Merging Quality Estimation for Binary Decision Diagrams with Binary Classifiers

Nikolaus Frohner(✉) and Günther R. Raidl

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

Abstract. Relaxed binary decision diagrams (BDDs) are used in combinatorial optimization as a compact representation of a relaxed solution space. They are directed acyclic multigraphs which are derived from the state space of a recursive dynamic programming formulation of the considered optimization problem. The compactness of a relaxed BDD is achieved by superimposing states, which corresponds to merging BDD nodes in the classical layer-wise top-down BDD construction. Selecting which nodes to merge crucially determines the quality of the resulting BDD and is the task of a merging heuristic, for which the *minimum longest path value* (minLP) heuristic has turned out to be highly effective for a number of problems. This heuristic sorts the nodes in a layer by decreasing current longest path value and merges the necessary number of worst ranked nodes into one. There are, however, also other merging heuristics available and usually it is not easy to decide which one is more promising to use in which situation. In this work we propose a prediction mechanism to evaluate a set of different merging mechanisms at each layer during the construction of a relaxed BDD, in order to always select and apply the most promising heuristic. This prediction is implemented by either a perfect or by a k-layers lookahead construction of the BDD, gathering feature vectors for two competing merging heuristics which are then fed into a binary classifier. Models based on statistical tests and a feed-forward neural network are considered for the classifier. We study this approach for the maximum weighted independent set problem and in conjunction with a parameterized merging heuristic that takes also the similarity between states into account. We train and validate the binary classifiers on random graphs and finally test on weighted DIMACS instances. Results indicate that relaxed BDDs can be obtained whose upper bounds are on average up to ≈16% better than those of BDDs constructed with the sole use of minLP.

Keywords: Binary decision diagrams · Merging heuristics · Lookahead construction · Binary classifiers

1 Introduction

Binary decision diagrams (BDDs) were introduced in the 1950s by Lee [7] as a compact representation for boolean functions. In the last decade, they have

© Springer Nature Switzerland AG 2019
G. Nicosia et al. (Eds.): LOD 2019, LNCS 11943, pp. 445–457, 2019.
https://doi.org/10.1007/978-3-030-37599-7_37

gained increasing popularity in the field of combinatorial optimization, where BDDs are used as a graphical representation of the solution space of a given optimization problem, constructed from a dynamic-programming-like recursive formulation of the solution space. Formally, a BDD is a directed acyclic multi-graph $B = (U, A)$ with node set U and arc set A. Each $u \in U$ is associated with a state $s(u)$ of the respective recursive formulation. Paths from a root node $\mathbf{r}$ through the BDD correspond to (partial) solutions and carry a length, corresponding to the solution costs; a longest path to a designated target node $\mathbf{t}$ then corresponds to an optimal solution for a maximization problem[1]. For a thorough introduction, we recommend the book by Bergman, Cire, van Hoeve, and Hooker [3].

Throughout this paper, we focus specifically on *relaxed limited-width* BDDs, which are constructed layer-by-layer in a breadth-first-search fashion. While exact BDDs model the solution space exactly but typically have exponential size for hard combinatorial optimization problems, relaxed BDDs represent a discrete relaxation and are kept compact by limiting the width, i.e., the number of nodes, at each layer. This width limitation is achieved by layer-wise *merging* of nodes: Whenever a layer is about to become too large, nodes are selected and merged, which means that their states are superimposed in a way that guarantees not to lose any feasible solutions. These merging operations, however, in general introduce new paths that do not represent feasible solutions. Therefore, the relaxed BDD represents a discrete relaxation of the original problem and the length of the longest path corresponds to an upper bound of the optimal solution value.

The method to select the nodes to be merged, called *merging heuristic*, is crucial for the quality of the resulting bound. We propose a method to evaluate multiple available merging heuristics at a given layer and to choose the believed-to-be locally best one.

The next section recaps the well-known *minimum longest path value* (minLP) merging heuristic and the parameterized state-similarity based merging heuristic we introduced in [5]. In Sect. 3, we introduce a method to estimate the quality relative to minLP of any merging heuristic applied at a given layer by conducting either a perfect or a k-layers lookahead and predicting the resulting bound at the final layer. This allows us to select different merging heuristics at each layer and gracefully deviate from the minLP merging heuristic. We always compare merging heuristics pairwise using a binary classifier, either based on a simple feature comparison test, linear regression, a Wilcoxon signed rank sum test or an artificial neural network, see Sect. 4, which gives us a probabilistic estimate whether one is better than the other. We present the data preparation for our binary classifier training and validation together with our computational study in Sect. 5, where we consider the maximum weighted independent set problem

[1] We consider only maximization throughout this paper. The methods are, however, equally applicable to minimization by changing the sign of the objective function.

(MWISP) with training on random weighted graphs and final tests on weighted DIMACS[2] instances. We conclude and give indications for future work in Sect. 6.

In the MWISP, we are given a graph $G = (V, E)$ and costs $c_j \in \mathbb{R}$ for each node $j \in V$. We seek to find a set of nodes $S \subset V$ with maximum costs $\sum_{j \in S} c_j$ for which no two nodes are adjacent in G. In a recursive formulation, we assign to each node of the graph a binary decision variable x_i and impose an ordering π_i on these. A state is the set of nodes that can still be added to the current independent set. When at step i of the recursion, we decide either to add the node $\pi_i = j$ to the independent set, setting $x_i = 1$, which removes the node and its neighborhood $N(j)$ from the set, or to leave it, setting $x_i = 0$, which only removes the node itself; this is encoded by a corresponding state transition function τ:

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

$$(0, s_i) \mapsto s_{i+1} = \tau(0, s_i) = s_i - \{j\} \tag{2}$$

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

We denote with D_{s_i} the admissible values for x_i in state s_i, which is $\{0, 1\}$ when $j \in s_i$, otherwise $\{0\}$. The maximization problem is then given by the following Bellman equations:

$$z^*(s_i) = \max_{d \in D_{s_i}} \{c_j d + z^*(\tau(d, s_i))\} \tag{4}$$

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

$z^*(s_0)$, where the initial state is given by $s_0 = V$, yields the cost of a maximum weighted independent set.

2 Related Work

For the layer-wise construction of limited-width BDDs for MWISP instances, we follow the algorithm described by Bergman et al. [2], which employs zero-suppressing long-arcs, the minimal state (minState) variable ordering heuristic, and the minimum longest path length (minLP) merging heuristic. For the MWISP, the state $s(u)$ associated with a node u of the BDD is the set of nodes that can still be included in the independent set. Successors of a state are obtained by setting a still open variable to either one or zero, representing the decision that the corresponding node is either included or excluded, respectively. Arc lengths correspond to the gains in the objective function, i.e., the weight of the respective node if included or zero otherwise. The used minState variable ordering heuristic selects at each layer always a variable that appears in the fewest number of states associated with the BDD nodes that were generated by previous layers and still need to be placed on a layer. The minLP merging heuristic sorts the nodes u at a current layer in decreasing order of their currently longest path length $z^{\text{lp}}(u)$ from the root to them and merges the necessary

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

number of nodes from the back into one node, so that the maximum width is kept. Merging is done by applying the set union over all affected states.

In [5], we identified the similarity between states as worth considering for the merging decision and introduced a parameterized merging algorithm that also begins by bulk merging nodes with smallest longest path value into one node but not enough to reach the maximum width. Instead, the method then applies pairwise merging of the remaining nodes that have longest path values below some threshold by iteratively selecting two nodes with minimal dissimilarity. This dissimilarity between two nodes $u, v \in U$ can be defined in different ways but in [5] we found that considering an upper bound on the costs-to-go from the state $s(w)$ we would obtain when merging the two nodes into one new node w is particularly useful. In case of the MWISP, we choose the weighted sum over the remaining graph vertices that can still be selected after merging u and v:

$$d_{\mathrm{ub}}(u, v) = z^{\mathrm{ub}}_{\mathrm{MWISP}}(w) = \sum_{j \in s(w)} c_j \tag{6}$$

The merging boundary is a contiguous set of nodes in a layer that have the same rank when sorted by the longest path length to them and that could participate in the classical minLP merging. The parameters that control which nodes take part in the iterative pairwise similarity based merging are $(\delta_{\mathrm{l}}, \delta_{\mathrm{r}}) \in [0, 1]^2$. They define as a relative measure depending on the longest path values of the nodes how many nodes left (determined by δ_{l}) and how many nodes right (determined by δ_{r}) of the merging boundary should be taken, see Fig. 1 for the conceptual differences between the merging heuristics. The extreme cases are $(1.0, 1.0)$, where all nodes are potential merging candidates, and $(0.0, 0.0)$, which corresponds to minLP with an additional tie breaking when nodes at the merging boundary have the same longest path value and the pure rank-based merging would not be unique—for them the similarity based merging is applied.

Lookahead approaches [9] are very common in deterministic games, where different possible moves are compared by conducting a limited playout and evaluating the resulting configurations by an approximating evaluation function. The idea of a k-layers lookahead approach for BDDs has been presented by Bergman et al. [1] in the context of a dynamic variable ordering for the maximum independent set problem.

3 Merging Quality Estimation

So far, when constructing a limited-width BDD layer-wise, a predefined merging heuristic is repeatedly applied at every layer that would exceed the maximum width. We aim at higher flexibility by having a set of merging heuristics $\mathcal{H}$ at our disposal, and hope that a careful application of different heuristics improves the resulting quality of the BDD. At a current layer l of the BDD construction, the central question then is which heuristic to select to conduct the actual merging. Unfortunately, a reliable free-standing way of judging the potential of each of the heuristics in dependence of the current situation is not obvious. We therefore

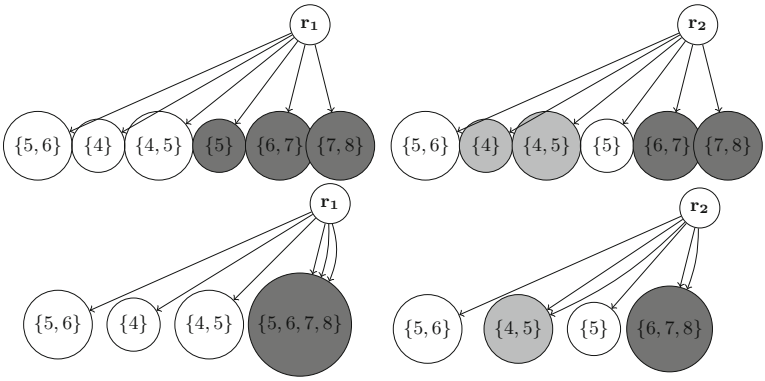

Fig. 1. An exemplary layer with set-based states where merging has to take place to reduce the width to 4. Left: the classical minLP merging heuristic which bulk-merges the necessary number of nodes with currently shortest path length (dark), potentially leading to large states; right: the merging heuristic from [5] which combines bulk-merging nodes with currently shortest path length (dark) with pairwise merging of nodes under consideration of their similarity (bright), distributing the state sizes more evenly.

suggest a different, amenable approach that always keeps the established minLP heuristic in $\mathcal{H}$ and uses it as a baseline to define a relative quality measure for each of the considered merging heuristics.

Definition 1 (Merging Quality). *Given are a set of merging heuristics $\mathcal{H}$, where minLP $\in \mathcal{H}$ and a BDD in construction facing a layer l to be merged. For each $H \in \mathcal{H}$ we create a shallow copy of the BDD, conduct the merging determined by H and finish the construction of the BDD afterwards by only applying minLP. The resulting upper bounds on the objective value act as measure of quality for H at layer l: We write $H \prec H'$ when H yields a strictly tighter bound than H', and H is then considered the locally better choice than H'.*

Overall, after evaluating each merging heuristic in the described way, we actually select and apply a dominating one and continue at the next layer in the same way; ties are broken randomly. Clearly, this complete lookahead for each considered merging heuristic at each level is computationally expensive, but what we obtain is the possibility to measure the impact of the different heuristics and finally we can also study how often each method has been applied in the construction of BDDs. Thus, we can see whether this combination of multiple merging heuristics may in principle improve the bounds of the resulting BDDs for our given problem instances. A pseudo-code for this lookahead algorithm is shown in Algorithm 1.

The approach also allows to generate the ground truth for a corresponding classification problem: given heuristics H and H', let f be a binary function returning 1, iff $H \prec H'$, i.e., H provides a tighter bound than H', and 0 otherwise. To make the approach viable in practice, we move from the

Input: BDD B under construction, current layer l, maximum layer $l_{\max}$, set of competing merging heuristics $\mathcal{H}$ including minLP
Output: Winning merging heuristic $H \in \mathcal{H}$

Function *perfect-lookahead(B, l, $l_{\max}$, $\mathcal{H}$)*
  for $H \in \mathcal{H}$ **do**
    $B' \leftarrow$ shallow copy of B;
    apply H to B' at layer l;
    continue B' with minLP until reaching final layer $l_{\max}$;
    $z_H^{\text{lp}} \leftarrow z^{\text{lp}}(\mathbf{t})$ in B';
  end
  $H^* \leftarrow \operatorname{argmin}_{H \in \mathcal{H}} z_H^{\text{lp}}$;
  return H^*;

Algorithm 1: Perfect lookahead algorithm for deciding which merging heuristic $H \in \mathcal{H}$ to use at layer l.

evaluation by *perfect lookahead* to a statistically estimated variant that only considers the next k layers, i.e., a *k-layers lookahead*. During construction with alternative heuristics, when at layer l, we apply H and continue the construction of a shallow copy of the BDD for $k-1$ more layers using minLP, yielding a feature matrix $\mathbf{Y_H} \in \mathbb{R}^{p \times k}$ for the looked-ahead layers $\{l, \ldots, l+k-1\}$, where p is the number of features. The distinguished baseline feature matrix is $\mathbf{Y_{minLP}}$, when only minLP was applied for the layers $\{l, \ldots, l+k-1\}$ in a shallow copy of the BDD. The learning goal now is to find a classifier function $h\colon \mathbb{R}^{p \times k} \times \mathbb{R}^{p \times k} \to \{0, 1\}$ for which wrong classifications in the sense of $h(\mathbf{Y_H}, \mathbf{Y_{H'}}) \neq f(\mathbf{Y_H}, \mathbf{Y_{H'}})$ are unlikely. Every layer provides a fixed number of features, for which we have to aggregate information from the variable number of nodes per layer. By taking the maximum, mean value, and minimum of the longest path values $z^{\text{lp}}(u)$ over all nodes u at the layer and likewise of the upper bound values $z^{\text{ub}}_{\text{MWISP}}(u)$, we identified six natural options to be used as features per layer. In the following sections, we consider different types of binary classifiers for the statistical lookahead problem based on a linear regression model, on the Wilcoxon signed rank sum test, and on training an artificial neural network on random weighted graphs.

4 Binary Classifiers

We consider parameterized binary classifiers $h_\alpha \in \mathcal{B}$ that are constructed by taking a function $\tilde{h}$ that provides an estimation of the probability that $H \prec H'$ and apply a threshold $\alpha \in (0, 1)$:

$$\tilde{h}\colon \mathbb{R}^{p \times k} \times \mathbb{R}^{p \times k} \to [0, 1]. \tag{7}$$

$$h(\mathbf{Y}, \mathbf{Y'}) = \begin{cases} 0, & \tilde{h}(\mathbf{Y}, \mathbf{Y'}) < \alpha \\ 1, & \tilde{h}(\mathbf{Y}, \mathbf{Y'}) \geq \alpha \end{cases} \tag{8}$$

Input: BDD under construction B, current layer l, number of layers to look ahead k, maximum layer $l_{\max}$, acceptance threshold α, set of merging heuristics to test $\mathcal{H}$, probabilistic binary classifier $\tilde{h}$
Output: Winning merging heuristic $H \in \mathcal{H}$

Function *k-layers-lookahead*(B, l, $l_{\max}$, $\mathcal{H}$, $\tilde{h}$, α)
  if $l + k \geq l_{\max}$ **then**
    return *perfect-lookahead*$(B, l, l_{\max}, \mathcal{H})$;
  end
  apply minLP to shallow copy B' at layer l, continue for $k-1$ layers with minLP;
  $\mathbf{Y}_{\text{minLP}} \leftarrow$ feature vectors for layers $\{l, \ldots, l+k-1\}$;
  for $H \in \mathcal{H} \setminus \{\text{minLP}\}$ **do**
    apply H to shallow copy B' at layer l, continue for $k-1$ layers with minLP;
    $\mathbf{Y}_H \leftarrow$ feature vectors for layers $\{l, \ldots, l+k-1\}$;
    $p_H \leftarrow \tilde{h}(\mathbf{Y}_H, \mathbf{Y}_{\text{minLP}})$;
  end
  $H^* \leftarrow \text{argmax}_{H \in \mathcal{H} \setminus \{\text{minLP}\}}\ p_H$;
  if $p_{H^*} \geq \alpha$ **then**
    return H^*;
  return minLP;

Algorithm 2: k-layers lookahead algorithm for deciding which merging heuristic $H \in \mathcal{H}$ to use at layer l by means of a probabilistic binary classifier $\tilde{h}$ parameterized by threshold α.

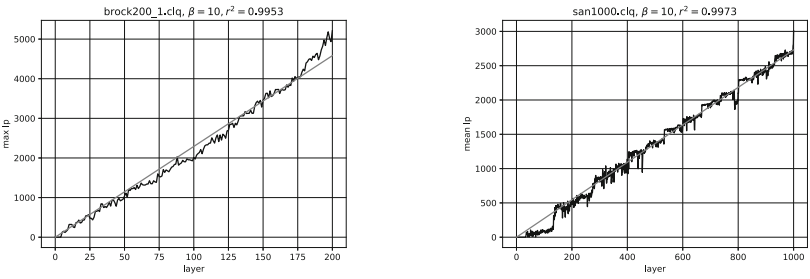

Fig. 2. Evolution of the maximum and mean longest path lengths over the layers with a linear regression line for different weighted DIMACS instances with maximum layer width $\beta = 10$.

Equipped with an $\tilde{h}$, we can formulate the k-layers-lookahead merging heuristic selection algorithm as listed in Algorithm 2. It compares every merging heuristic with minLP by feeding their feature matrices to the probabilistic binary classifier and saves for each H the resulting probability estimate p_H. If the largest p_H is greater than or equal to the acceptance threshold α, the corresponding winning heuristic H^* is returned, otherwise minLP.

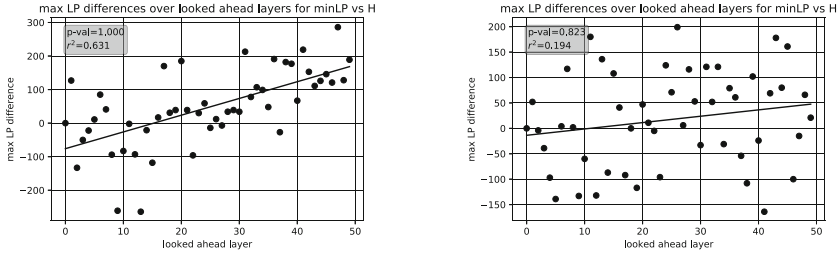

Fig. 3. Examples for a $k = 50$ layers lookahead with minLP versus a competing heuristic H, regressing the differences of the maximum longest path length (maxLP) values on the layers. Left: a true positive case (minLP worse than H), right: a true negative case (minLP not worse than H).

We now consider different possibilities for the probabilistic binary classifier $\tilde{h}$. In some preliminary experiments we observed that as a first approximation a linear dependence between layers and the maximum and mean longest path values is reasonable to assume, see Fig. 2. Considering minLP as default merging strategy, we want to test whether the growth trend for another merging heuristic H is significantly smaller, i.e., $H \prec \text{minLP}$. To do so, we restrict ourselves to one feature, for example the maximum longest path length for a layer (maxLP), calculate $\Delta\mathbf{Y}_H = \mathbf{Y}_{\text{minLP}} - \mathbf{Y}_H$, where $\Delta\mathbf{Y}_H \in \mathbb{R}^{1\times k}$, and solve the linear regression (LR) model

$$\Delta\hat{Y}_H(l;\theta,d) = \theta\, l + d. \tag{9}$$

A subsequent student t-test for the significance of $\hat{\theta} \neq 0$ yields a corresponding p-value, which can be transformed to a belief in $[0,1]$ that minLP grows towards a worse upper bound than H. We denote this linear regression based classifier as $\tilde{h}^{\text{LR}}$. For examples of a true positive and a true negative case for a lookahead length of $k = 50$, see Fig. 3. Since the linear regression based classifier considers only the slope, a non-parametric alternative is to use the p-value of a Wilcoxon signed rank sum test over the $\Delta\mathbf{Y}_H$, the layer-wise differences of the features.

As a more powerful alternative to the linear regression based classifier, we consider a feed-forward neural network (NN) $\tilde{h}^{\text{NN}}(\mathbf{Y}_H, \mathbf{Y}_{\text{minLP}})$ that yields a score in $[0,1]$ which we interpret as probability for $H \prec H'$. As features for the k layers created by the two different merging heuristics applied at layer l, we consider two from the possibilities namely the differences of the maximum longest path values and the difference of the maxima of the upper bounds over the given nodes by layer. Furthermore, we provide the graph density and the layer progress $l/l_{\max}$ as input, resulting in an input layer of the NN consisting of $2(k+1)$ neurons. These differences are normalized by dividing them by their maximum absolute value to feed values from $[-1,1]$ into the network in order to facilitate the training. Two hidden layers with twice the neurons of the input layer, i.e., $4(k+1)$, follow with a single neuron in the output layer. Each non-final neuron is configured with a reLU activation function, the final one with

a sigmoid activation function to obtain a value in $[0, 1]$. We use binary cross-entropy loss and train with minibatch gradient descent with a batch size of 64. We start with a learning rate of 10^{-3} and decrease by factors of 10 after reaching a plateau. To prevent overfitting, we use a weight decay factor of 10^{-3} and use early stopping by monitoring the accuracy on a validation set consisting of 20% of the original training samples. We trained the neural network using Keras[3] 2.0.8 with TensorFlow 1.2.1 backend.

To evaluate the performance of the binary classifiers, we calculate precision-recall curves by varying the acceptance threshold α of the binary classifier and gather the corresponding precision-recall data points. The precision is the number of true positives divided by the number of classified as positives and the recall the number of true positives divided by the overall positives. This allows to tune the classifier to the required behavior. Furthermore, a single area-under-curve (AUC) value can be calculated from this curve and compared for different classifiers. As a simple baseline classifier that is not based on a probability score, given one-dimensional feature vectors for both merging heuristics, we compare the maxima of the feature vectors and return 1 if $\max \mathbf{Y}_{\mathbf{H}} < \max \mathbf{Y}_{\mathbf{H}'}$ and otherwise zero. Its performance is then evaluated by a single precision-recall point. When we use the maxLP values as features, we call this max-maxLP, since we compare the maxima of the maxima.

5 Computational Study

For training and validation data, we created 1000 $G(n, p)$ random weighted graphs with parameters $n \in [100, 2000]$ and $p \in [0.05, 0.95]$ drawn uniformly at random. Weights are assigned from $\{1, \ldots, 200\}$ in dependence of the index of a vertex j via $j \bmod 200 + 1$. For each graph G_i, $i = 1, \ldots, 1000$, we construct a binary decision diagram with maximum width $\beta = 10$ and merging parameters $\delta_\mathrm{l} = \delta_\mathrm{r} = 0$ (i.e., minLP with tie breaking according to [5]) and save the resulting upper bound u_i and the features by layer $\mathbf{Y}^i_{\mathrm{minLP}}$. Furthermore, we sample a layer $l' \in \{1, \ldots, n_i\}$ uniformly at random, construct the corresponding binary decision diagram up to layer l' in the same fashion as before. If no merging is needed, we restart; otherwise we sample merging parameters $\delta_\mathrm{l}, \delta_\mathrm{r}$ uniformly at random and likely apply a different merging. After this, we continue the construction as before with $\delta_\mathrm{l} = \delta_\mathrm{r} = 0$ and save the resulting upper bounds $\tilde{u}^m_i$ and feature vectors by layer $\mathbf{Y}^i_m$ from l' to the last layer, where $m \in \{1, \ldots, 20\}$, resulting into 20 training samples per graph. This creates the ground truth for $f(\mathbf{Y}^i_m, \mathbf{Y}^i_{\mathrm{minLP}})$ which is one if the resulting upper bound $\tilde{u}^m_i$ of the alternative merging is strictly smaller than the pure minLP upper bound u_i, otherwise zero. When we train or test our binary classifiers, we set the fixed lookahead length k and extract only the corresponding subpart of the saved feature matrices in a preprocessing step.

[3] https://keras.io.

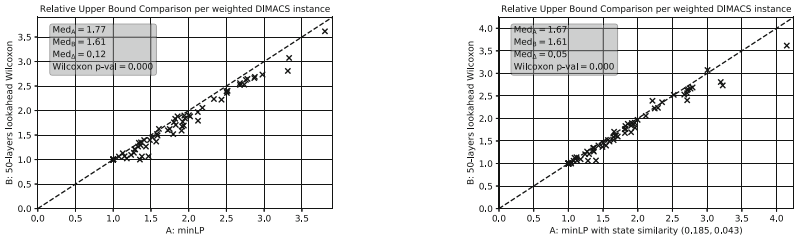

Fig. 4. Comparison of pure minLP (left) vs. minLP with state similarity and raced parameters (right) vs a combination of both on weighted DIMACS instances with statistical lookahead of 50 layers, using the Wilcoxon test classifier and an acceptance threshold of $\alpha = 0.95$.

We conducted the final tests on a weighted DIMACS graph set[4], from which we consider $N = 64$ instances $\mathcal{I}_{\text{WDIMACS}}$ that we could solve to optimality. This allows us to calculate for each graph instance $I \in \mathcal{I}_{\text{WDIMACS}}$ a relative bound u_I^{rel} from the absolute bound $u_I^{\text{rel}} = u_I / z_I^*$ derived from the construction of a relaxed BDD with one of the described approaches. As figures of merit when comparing two approaches, we consider the median and the mean of the pairwise differences $\widetilde{\Delta} := \text{Med}_{I \in \mathcal{I}_{\text{WDIMACS}}}[\Delta u_I^{\text{rel}}]$, $\overline{\Delta}$ of the relative upper bounds. First, we applied the perfect lookahead approach with the parameterized similarity based merging heuristic as introduced in [5]. The baseline merging heuristic is minLP with tie breaking, corresponding to parameters $\delta_\text{l} = \delta_\text{r} = 0$ and four further competing parameter sets $(\delta_\text{l}, \delta_\text{r}) \in \{(0.185, 0.043), (0.2, 0.2), (0.3, 0.3), (0.4, 0.4)\}$. The competing merging heuristics are used in only 4% of the layers with merging, still resulting in a median relative bound improvement of 0.16 over using only minLP alone. In [5], we tuned the parameter set $(0.185, 0.043)$ using irace [8], which gave a median improvement of 0.05, when applied alone. For the k-layers lookahead, we evaluate the linear regression, the Wilcoxon signed rank sum based variant, the NN, and the simple max-maxLP binary classifiers on the validation set of the random weighted graph instances described in Sect. 5. The NN classifier outperforms the other classifiers beginning from a lookahead length of $k = 30$ as can be seen in the precision-recall plot in Fig. 5 for $k = 50$ and for the precision-recall area-under-curve values calculated for $k \in \{10, 20, \ldots, 90\}$. For example, when we tune to a modest recall of 0.1, we get only 0.58 precision for the LR and Wilcoxon classifier but approximately 0.68 for the NN, which is not surprising since the latter considers more features, including upper bound information and graph density, whereas the former only relies on the maximum longest path values as feature. In general, the precision of the classifiers is relatively weak indicating a difficult classification problem.

The final test is conducted on the weighted DIMACS instances, and the median and mean of differences of relative upper bounds are used again as figures of merit and summarized in Table 1 for three different lookahead-lengths

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

$\{30, 50, 70\}$. We compare with the pure application of minLP and pure application of minLP with state similarity merging parameterized by $(0.185, 0.043)$. We see that the naive, completely parameter-less max-maxLP classifier that considers one heuristic to be worse than the other when the maximum of the maximum longest path values over the looked ahead layers is strictly greater yields results comparable to using the linear regression with a median relative bound improvement between 0.09 and 0.10. The also rather simple Wilcoxon test performed second best in our final test, yielding figures between 0.09 and 0.12 (Fig. 4), slightly worse than the NN classifier with figures between 0.11 and 0.16.

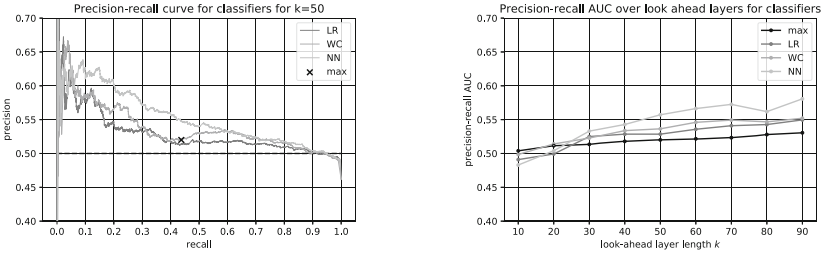

Fig. 5. Left: precision-recall curves for different classifiers with k-layers lookahead with $k = 50$ on random weighted graph instances; right: corresponding precision-recall AUC over lookahead length $k \in \{10, \ldots, 90\}$.

Table 1. Median $\widetilde{\Delta}$ and mean $\overline{\Delta}$ of pairwise differences of relative bounds for perfect lookahead and $\{30, 50, 70\}$-layers lookahead with different classifiers vs pure minLP and vs minLP with state similarity based merging with parameters $\delta_l = 0.185, \delta_r = 0.043$, on weighted DIMACS instances. (PLA = perfect lookahead, max = max-maxLP, LR = linear regression, WC = Wilcoxon, NN = neural network).

Comparing approach	PLA		k	max		LR		WC		NN	
	$\widetilde{\Delta}$	$\overline{\Delta}$		$\widetilde{\Delta}$	$\overline{\Delta}$	$\widetilde{\Delta}$	$\overline{\Delta}$	$\widetilde{\Delta}$	$\overline{\Delta}$	$\widetilde{\Delta}$	$\overline{\Delta}$
pure minLP	0.16	0.17	30	0.09	0.11	0.07	0.08	0.09	0.11	0.11	0.11
minLP with state similarity	0.09	0.11		0.04	0.06	0.02	0.03	0.04	0.06	0.04	0.06
pure minLP	0.16	0.17	50	0.09	0.11	0.09	0.11	0.12	0.13	0.12	0.13
minLP with state similarity	0.09	0.11		0.03	0.06	0.03	0.06	0.05	0.08	0.08	0.08
pure minLP	0.16	0.17	70	0.10	0.12	0.10	0.12	0.12	0.14	0.15	0.16
minLP with state similarity	0.09	0.11		0.04	0.06	0.03	0.07	0.05	0.09	0.08	0.11

To do a runtime comparison, we conducted all experiments on an Intel Xeon E5-2640 processor with 2.40 GHz in single-threaded mode and a memory limit of 8 GB using Python3.6. We measured runtimes when constructing the BDDs with 30-, 50-, and 70-layers-lookahead for two competing merging heuristics and relative to plain minLP, which gave us a median factor of $\approx 2k\times$, for example for a 30-layers-lookahead with the NN classifier $\approx 70\times$. This is what we expected

given this computationally demanding approach. Still, if the resulting decision diagram is more compact than another one with the same bound and is traversed many times afterwards, then this initial construction overhead may pay off. See for instance [6], where arcs in an already constructed relaxed decision diagram are repeatedly filtered to remove infeasible paths. An idea under investigation for further runtime reduction is to conduct the lookahead with a smaller width.

6 Conclusion and Future Work

In this paper, we have shown a method to locally evaluate the quality of merging heuristics in small-width binary decision diagrams by conducting a lookahead using the simple yet strong minimum longest path merging heuristic. We used this method to devise algorithms that allow different merging heuristics to compete against each other on a layer where merging is needed and subsequently apply the winning heuristic. The evaluation is either done by a computationally intensive perfect lookahead or by a k-layers lookahead where we try to approximate the perfect lookahead by means of binary classifiers based on either statistical tests or a neural network classifier. We trained, validated, and tuned the classifiers on random weighted graph instances and finally tested on a weighted DIMACS graph set where we could show significant bound improvements by combining minLP and our parameterized state-similarity based merging heuristic over only using either one alone.

Further research is needed to validate this lookahead approach on other problems, compare with other classification approaches such as logistic regression or support vector machines, and to possibly find computationally less demanding local quality estimation methods. In the context of branching heuristics of mixed-integer programming solvers, the "Dynamic approach for switching heuristics" [4] creates clusters of sub-problems in a feature space during an off-line training phase where a different heuristic works best for each cluster and then dynamically switches between these heuristics during traversal of the branch-and-bound tree. This could potentially also be an interesting approach for selecting different merging heuristics during the construction of a BDD.

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.: Optimization bounds from binary decision diagrams. INFORMS J. Comput. **26**(2), 253–268 (2013)
3. 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
4. Di Liberto, G., Kadioglu, S., Leo, K., Malitsky, Y.: DASH: dynamic approach for switching heuristics. Eur. J. Oper. Res. **248**(3), 943–953 (2016)

5. Frohner, N., Raidl, G.R.: Towards improving merging heuristics for binary decision diagrams. In: Proceedings of LION 13–13th International Conference on Learning and Intelligent Optimization. Lecture Notes in Computer Science, Springer (2019, to appear)
6. Horn, M., Raidl, G.R.: Decision diagram based limited discrepancy search for a job sequencing problem. In: Computer Aided Systems Theory - EUROCAST 2019. Lecture Notes in Computer Science, Springer (2019, to appear)
7. Lee, C.Y.: Representation of switching circuits by binary-decision programs. Bell Syst. Tech. J. **38**(4), 985–999 (1959)
8. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Persp. **3**, 43–58 (2016)
9. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley Publishing Co., Inc., Reading (1984)