text{ub}}_{\text{MWISP}}(u) = \sum_{j \in s(u)} f_j(x_j = 1)$ to get a coarse upped bound. With SCP, we are facing a minimization problem; there the distance can be defined as maximum *lower bound* change:

$$d_{\text{lb}}(u,v) = \max\{z^{\text{lb}}(u) - z^{\text{lb}}(w), z^{\text{lb}}(v) - z^{\text{lb}}(w)\} \tag{15}$$

In this case, the calculation of a bound on the maximum remaining path length takes a little more work: We go over the remaining elements to be covered and if at the i-th, we increase a counter by one, if none of its covering sets was also a covering set for some $j < i$. The resulting counter value is a lower bound for the number of sets to cover the universe.

Another construction method is to take only the maximum remaining path length after merging $w = u \oplus v$:

$$\tilde{d}(u,v) = \max_{e \in F_{s(w)}} f(e) \tag{16}$$

The rationale is that it should be less likely to merge nodes that have high upper bounds even if they are similar with respect to $d(u,v)$, to balance the resulting upper bounds over the layer:

$$\tilde{d}_{\text{ub}}(u,v) = z^{\text{ub}}(w) \tag{17}$$

As a baseline, we suggest to also include the weighted Hamming distance d_H. It sums the weights of elements that are part of state $s(u)$ but not $s(v)$ or vice versa; for the unweighted case this amounts simply the cardinality of the symmetric set difference.

To summarize, the main idea of these construction methods was to greedily impede the estimated growth in bound by the remaining layers of the decision diagram induced by merging. One subtlety is that a problem specific upper bound does not take future merging operations into account but considers the case when we would continue constructing the BDD *without* merging.

5 Computational Study

We tested the relaxed DD construction applying the minLP merging heuristic with simple tie breaking based on the natural node order as done in [3] (i.e., the classic minLP) and minLP with our new similarity-based tie breaking using different distance functions for the MISP on random graphs from [5] with $n = 200$ and densities from $\{0.1, 0.2 \ldots, 0.9\}$ (20 instances per combination) and on the

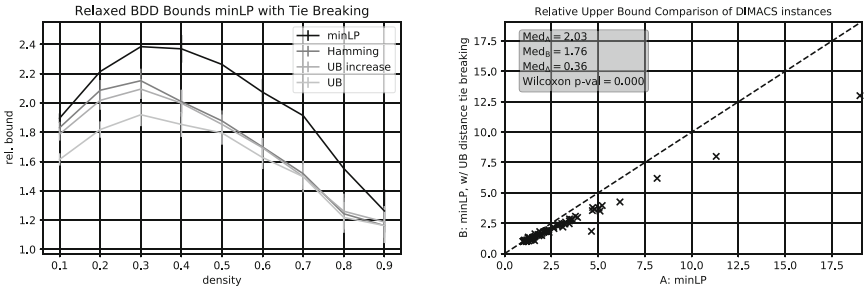

Fig. 4. Comparison of relative bounds of relaxed BDDs with $\beta = 10$ obtained with the classic minLP merging heuristic and with minLP with similarity-based tie breaking using different distance functions. Left: plotted over densities with means and error bars of 1σ; right: scatter plot for classic minLP vs. minLP with $\tilde{d}_{\text{ub}}$ based tie breaking.

DIMACS [8] max clique set instances[1]. For the SCP, we created random instances with $n = 500$ elements that are covered by exactly $k = 20$ sets each following the creation procedure for structured random instances from [6]. The constraint matrices describing which sets cover which elements follow a specific staircase like structure with limited bandwidths from $\{21, \ldots, 27\}$, and there are 20 instances per bandwidth. For the weighted MISP, we used 64 extended DIMACS graphs that we could solve to optimality in which vertex $i \in \{1, \ldots, n\}$ has weight $i \bmod 200 + 1$[2]. All tests were conducted on an Intel Xeon E5-2640 processor with 2.40 GHz in single-threaded mode and a memory limit of 8 GB.

On the left side of Fig. 4 we see the performance of the different tie breaking distance functions from Sect. 4 in comparison to the classic minLP approach in terms of the obtained relative bounds (i.e., obtained bounds divided by known optimal objective values) on the MISP random graph instances when compiling relaxed BDDs of maximum width $\beta = 10$. The tie breaking that seeks for pairs for which merging yields the smallest trivial upper bound (cardinality of state set), gives the strongest results. Differences among the approaches are generally larger for sparser graphs and start to vanish for denser graphs. This is plausible since the trivial upper bound is tighter for sparser graphs. The difference reaches a maximum for density 0.3 of about 40%. On its right side Fig. 4 shows a scatter plot with the relative bounds obtained for the DIMACS graph instances for classic minLP and our minLP with similarity-based tie breaking with the upper bound distance function. The median of the pairwise difference is 36% in favor of our tie breaking. A Wilcoxon signed rank sum test indicated that this difference is significant with an error probability of less than one percent.

Mean values of relative upper bounds and corresponding standard deviations for the different densities and algorithm variants are listed in Table 1 for $\beta = 10$

[1] http://www.andrew.cmu.edu/user/vanhoeve/mdd/code/opt_bounds_bdd-instances.tar.gz.

[2] https://github.com/jamestrimble/max-weight-clique-instances/tree/master/DIMACS.

Table 1. Mean relative upper bounds $\bar{u}_{\text{rel}}$ and standard deviations of relaxed BDDs over the 20 random graphs per density p obtained by the different merging heuristics for DD widths $\beta \in \{10, 100\}$.

p	$\beta = 10$								$\beta = 100$							
	minLP		d_H		d_{ub}		$\tilde{d}_{\text{ub}}$		minLP		d_H		d_{ub}		$\tilde{d}_{\text{ub}}$	
	$\bar{u}_{\text{rel}}$	σ_{ub}	$\bar{u}_{\text{rel}}$	σ_{ub}	$\bar{u}_{\text{rel}}$	σ_{ub}	$\bar{u}_{\text{rel}}$	σ_{ub}	$\bar{u}_{\text{rel}}$	σ_{ub}	$\bar{u}_{\text{rel}}$	σ_{ub}	$\bar{u}_{\text{rel}}$	σ_{ub}	$\bar{u}_{\text{rel}}$	σ_{ub}
0.10	1.90	0.03	1.83	0.04	1.79	0.04	**1.61**	0.04	1.63	0.03	1.57	0.04	1.56	0.04	**1.49**	0.03
0.20	2.21	0.07	2.09	0.07	2.02	0.06	**1.82**	0.06	1.80	0.06	1.70	0.04	1.67	0.05	**1.62**	0.06
0.30	2.38	0.10	2.15	0.08	2.09	0.07	**1.92**	0.08	1.80	0.08	**1.63**	0.04	**1.63**	0.06	**1.63**	0.07
0.40	2.37	0.09	2.01	0.08	2.00	0.09	**1.85**	0.09	1.69	0.07	1.51	0.07	1.54	0.06	**1.50**	0.07
0.50	2.26	0.09	1.88	0.07	1.85	0.07	**1.80**	0.08	1.54	0.08	**1.38**	0.06	1.39	0.05	1.40	0.06
0.60	2.07	0.09	1.70	0.06	1.69	0.07	**1.63**	0.08	1.35	0.05	**1.21**	0.05	1.24	0.04	1.22	0.04
0.70	1.91	0.11	1.52	0.08	**1.50**	0.11	**1.50**	0.08	1.22	0.07	1.12	0.05	**1.11**	0.06	**1.11**	0.07
0.80	1.55	0.16	1.24	0.11	1.26	0.12	**1.22**	0.11	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00
0.90	1.26	0.15	**1.16**	0.12	1.19	0.11	**1.16**	0.12	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00

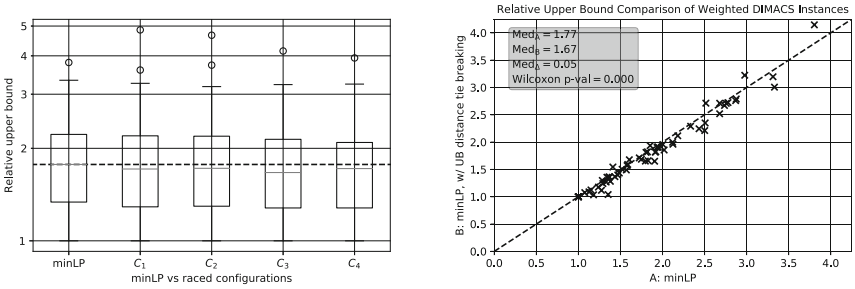

Fig. 5. Comparison of relative bounds of relaxed BDDs with $\beta = 10$ for weighted DIMACS instances for classical minLP vs. minLP with similarity-based merging in configuration C_3 with distance function $\tilde{d}_{\text{ub}}$.

and $\beta = 100$. For selected DIMACS instances relative upper bound values are shown likewise in Table 2. For the weighted DIMACS graph instances, where real ties are virtually non-existent, we tuned the left and right threshold parameters δ_l and δ_r for the region to which similarity-based merging is applied with irace [11] and test on a different set of weighted DIMACS graphs where the vertices have been randomly permuted. Here, we always used the superior upper bound distance function. On the left side of Fig. 5 we see boxplots comparing the raced parameter configurations with the classical minLP approach. The right side of Fig. 5 shows the comparison of the DDs' relative bounds when using the most promising configuration $C_3 = (0.185, 0.043)$. We observe that occasionally worse bounds are obtained but still in the clear majority of the cases the state similarity-based merging yields tighter bounds, which is also confirmed by a Wilcoxon signed rank sum test with an error probability of less than one percent. The median of the pairwise differences is 0.05.

Table 2. Relative upper bounds of relaxed BDDs obtained with different merging heuristics and widths $\beta \in \{10, 100\}$ for selected DIMACS instances.

inst	$\beta = 10$				$\beta = 100$			
	minLP	d_H	d_{ub}	$\tilde{d}_{\text{ub}}$	minLP	d_H	d_{ub}	$\tilde{d}_{\text{ub}}$
brock200_1	2.29	2.14	2.14	**1.90**	1.81	**1.62**	1.67	1.67
C500.9	3.05	3.00	2.81	**2.47**	2.61	2.46	2.40	**2.28**
gen400_p0.9_55	2.25	2.13	2.04	**1.82**	1.91	1.82	1.80	**1.73**
keller4	1.91	**1.55**	1.64	**1.55**	1.45	**1.18**	**1.18**	**1.18**
MANN_a45	1.34	1.34	**1.21**	1.30	**1.08**	1.32	1.27	1.19
p_hat300-3	2.19	2.11	2.08	**1.86**	1.86	1.75	1.81	**1.69**
p_hat700-2	2.59	2.45	2.32	**2.18**	2.14	1.98	1.95	**1.93**

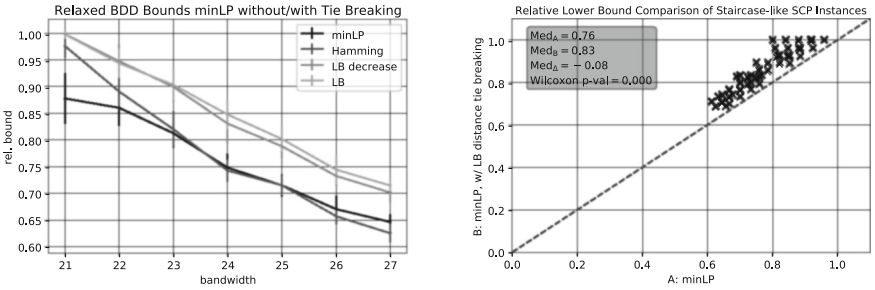

Fig. 6. Comparison of relative bounds of relaxed BDDs with $\beta = 10$ for staircase-like set cover problem instances with $n = 500$ elements to cover with varying bandwidths $b_w \in \{21, \ldots, 27\}$ obtained with the classic minLP merging heuristic and with minLP with similarity-based tie breaking using different distance functions. Left: plotted over different bandwidths with error bars for 1σ; right: scatter plot for classic minLP vs. minLP with tie-breaking based on $\tilde{d}_{\text{ub}}$.

In Fig. 6, we see the results for analogous comparisons for the set cover problem. As this is a minimization problem, we seek high lower bound values. Again, the lower bound distance turns out to be the most promising and gives statistically significant improvements with a median increase in the lower bound value of 0.08.

6 Conclusion and Future Work

We presented a possibility to improve the minLP merging heuristic in the layer-wise construction of a relaxed BDD. This extension turns in case of ties to a pairwise merging strategy that considers the state similarities for deciding which nodes to merge next. For unweighted problems, ties occur naturally and we obtain significant improvements for MISP random graphs, DIMACS instances, and for the set cover problem with random staircase-like instances.

In the weighted case, due to too few real ties, we generalized the method by considering a range of nodes with close longest path lengths for our similarity-based merging. We see a small but significant improvement for weighted DIMACS instances, after having tuned the corresponding parameters. The computational overhead introduced by our approach depends on the number of ties or the parameters δ_l and δ_r in the generalized variant as well as the applied distance function. However, since minLP still is the dominant criterion for deciding which nodes to merge, the set of nodes to be processed by the pairwise similarity-based merging is typically quite restricted. Our focus was on obtaining relaxed DDs of small width that provide stronger bounds. Such DDs are particularly important when they are used in some further algorithm many times, as frequently is the case in practical applications. Then, an overhead in the DD's construction will quickly pay off. Our ongoing research is concerned with achieving more effect on weighted instances, testing on further problem classes and reducing the time complexity so that state similarity-based approaches become also more effective for larger decision diagram width.

References

1. Bergman, D., Cire, A.A., van Hoeve, W.-J., Hooker, J.N.: Variable ordering for the application of BDDs to the maximum independent set problem. In: Beldiceanu, N., Jussien, N., Pinson, É. (eds.) CPAIOR 2012. LNCS, vol. 7298, pp. 34–49. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29828-8_3
2. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.N.: Decision Diagrams for Optimization. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42849-9
3. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.N.: Discrete optimization with decision diagrams. INFORMS J. Comput. **28**(1), 47–66 (2016)
4. Bergman, D., Cire, A.A., van Hoeve, W.J., Yunes, T.: BDD-based heuristics for binary optimization. J. Heuristics **20**(2), 211–234 (2014)
5. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.N.: Optimization bounds from binary decision diagrams. INFORMS J. Comput. **26**(2), 253–268 (2013)
6. 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
7. Hadzic, T., Hooker, J.: Postoptimality analysis for integer programming using binary decision diagrams. In: GICOLAG Workshop (Global Optimization: Integrating Convexity, Optimization, Logic Programming, and Computational Algebraic Geometry), Vienna. Technical report, Carnegie Mellon University (2006)
8. Johnson, D.S., Trick, M.A.: Cliques, coloring, and satisfiability: second DIMACS implementation challenge, 11–13 October 1993, vol. 26. American Mathematical Soc. (1996)
9. Kell, B., van Hoeve, W.-J.: An MDD approach to multidimensional bin packing. In: Gomes, C., Sellmann, M. (eds.) CPAIOR 2013. LNCS, vol. 7874, pp. 128–143. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38171-3_9

10. Lee, C.Y.: Representation of switching circuits by binary-decision programs. Bell Syst. Tech. J. **38**(4), 985–999 (1959)
11. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016)