# On the Use of Decision Diagrams for Finding Repetition-Free Longest Common Subsequences

Matthias Horn[1(✉)], Marko Djukanovic[1], Christian Blum[2], and Günther R. Raidl[1]

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria
{horn,djukanovic,raidl}@ac.tuwien.ac.at
[2] Artificial Intelligence Research Institute (IIIA-CSIC), Campus UAB, Bellaterra, Spain
christian.blum@iiia.csic.es

**Abstract.** We consider the repetition-free longest common subsequence (RFLCS) problem, where the goal is to find a longest sequence that appears as subsequence in two input strings and in which each character appears at most once. Our approach is to transform a RFLCS instance to an instance of the maximum independent set (MIS) problem which is subsequently solved by a mixed integer linear programming solver. To reduce the size of the underlying conflict graph of the MIS problem, a relaxed decision diagram is utilized. An experimental evaluation on two benchmark instance sets shows the advantages of the reduction of the conflict graphs in terms of shorter total computation times and the number of instances solved to proven optimality. A further advantage of the created relaxed decision diagrams is that heuristic solutions can be effectively derived. For some instances that could not be solved to proven optimality, new state-of-the-art results were obtained in this way.

**Keywords:** Repetition-free longest common subsequence · Decision diagram · Maximum independent set

## 1 Introduction

The *longest common subsequence* (LCS) problem asks for the longest string which is a subsequence of a set of input strings. A subsequence is a string that can be obtained from another string by possibly deleting characters. For instance a longest common subsequence of the two input strings `ABCDBA` and `ACBDBA` is `ABDBA`. The LCS problem has applications in bioinformatics, where strings often

This project is partially funded by the Doctoral Program "Vienna Graduate School on Computational Optimization", Austrian Science Foundation (FWF) Project No. W1260-N35. Furthermore, this work was also supported by project CI-SUSTAIN funded by the Spanish Ministry of Science and Innovation (PID2019-104156GB-I00).

© Springer Nature Switzerland AG 2020
N. Olenev et al. (Eds.): OPTIMA 2020, LNCS 12422, pp. 134–149, 2020.
https://doi.org/10.1007/978-3-030-62867-3_11

represent segments of RNA or DNA [13,16,18]. Other fields where the LCS problem appears are text editing [17], data compression, file comparison [2,19], and the production of circuits in field programmable gate arrays [8]. If the number of input strings $m$ is constant, the problem is solvable by *dynamic programming* (DP) in $O(n^m)$ time, where $n$ is the length of the longest input string [13]. Otherwise, if the number of input strings is arbitrary, the problem is $\mathcal{NP}$-hard. An additional constraint which arises in the context of gene duplication in the domain of genome rearrangement and which we consider in this work is that each character may appear in a *common subsequence* (CS) at most once. This problem, first introduced by Adi et al. [1] and denoted as the *repetition-free LCS* (RFLCS) problem, is usually considered for two input strings and is even then APX-hard [1].

This work builds upon the work of Blum et al. [6], where instances of the RFLCS problem are transformed to instances of the *maximum independent set* (MIS) problem. Hereby, an independent set of the underlying conflict graph of the MIS problem corresponds to a *repetition-free common subsequence* (RFCS) of the RFLCS instance. To solve the MIS problem the *integer linear programming* (ILP) solver CPLEX is applied. The performance of the ILP solver depends to a large extent on the size of the conflict graph. Therefore, in [6] the size of the conflict graph is reduced by filtering redundant nodes based on lower and upper bounds. This boosts the range of instances that can be solved to optimality as well as the quality of heuristic solutions obtained for larger instances. In this way, numerous new state-of-the-art results were obtained.

*Contributions.* To reduce the size of the conflict graph even further we compile a relaxed *multivalued decision diagram* (MDD) for the RFLCS problem, yielding a performance improvement of the subsequently applied ILP solver. In the last decade, *decision diagrams* (DDs) have been recognized as a powerful tool for combinatorial optimization problems; see [3] for a comprehensive survey. In particular, *relaxed DDs* may provide compact representations of discrete relaxations. Besides allowing for new interference techniques in constraint programming and novel branching schemes, they may also provide tight dual bounds. In case of the RFLCS problem it is further possible to effectively derive heuristic solutions directly from the relaxed MDD. This has the advantage that if the ILP solver is not able to solve an instance to proven optimality within a given time limit then the compiled relaxed MDD may be able to provide a tighter upper bound as the ILP solver does and/or may be able to deliver a better heuristic solution.

After an overview of related work in Sect. 2 we give a formal problem definition in Sect. 3. The MIS problem and a corresponding ILP model are described in Sect. 4. Decision diagrams for the RFLCS are introduced in Sect. 5, and Sect. 6 describes the incremental refinement of relaxed MDDs. Section 7 provides experimental results, showing that the suggested approach yields to a performance improvement in terms of average computation times, number of instances solved to optimality, and average solution quality.

## 2 Related Work

As already mentioned, the current work builds upon the approach of Blum et al. [6]. Besides the RFLCS, Blum et al. also consider the *longest arc-preserving common subsequence* (LAPCS) problem [15], where additional dependencies among characters must be respected in a solution, as well as the *longest common palindromic subsequence* (LCPS) problem [11], where the resulting sequence must also be a palindrome. All these LCS variants where solved by transforming instances to instances of the MIS problem. Moreover, the equivalent maximum clique problem of the complement of the conflict graph is solved heuristically by the LSCC-BMS solver as well as exactly by the LMC solver. Both solvers are currently among the leading solvers for the maximum clique problem.

In the literature LCS related problems with additional constraints are well known for almost 40 years and research in that field is still active due the practical relevance and computational difficulties. Besides RFLCS and the already mentioned LAPCS and LCPS problems other considered variants are, for instance, the constrained longest common subsequence problem [20] or the generalized constrained longest common subsequence problem [10]. For further problem variants we refer to survey papers such as [7].

The RFLCS problem in particular was tackled by several heuristic approaches [1,4,9]. The so far best heuristic is a *construct, merge, solve and adapt* (CMSA) metaheuristic combined with beam search as proposed by Blum and Blesa [5]. The authors showed that this approach can outperform other heuristics as well as the CPLEX solver applied to an ILP model of the RFCLS problem.

## 3 Problem Definition

The RFLCS problem considers a set of two input strings $S = \{s_1, s_2\}$ over a finite alphabet $\Sigma$. The goal is to find the longest subsequence which is common for both input strings $s_1$ and $s_2$ such that there is no character which occurs more than once. The character at position $i$ is denoted by $s[i]$. A matching $\boldsymbol{m} = (m_1, m_2)$ is a pair of positions s.t. $s_1[m_1] = s_2[m_2]$ and the corresponding character is denoted by $c(\boldsymbol{m}) = s_1[m_1]$. Hence, the character $c(\boldsymbol{m})$ of a matching $\boldsymbol{m}$ is a possible candidate to appear in a common sequence (CS). A matching $\boldsymbol{m}$ *dominates* a matching $\boldsymbol{n}$, denoted as $\boldsymbol{m} \succeq \boldsymbol{n}$, if $m_1 \leq n_1 \wedge m_2 \leq n_2$, meaning that in a possible CS $c(\boldsymbol{m})$ may appear before $c(\boldsymbol{n})$. Therefore, a CS can be represented by a sequence of matchings $(m_1, m_2, \ldots)$ s.t. $c(m_1), c(m_2), \ldots$ maps to the CS and each matching of the sequence dominates each subsequent matching of the sequence. This observation is important since relaxed MDDs for the RFLCS problem will encode such sequences of matchings. If for two matchings $\boldsymbol{m}$ and $\boldsymbol{n}$ neither $\boldsymbol{m} \succeq \boldsymbol{n}$ nor $\boldsymbol{n} \succeq \boldsymbol{m}$ holds then $c(\boldsymbol{m})$ and $c(\boldsymbol{n})$ cannot appear together in a CS which will be henceforth referred to as $\boldsymbol{m}$ and $\boldsymbol{n}$ are *in conflict*, denoted as $\boldsymbol{n} \curlyvee \boldsymbol{m}$. Figure 1a shows an example of a RFLCS instance with input strings $s_1 =$ `ABCDBA` and $s_2 =$ `ACBDBA` and an optimal solution of `ACDB`. In this

example, matching $\boldsymbol{m}_1$ dominates matching $\boldsymbol{m}_2$ and $\boldsymbol{m}_3$ whereas matching $\boldsymbol{m}_2$ is in conflict with matching $\boldsymbol{m}_3$.

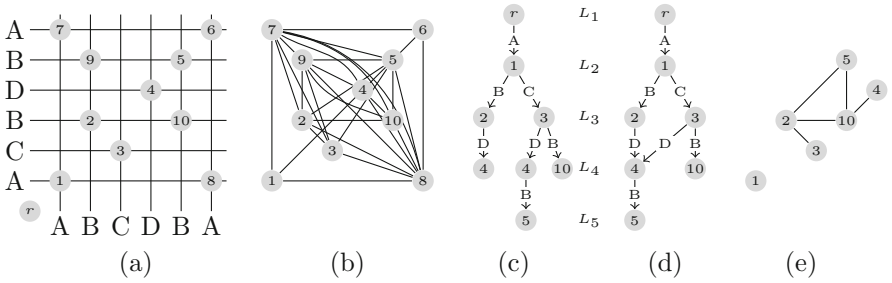


**Fig. 1.** (a) Example of a RFLCS instance with input strings $s_1 =$ `ABCDBA` and $s_2 =$ `ACBDBA`. Gray circles correspond to the matchings $M = \{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{10}\}$ of the instance. (b) Corresponding MIS instance. (c) Exact MDD $\mathcal{D}_M$ with $\text{mat}(\mathcal{D}_M) = \{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_5, \boldsymbol{m}_{10}\}$. The state of each node $u$ is partially indicated by $\boldsymbol{m}(u)$. (d) Relaxed MDD where nodes associated with matching $\boldsymbol{m}_4$ in layer $L_3$ are merged. (e) MIS instance obtained from matchings $\text{mat}(\mathcal{D}_M)$.

## 4 Integer Linear Program and Independent Set Model

An instance of the RFLCS problem can be solved by transforming it into an instance of the MIS problem. Thereby, each matching corresponds to a node of the underlying conflict graph of the MIS problem. An edge is added between two nodes if the corresponding matchings are in conflict or they refer to the same character; see Fig. 1b for an example. A solution of the MIS instance corresponds to a solution of the RFLCS instance and vice versa, since only matchings are selected that are not in conflict with each other and can therefore appear in the same CS and for each character there is at most one matching selected. The resulting CS can be derived from the set of selected matchings by a topological sort considering the domination relationship.

We solve the MIS instance by a corresponding ILP model. Let $M$ be the set of all matchings of the RFLCS instance and thus nodes of the MIS instance. We use a binary decision variable $x_{\boldsymbol{m}}$ for each matching $\boldsymbol{m} \in M$ indicating whether the matching is selected (=1) for the solution or not (=0). The model is:

$$\text{ILP}(M) = \max \quad \sum_{x_m \in M} x_{\boldsymbol{m}} \tag{1a}$$

$$\text{s.t.} \quad x_{\boldsymbol{m}} + x_{\boldsymbol{n}} \leq 1 \qquad \boldsymbol{m}, \boldsymbol{n} \in M : \boldsymbol{m} \curlyvee \boldsymbol{n} \tag{1b}$$

$$\sum_{x_m \in M_a} x_{\boldsymbol{m}} \leq 1 \qquad a \in \Sigma \tag{1c}$$

$$x_{\boldsymbol{m}} \in \{0, 1\} \qquad \boldsymbol{m} \in M \tag{1d}$$

The number of selected matchings is maximized. Inequalities (1b) ensure that the CS constraints are satisfied, i.e., no conflicting matchings are selected together. The *repetition-free* (RF) constraint is realized by Inequalities (1c), where set $M_a = \{\boldsymbol{m} \in M \mid c(\boldsymbol{m}) = a\}$ contains all matchings corresponding to the same character $a \in \Sigma$.

## 5 Decision Diagrams for the RFLCS Problem

We use a relaxed MDD to derive a reduced set of matchings $M' \subseteq M$ to subsequently solve the model ILP($M'$). Our approach compiles relaxed MDDs in an iterative way s.t. if set $M'$ is derived from a relaxed MDD then another relaxed MDD is compiled w.r.t. set $M'$ to possible derive an even smaller set $M'' \subseteq M'$. This procedure is repeated until some termination criterion is fulfilled.

A MDD w.r.t. a set of matchings $M$ is a directed acyclic multi-graph $\mathcal{D}_M = (V, A)$ with one root node $\mathbf{r}$. All nodes are partitioned into at most $|\Sigma| + 1$ layers $L_1, \ldots, L_{|\Sigma|+1}$, where $L_i$, $i > 0$ contains only nodes that are reachable from $\mathbf{r}$ over exactly $i - 1$ arcs and $L_1$ is a singleton containing only $\mathbf{r}$. An arc $\alpha = (u, v) \in A(\mathcal{D}_M)$ is always directed from a source node $u$ in some layer $L_i$ to a target node $v$ in a subsequent layer $L_{i+1}$. Each arc $\alpha$ is associated with a matching $\text{mat}(\alpha) \in M$ that represents the assignment of character $c(\text{mat}(\alpha)) \in \Sigma$ to the $i$-th position of a CS. For convenience we write $c(\alpha)$ for $c(\text{mat}(\alpha))$. Any directed path $\varphi = (\alpha_1, \alpha_2, \ldots)$ originating from $\mathbf{r}$ identifies a sequence of characters $(c(\alpha_1), c(\alpha_2), \ldots)$ and thus a (partial) solution. A node without any further outgoing arcs is a *sink node*. An *exact* MDD encodes precisely the set of all feasible solutions. Due to the NP-hardness of the RFLCS problem such exact MDDs tend to have exponential size.

Therefore we consider more compact *relaxed* MDDs which encode supersets of all feasible solutions. In such a relaxed MDD nodes of an exact MDD are superimposed (*merged*) so that at each layer a maximum allowed number of nodes, called width, is not exceeded. We do this merging in such a way that any path from the root still represents a CS, but RF constraints may be violated. The length of the longest path in such a relaxed MDD then represents an upper bound to the length of a RFLCS.

To compile a MDD, a DP formulation of the considered problem is usually the starting point [14]. Each node $u \in V(\mathcal{D}_M)$ is associated to a state of the DP formulation. For the RFLCS problem, the DP formulation is defined as follows. Consider for a matching $\boldsymbol{m} \in M$ the set $\text{D}_M(\boldsymbol{m}) = \{\boldsymbol{m}' \in M \setminus \{\boldsymbol{m}\} \mid \boldsymbol{m} \succeq \boldsymbol{m}'\}$ of possible successor matchings of $\boldsymbol{m}$ that may appear in the same CS after $\boldsymbol{m}$. Note that set $\text{D}_M(\boldsymbol{m})$ can be efficiently pre-computed for each $\boldsymbol{m} \in M$. Then a state $(\boldsymbol{m}(u), P(u), S(u))$ associated to node $u$ consists of

- a matching $\boldsymbol{m}(u)$ whose successor matchings $\text{D}_M(\boldsymbol{m}(u))$ represent the remaining matchings to consider further,
- set $P(u) \subseteq \Sigma$ containing all letters that may still be appended to the CS,
- set $S(u) \subseteq \Sigma$ containing all letters that appear on some paths from $\mathbf{r}$ to $u$.

The root state is $(\boldsymbol{m}_r, \Sigma, \emptyset)$ with the artificial matching $\boldsymbol{m}_r = (-1, -1)$, and all characters may be appended to it. Note that $\mathrm{D}_M(\boldsymbol{m}_r) = M$. An arc $\alpha = (u, v)$ corresponds to a transition from state $(\boldsymbol{m}(u), P(u), S(u))$ to state $(\boldsymbol{m}(v), P(v), S(v))$ that is achieved by appending character $c(\alpha)$ to the CS w.r.t. to the remaining matchings $\mathrm{D}_M(\boldsymbol{m}(u))$. Instead of considering all matchings from $\mathrm{D}_M(\boldsymbol{m}(u))$ as possible outgoing transitions, we consider only matchings that can appear *directly* after $\boldsymbol{m}(u)$ in a longest CS, i.e., matchings from the subset $\mathrm{ND}(u) = \{\boldsymbol{m}' \in \mathrm{D}_M(\boldsymbol{m}(u)) \mid \nexists \boldsymbol{m}'' \in \mathrm{D}_M(\boldsymbol{m}(u)) \setminus \{\boldsymbol{m}'\} : c(\boldsymbol{m}'') \notin S(u) \wedge \boldsymbol{m}'' \succeq \boldsymbol{m}'\}$ which are not dominated by any other matching in $\mathrm{D}_M(\boldsymbol{m}(u))$. The transition function to obtain the successor state $(\boldsymbol{m}(v), P(v), S(v))$ by considering matching $\mathrm{mat}(\alpha) \in \mathrm{D}_M(u)$ is defined as

$$\tau((\boldsymbol{m}(u), P(u), S(u)), \mathrm{mat}(\alpha)) = \tag{2}$$
$$\begin{cases} (\mathrm{mat}(\alpha), P(u) \setminus \{c(\alpha)\}, S(u) \cup \{c(\alpha)\}) & \text{if } c(\alpha) \in P(u) \wedge \mathrm{mat}(\alpha) \in \mathrm{ND}(u) \\ \hat{0} & \text{otherwise} \end{cases}$$

where $\hat{0}$ represents the infeasible state. Note that no node $\hat{0}$ is created in $\mathcal{D}_M$ and the respective arcs are also skipped.

Moreover, a state $(\boldsymbol{m}(u), P(u), S(u))$ may be replaced by a *strengthened state* $(\boldsymbol{m}(u), P'(u), S(u))$, where $P'(u) = \{a \in P(u) \mid \exists \boldsymbol{m'} \in \mathrm{D}_M(\boldsymbol{m}(u)) : a = c(\boldsymbol{m}')\} \subset P(u)$ without excluding any feasible solutions.

So far, we considered exact MDDs. For relaxed MDDs we have to define a state merger which computes the state of merged nodes. To still encode all feasible CSs in the relaxed MDD, only nodes of the same layer and with the same associated matching are merged. Let $U$ be a subset of nodes s.t. all nodes are associated to matching $\boldsymbol{n}$, i.e. $\forall u \in U : \boldsymbol{m}(u) = \boldsymbol{n}$, then an appropriate state merger is $\oplus(U) = \big(\boldsymbol{n}, \bigcup_{u \in U} P(u), \bigcup_{u \in U} S(u)\big)$. Since we restrict the state merger to nodes with the same associated matching, the possibilities to reduce the size of the relaxed MDD are also limited. However, since $|M|$ is at most the product $|s_1|\,|s_2|$ of the lengths of the two input strings $s_1$ and $s_2$, the size of each layer is still polynomially bounded by $O(|s_1|\,|s_2|)$.

Let $\mathrm{mat}(\mathcal{D}_M) = \{\boldsymbol{m}(u) \mid u \in V(\mathcal{D}_M) \setminus \{\mathbf{r}\}\} \subseteq M$ be the set of matchings derived from $\mathcal{D}_M$. To see that $\mathrm{mat}(\mathcal{D}_M)$ is indeed a feasible set of matchings to solve the model $\mathrm{ILP}(\mathrm{mat}(\mathcal{D}_M))$ from Sect. 4, remember that each path from $\mathbf{r}$ in $\mathcal{D}_M$ encodes a feasible CS. Hence, each such path can also be described as a sequence of matchings from $M$. In particular this is true for the matchings of a RFLCS, which must be therefore also contained in $\mathrm{mat}(\mathcal{D}_M)$.

*Problem Specific Upper Bounds.* To reduce $\mathrm{mat}(\mathcal{D}_M)$ further we filter arcs and nodes based on sub-optimality. The idea is to compute for each node $u$ an upper bound $Z^{\mathrm{ub}}(u)$ on the number of characters that can appear in a common subsequence after the character $c(\boldsymbol{m}(u))$ of matching $\boldsymbol{m}(u)$. Then we can prune each

**Algorithm 1.** Incremental Refinement

**Input:** set of matchings $M$, lower bound $lb$, maximum width threshold $W$
$s^{\text{best}} \leftarrow \varepsilon$
construct initial relaxed decision diagram $\mathcal{D}_M$
**repeat**
  filter-bottom-up($\mathcal{D}_M$, max($lb$, $|s^{\text{best}}|$))
  $s^{\text{rfcs}} \leftarrow$ derive-primal-solution($\mathcal{D}_M$) and update $s^{\text{best}}$ **if** $|s^{\text{best}}| < |s^{\text{rfcs}}|$
**until** no new best solution $s^{\text{best}}$ found
determine priority ranking $a^*_1, \ldots, a^*_{|\Sigma|}$ of all characters
**repeat**
  $M \leftarrow \text{mat}(\mathcal{D}_M)$
  **for** $i \leftarrow 1$ to $|\Sigma| + 1$ **do**
    refine($L_i$, $a^*_1, \ldots, a^*_{|\Sigma|}$, $W$)
    filter arcs between $L_i$ and $L_{i+1}$
  **end for**
  **repeat**
    filter-bottom-up($\mathcal{D}_M$, max($lb$, $|s^{\text{best}}|$))
    $s^{\text{rfcs}} \leftarrow$ derive-primal-solution($\mathcal{D}_M$) and update $s^{\text{best}}$ **if** $|s^{\text{best}}| < |s^{\text{rfcs}}|$
  **until** no new best solution $s^{\text{best}}$ found
**until** $|M| < |\text{mat}(\mathcal{D}_M)|$
**return** ($\mathcal{D}_M$, $s^{\text{best}}$)

node $u$ in the relaxed MDD where $Z^{\text{lp}}(u) + Z^{\text{ub}}(u) < lb$ holds, where $lb$ is a known lower bound on the length of the RFLCS and $Z^{\text{lp}}(u)$ is the length of the longest path from $\mathbf{r}$ to $u$. We compute the upper bound for each node $u$ by

$$Z^{\text{ub}}(u) = \min\{|P(u)|, \text{UB}^{\text{lcs}}(\boldsymbol{m}(u)), \max_{\alpha=(w,u)} \{Z^{\text{ub}}(w) - 1\}, Z^{\text{lp}\uparrow}(u)\}. \quad (3)$$

The first term takes the number of characters into account that can still be appended to the CS after matching $\boldsymbol{m}(u)$. The second term $\text{UB}^{\text{lcs}}(\boldsymbol{m}(u)) = \text{LCS}(m_1, m_2)$ is based on DP and computes the length of the longest common subsequence from matching $\boldsymbol{m}(u)$ onward. Note that this bound can be obtained in constant time by using a data structure known as scoring matrix, which can be computed during preprocessing for two input strings in $O(|s_1|\,|s_2|)$ time [6]. The third term takes the upper bounds from the parent nodes of $u$ into account. Finally, the last term corresponds to the length of the longest path from $u$ to any sink node in the relaxed MDD. Note that this term is only available if the whole relaxed MDD is already compiled.

## 6 Incremental Refinement

Our approach to compile a relaxed MDD $\mathcal{D}_M$ w.r.t. matching set $M$ for the RFLCS problem is based on the *incremental refinement* (IR) algorithm from Cire and van Hoeve [12] for sequencing problems. Since $\mathcal{D}_M$ considers the CS constraints exactly and only relaxes the RF constraint, paths in $\mathcal{D}_M$ originating from $\mathbf{r}$ will correspond to CSs where characters may appear more than once. We use the ideas from [12] to ensure at least for some characters that they occur at most once at each path for refining $\mathcal{D}_M$. Cire and van Hoeve [12] showed that the size of a given relaxed MDD will be at most doubled to establish this property for one more character.

The algorithm applies repeatedly two major steps—filtering and refinement—until some termination condition is fulfilled. Let $a_1^*, a_2^*, \ldots, a_{|\Sigma|}^*$ be a ranking of the characters in $\Sigma$ s.t. $a_1^*$ is the most important character to appear at most once at each path in $\mathcal{D}_M$ to get a strong relaxation. The following refinement step is applied layer by layer starting with $L_1$: For each character $a^* = a_1^*, a_2^*, \ldots, a_{|\Sigma|}^*$ we identify nodes $u$ s.t. $a^* \in P(u) \cap S(u)$ and split them into two new nodes $u_1$ and $u_2$ where an incoming arc $\alpha = (v, u)$ is redirected to $u_1$ if $a^* \in P(u) \setminus \{c(\alpha)\}$ and to $u_2$ otherwise. All outgoing arcs are replicated for both nodes $u_1$ and $u_2$. We do this as long as the size of the layer is below a maximum width threshold $W$. For more details and a correctness proof in the context of sequencing problems see [12]. Due to the splitting of nodes the corresponding states may be changed and some of the outgoing arcs from the current layer to the next layer may become infeasible. Those arcs are filtered for each layer after the refinement step finishes. Algorithm 1 shows this at lines 12 and 13. The algorithm terminates if set $\mathrm{mat}(\mathcal{D}_M)$ could not be further reduced by the previously applied refinement/filtering round. The other main parts of the algorithm are:

*Initial Relaxed MDD.* The IR algorithm starts with an initial relaxed MDD. Usually, this initial relaxed MDD is a naive one of width one, i.e., a relaxed MDD with just a single node at each layer. However, in our case we want to respect the CS constraints and only superimpose states that correspond to the same matching. Therefore we compile the initial $\mathcal{D}_M$ layer-by-layer in a top-down approach. At each layer $L_i$, $i \geq 1$, we expand all nodes using the transition function (2), thus creating for each feasible transition a corresponding node in $L_{i+1}$ and adding the corresponding arc if the node is not sub-optimal according to Eq. 3. Then all nodes in $L_{i+1}$ with the same corresponding matching are merged. Since no feasible CS can be longer than the upper bound $Z^{\mathrm{ub}}(\mathbf{r})$, the compilation of $\mathcal{D}_M$ stops at the $(Z^{\mathrm{ub}}(\mathbf{r}) + 1)$-th layer.

*Character Ranking for Refinement.* To determine priorities for the characters we use some structural information obtained from the initial MDD. For this purpose let $\mathrm{All}^{\uparrow}(u)$ for each node $u \in V(\mathcal{D}_M)$ be the set of characters that appear on all paths from node $u$ to a sink node. Note that set $\mathrm{All}^{\uparrow}(u)$ can be efficiently computed in a recursive way by a single bottom-up pass. If there exists a node $v$ with an incoming arc $\alpha = (u, v)$ s.t. $c(\alpha) \in \mathrm{All}^{\uparrow}(v)$ holds, then each path

originating from $\mathbf{r}$ and leading to any sink node will be infeasible if the path traverses $\alpha$ since character $c(\alpha)$ will appear more than once in a corresponding CS, i.e., the RF constraint will be violated. In [12] such arcs could be safely removed without also removing any feasible solution from the relaxed MDD. In our case this is not possible since solutions have arbitrary length and the path from $\mathbf{r}$ to $v$ could still correspond to a complete feasible solution. However, we can use these violations to determine for which character it is most important to appear on all paths at most once to get a strong relaxation. Hence, we count for each character how often such a violation occurs in $\mathcal{D}_M$ and sort the characters according to non-increasing numbers of violations. Ties are resolved by preferring characters that appear in more matchings.

*Filtering and Deriving New Primal Solutions.* Lines 4–7 and 15–18 perform the following steps. First the function filter-bottom-up performs a single bottom up pass where for each node $u$ the length of the longest path $Z^{\text{lp}\uparrow}(u)$ from $u$ to any sink node is computed and the upper bound $Z^{\text{ub}}(u)$ is updated accordingly. If $Z^{\text{lp}}(u) + Z^{\text{ub}}(u) < lb$ then node $u$ and all incident arcs are removed from $\mathcal{D}_M$.

After filtering we try to derive from $\mathcal{D}_M$ a new best heuristic solution. Since each path in $\mathcal{D}_M$ originating from $\mathbf{r}$ corresponds to a CS, we can derive a RFCS by removing duplicate letters. This is done in two steps. First, a bottom-up pass is performed where primal bounds are computed: For each node $u \in \mathcal{D}_M$ we recursively determine set $B^{\uparrow}(u) = B^{\uparrow}(v) \cup \{c(\alpha')\}$ where outgoing arc $\alpha' = (u, v)$ maximizes $\alpha' = \arg\max_{\alpha=(u,v)} |B^{\uparrow}(v) \cup \{c(\alpha)\}|$. Ties are resolved by sticking at the first arc that maximizes the expression. If $u$ has no outgoing arcs then $B^{\uparrow}(u) = \emptyset$. Note that $|B^{\uparrow}(\mathbf{r})|$ is a valid primal bound on the RFLCS problem, since only the union is taken to compute $B^{\uparrow}(.)$. To improve this bound further, the second step performs a top-down pass where set $B^{\downarrow}(v)$ is recursively computed for each node $v$ using the information of the precisely computed set $B^{\uparrow}(v)$. Hence, $B^{\downarrow}(v) = B^{\downarrow}(u) \cup \{c(\alpha')\}$ where incoming arc $\alpha' = (u, v)$ maximizes $\alpha' = \arg\max_{\alpha=(u,v)} |B^{\downarrow}(u) \cup \{c(\alpha)\} \cup B^{\uparrow}(v)|$ using $|B^{\downarrow}(v) \cup \{c(\alpha)\}|$ as tie breaking criterion. A sink node $v'$ that maximizes $|B^{\downarrow}(v')|$ then provides the strongest primal bound. A respective RFCS is derived by going from $v'$ backwards to $\mathbf{r}$, skipping any character that already occurred along the path.

If a new best heuristic solution could be obtained in this way then the filter-bottom-up step is repeated and we try again to obtain a new best heuristic solution.

*Main Procedure:* Algorithm 2 shows the main procedure to solve an instance of the RFLCS problem. As input the algorithm takes the set of input strings $S$, a possibly known lower bound on the RFLCS length or zero, and the maximum width threshold $W$ for the relaxed MDDs. The original set of matchings $M$ reduced by performing iteratively the following steps. The first step processes the input strings $s_1$ and $s_2$ by removing characters that have no associated matching in $M$ and characters that appear immediately one after the other in the input strings. For example, if character $a \in \Sigma$ appears in an input string at both position $i$ and $i + 1$ then $a$ can be removed from $i + 1$ without removing

**Algorithm 2.** Main Procedure for solving the RFLCS problem

**Input:** input strings $S$, lower bound $lb$, maximum width threshold $W$
$s^{\text{best}} \leftarrow \varepsilon$
derive original $M$ w.r.t. $S$
**repeat**
  process $S$ w.r.t. $M$
  $M \leftarrow \{\boldsymbol{m} \in M \mid \text{UB}^{\text{BLUM}}(\boldsymbol{m}) \geq \max(lb, |s^{\text{best}}|)\}$
  $(\mathcal{D}_M, s^{\text{rfcs}}) \leftarrow \text{IR}(M, \max(lb, |s^{\text{best}}|), W)$
  $M \leftarrow \text{mat}(\mathcal{D}_M)$ and update $s^{\text{best}}$ **if** $|s^{\text{rfcs}}| > |s^{\text{best}}|$
  **return** $s^{\text{best}}$ **if** $s^{\text{best}} = Z^{\text{lp}\uparrow}(\mathbf{r})$
**until** no characters can be removed from input strings
$s^{\text{ilp}} \leftarrow$ solve ILP($M$)
update $s^{\text{best}}$ **if** $|s^{\text{ilp}}| > |s^{\text{best}}|$
**return** $s^{\text{best}}$

any feasible solution. Furthermore, if the pattern $abab$ with $a, b \in \Sigma$ has been discovered in one of the input strings then the last $b$ can be removed from the input string due to the RF constraint. Next, $M$ is reduced by removing matchings $\boldsymbol{m} \in M$ where the upper bound used in [6], denoted by $\text{UB}^{\text{BLUM}}(\boldsymbol{m})$ is lower than our currently best primal bound. This upper bound is based on the first two terms in Eq. (3), i.e., on the number of characters that can appear in a RFCS that contains $\boldsymbol{m} \in M$ and on the length of the LCS that contains $\boldsymbol{m}$. Note that the difference to Eq. (3) is that $\text{UB}^{\text{BLUM}}(\boldsymbol{m})$ is an upper bound on the length of a complete RFCS containing $\boldsymbol{m}$ wheres Eq. (3) describes an upper bound on the remaining part from $\boldsymbol{m}$ onward. With this reduced set $M$ we compile a relaxed MDD $\mathcal{D}_M$. If the length of the hereby derived RFCS $s^{\text{rfcs}}$ is equal to the longest path in $\mathcal{D}_M$ then $s^{\text{rfcs}}$ is an optimal solution and the algorithm terminates. Otherwise, if due to the reduced set $\text{mat}(\mathcal{D}_M)$ further characters can be removed from $s_1$ and $s_2$ then we repeat the procedure until no further characters can be removed. Note that since the size of the input strings are reduced at each iteration also $\text{UB}^{\text{BLUM}}(\boldsymbol{m})$ changes, which may further reduce set $M$. Finally the ILP model from Sect. 4 is solved for set $M$.

## 7 Experimental Results

To test and compare our approach we used two benchmark sets from [4]. The first set, Set1, consists of 1680 randomly generated instances. For each combination of the input string lengths $n \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$ and the alphabet sizes $|\Sigma| \in \{\frac{n}{8}, \frac{n}{4}, \frac{3n}{8}, \frac{n}{2}, \frac{5n}{8}, \frac{3n}{4}, \frac{7n}{8}\}$ there are 30 instances. The second set, Set2, consists of 30 randomly generated instances for each combination of the alphabet size $|\Sigma| \in \{4, 8, 16, 32, 64, 128, 256, 512\}$ and the maximal repetition of each character, reps $\in \{3, 4, 5, 6, 7, 8\}$. This set has a total of 1440 instances.

The algorithms were implemented using GNU C++ 5.4.1. All tests were executed on a single core of an Intel Xeon E5649 with 2.53 GHz and 16 GB RAM. The ILP model from Sect. 4 was solved with CPLEX 12.7 with a CPU-time limit of 3600 s. For Algorithm 2, henceforth denoted as MDD+CPLEX, the

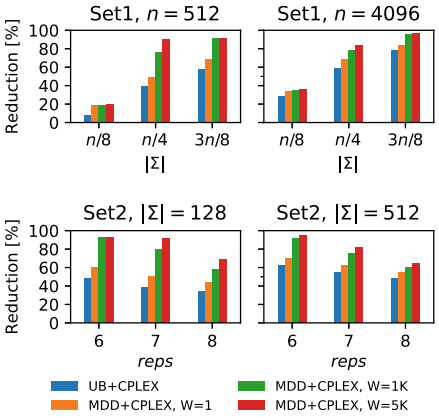


**Fig. 2.** Average reduction of matchings obtained from UB+CPLEX and MDD+CPLEX with different maximum width thresholds $W$.

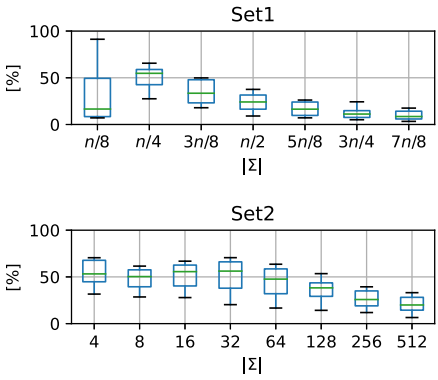


**Fig. 3.** Average difference between the obtained reduction rates of UB+CPLEX and MDD+CPLEX with $W = 5000$.

maximum width threshold was set to $W = 5000$. This value was determined in preliminary experiments s.t. set $M$ could be reduced as much as possible and as many instances as possible can be solved to optimality within the memory limit of 16 GB. MDD+CPLEX is compared to the approach from Blum et al. [6], henceforth denoted as UB+CPLEX, where the ILP($M'$) model is solved with the reduced set of matchings $M' = \{\boldsymbol{m} \in M \mid \text{UB}^{\text{BLUM}}(\boldsymbol{m}) < lb\}$. Both approaches use the lengths of the currently best known solutions from the literature as initial lower bound $lb$. Note that the compiled relaxed MDDs from Algorithm 1 are not strictly limited to $W$ since the initial relaxed MDD could already contain layers that contain more nodes than $W$. However, such layers are not further refined during the compilation.

The average reduction rate of matchings from $M$ is shown in Fig. 2 for 12 instance classes by means of bar plots. The first bar corresponds always to the UB+CPLEX approach whereas the next three bars corresponds to the MDD+CPLEX approach with different values for $W \in \{1, 1000, 5000\}$. Note that $W = 1$ means that only the initial relaxed MDD is compiled and no further refinement will take place. As expected, the obtained reduction rate increases with $W$. The boxplots in Fig. 3 report the average difference $\text{red}_{\text{MDD}} - \text{red}_{\text{UB}}$ between the average reduction rate $\text{red}_{\text{UB}}$ obtained from UB+CPLEX and $\text{red}_{\text{MDD}}$ obtained from MDD+CPLEX in percentage points aggregated over the ratio between $n$ and $|\Sigma|$ in case of SET1 and over $|\Sigma|$ in case of SET2. On average the MDD+CPLEX approach is able to reduce the original set of matchings by more than 25.79% and 41.28% as UB+CPLEX does, for SET1 and SET2, respectively.

Detailed aggregated results are presented in Tables 1 and 2 where the first two columns show the instance characteristics and the third column shows the average length of the so far best known solution from the literature. Columns obj report for each tested approach the average length of the best obtained solutions. In case of MDD+CLPEX these solutions are either those obtained from the ILP model or the ones found during the compilation of the relaxed MDDs. In case of UB+CPLEX a "-" symbol indicates that CPLEX was not able at all to derive a primal solution within the time and memory limits. Average optimality gaps, shown in columns gap, are calculated by $100\% \times (\text{ub} - \text{obj})/\text{ub}$ where ub is for each approach the best obtained upper bound. In case of MDD+CPLEX this is either the upper bound obtained from the ILP model or the length of the longest path from a compiled relaxed MDD. Columns $t_{\text{prep}}$ list average preprocessing times in CPU seconds including the computation of the reduced set of matchings $M$ (see Algorithm 2, Line 10). Columns $t_{\text{tot}}$ list average total computation times in CPU seconds until the algorithm terminates including $t_{\text{prep}}$ plus the time CPLEX needs and columns #opt report the total numbers of instances solved to optimality. In case of MDD+CPLEX the second number corresponds to the number of instances where optimality could already be proven by the compiled relaxed MDD at Line 9 in Algorithm 2. Hence, the number of times where it was not required to solve the ILP model at all. Average reduction rates of the original set of matchings are reported by columns red.

Regarding the number of instances solved to proven optimality, note that already UB+CPLEX was quite successful with a total of 90.26%. More precisely, 1489 out of 1680 instances from Set1 and 1327 out of 1440 instances of Set2 could be solved to proven optimality by UB+CPLEX. Nevertheless, MDD+CPLEX is able to solve significantly more instances to proven optimality: 1541 instances from Set1 and 1381 instances of Set2, and thus a total of 93.65%. Moreover, in 90.90% of all instances it was not necessary to solve the ILP model at all, since Algorithm 2 terminated early at Line 9. Hence, the obtained upper bound from the compiled relaxed MDD was equal to the length of the currently best found solution in these cases. Concerning the computation times, the UB+CPLEX approach was on average in only two cases faster then the MDD+CPLEX approach regarding benchmark set Set1 and only in one case regarding benchmark set Set2. Finally, the MDD+CPLEX approach is able to obtain in 135 cases better results than the currently best-known-solutions from the literature. For each considered problem class, MDD+CPLEX is able to provide on average better results than UB+CPLEX.

**Table 1.** Results on SET1 instances.

| $\|\Sigma\|$ | $n$ | so far | UB+CPLEX | | | | | MDD+CPLEX | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | best. | obj | gap [%] | $t$ [s] | #opt | red [%] | obj | gap [%] | $t_{\text{prep}}$ [s] | $t_{\text{tot}}$ [s] | #opt | red [%] |
| $n/8$ | 32 | **4.00** | **4.00** | **0.00** | 0.16 | **30** | 3.72 | **4.00** | **0.00** | <0.01 | **<0.01** | **30/30** | 94.94 |
| | 64 | **8.00** | **8.00** | **0.00** | 1.69 | **30** | 4.67 | **8.00** | **0.00** | <0.01 | **<0.01** | **30/30** | 78.23 |
| | 128 | **16.00** | **16.00** | **0.00** | 11.46 | **30** | 4.80 | **16.00** | **0.00** | 0.04 | **0.04** | **30/30** | 44.45 |
| | 256 | **31.97** | **31.97** | **0.00** | 298.59 | **30** | 5.47 | **31.97** | **0.00** | 0.42 | **0.42** | **30/30** | 25.47 |
| | 512 | **63.90** | 32.30 | 49.50 | 3615.63 | 0 | 8.16 | 62.80 | **1.82** | 60.82 | **3434.53** | **2/2** | 20.29 |
| | 1024 | **116.30** | 0.00 | 100.00 | **3676.46** | **0** | 15.89 | 113.47 | **11.22** | 271.15 | 3946.52 | **0/0** | 23.34 |
| | 2048 | 185.07 | - | - | **0.48** | **0** | 22.45 | **186.23** | **16.29** | 1134.30 | 4135.18 | **0/0** | 28.49 |
| | 4096 | 284.80 | - | - | **2.04** | **0** | 28.71 | **292.03** | **11.05** | 4297.07 | 4297.07 | **0/0** | 36.59 |
| $n/4$ | 32 | **7.83** | **7.83** | **0.00** | 0.03 | **30** | 22.13 | **7.83** | **0.00** | <0.01 | **<0.01** | **30/30** | 79.54 |
| | 64 | **14.67** | **14.67** | **0.00** | 0.21 | **30** | 21.08 | **14.67** | **0.00** | <0.01 | **<0.01** | **30/30** | 84.06 |
| | 128 | **25.93** | **25.93** | **0.00** | 4.82 | **30** | 21.76 | **25.93** | **0.00** | 0.47 | **0.47** | **30/30** | 87.34 |
| | 256 | **43.97** | **43.97** | **0.00** | 22.93 | **30** | 33.17 | **43.97** | **0.00** | 4.61 | **4.61** | **30/30** | 90.67 |
| | 512 | **68.57** | **68.57** | **0.00** | 389.04 | **30** | 39.54 | **68.57** | **0.00** | 28.43 | **38.93** | **30/28** | 91.01 |
| | 1024 | **105.07** | 104.97 | 0.96 | 2064.94 | 21 | 48.42 | **105.07** | **0.00** | 122.81 | **188.27** | **30/24** | 91.56 |
| | 2048 | 155.73 | 120.47 | 26.52 | 3314.28 | 4 | 56.72 | **156.87** | **0.37** | 576.39 | **1571.46** | **24/15** | 85.91 |
| | 4096 | 227.23 | 12.77 | 95.05 | **3650.80** | 0 | 59.48 | **230.37** | **0.73** | 1996.26 | 4036.48 | **15/6** | 83.99 |
| $3n/8$ | 32 | **8.77** | **8.77** | **0.00** | 0.02 | **30** | 30.44 | **8.77** | **0.00** | <0.01 | **<0.01** | **30/30** | 78.03 |
| | 64 | **15.53** | **15.53** | **0.00** | 0.06 | **30** | 32.30 | **15.53** | **0.00** | <0.01 | **<0.01** | **30/30** | 81.16 |
| | 128 | **24.90** | **24.90** | **0.00** | 0.70 | **30** | 36.58 | **24.90** | **0.00** | 0.06 | **0.06** | **30/30** | 86.44 |
| | 256 | **39.97** | **39.97** | **0.00** | 1.52 | **30** | 55.50 | **39.97** | **0.00** | 0.50 | **0.50** | **30/30** | 89.20 |
| | 512 | **59.97** | **59.97** | **0.00** | 17.62 | **30** | 57.92 | **59.97** | **0.00** | 4.26 | **4.26** | **30/30** | 91.16 |
| | 1024 | **90.73** | **90.73** | **0.00** | 29.12 | **30** | 70.02 | **90.73** | **0.00** | 13.59 | **13.59** | **30/30** | 93.59 |
| | 2048 | 131.13 | **131.17** | 0.05 | 476.47 | 29 | 72.92 | **131.17** | **0.00** | 50.39 | **50.39** | **30/30** | 94.79 |
| | 4096 | 193.20 | 192.77 | 0.50 | 1030.16 | 25 | 78.79 | **193.37** | **0.00** | 163.90 | **163.99** | **30/29** | 96.65 |
| $n/2$ | 32 | **8.87** | **8.87** | **0.00** | 0.01 | **30** | 37.51 | **8.87** | **0.00** | <0.01 | **<0.01** | **30/30** | 75.08 |
| | 64 | **14.80** | **14.80** | **0.00** | 0.02 | **30** | 46.60 | **14.80** | **0.00** | <0.01 | **<0.01** | **30/30** | 81.22 |
| | 128 | **22.93** | **22.93** | **0.00** | 0.08 | **30** | 52.04 | **22.93** | **0.00** | 0.01 | **0.01** | **30/30** | 82.57 |
| | 256 | **35.20** | **35.20** | **0.00** | 0.46 | **30** | 60.22 | **35.20** | **0.00** | 0.07 | **0.07** | **30/30** | 87.37 |
| | 512 | **53.13** | **53.13** | **0.00** | 2.72 | **30** | 69.40 | **53.13** | **0.00** | 0.92 | **0.92** | **30/30** | 90.49 |
| | 1024 | **79.13** | **79.13** | **0.00** | 7.38 | **30** | 75.93 | **79.13** | **0.00** | 3.45 | **3.45** | **30/30** | 93.05 |
| | 2048 | **115.70** | **115.70** | **0.00** | 21.32 | **30** | 80.40 | **115.70** | **0.00** | 19.65 | **19.65** | **30/30** | 94.59 |
| | 4096 | **167.97** | **167.97** | **0.00** | 93.68 | **30** | 86.74 | **167.97** | **0.00** | 26.89 | **26.89** | **30/30** | 95.84 |
| $5n/8$ | 32 | **8.60** | **8.60** | **0.00** | 0.01 | **30** | 46.29 | **8.60** | **0.00** | <0.01 | **<0.01** | **30/30** | 72.45 |
| | 64 | **13.30** | **13.30** | **0.00** | 0.01 | **30** | 52.75 | **13.30** | **0.00** | <0.01 | **<0.01** | **30/30** | 78.35 |
| | 128 | **21.20** | **21.20** | **0.00** | 0.03 | **30** | 59.95 | **21.20** | **0.00** | <0.01 | **<0.01** | **30/30** | 83.47 |
| | 256 | **32.53** | **32.53** | **0.00** | 0.11 | **30** | 67.54 | **32.53** | **0.00** | 0.02 | **0.02** | **30/30** | 86.36 |
| | 512 | **47.83** | **47.83** | **0.00** | 0.61 | **30** | 74.69 | **47.83** | **0.00** | 0.10 | **0.10** | **30/30** | 88.68 |
| | 1024 | **70.20** | **70.20** | **0.00** | 1.12 | **30** | 81.45 | **70.20** | **0.00** | 0.76 | **0.76** | **30/30** | 91.35 |
| | 2048 | **103.97** | **103.97** | **0.00** | 4.66 | **30** | 84.96 | **103.97** | **0.00** | 3.27 | **3.27** | **30/30** | 93.98 |
| | 4096 | **150.57** | **150.57** | **0.00** | 16.11 | **30** | 88.65 | **150.57** | **0.00** | 10.26 | **10.26** | **30/30** | 95.77 |
| $3n/4$ | 32 | **8.17** | **8.17** | **0.00** | 0.01 | **30** | 47.56 | **8.17** | **0.00** | <0.01 | **<0.01** | **30/30** | 71.83 |
| | 64 | **12.53** | **12.53** | **0.00** | 0.01 | **30** | 53.92 | **12.53** | **0.00** | <0.01 | **<0.01** | **30/30** | 71.72 |
| | 128 | **19.70** | **19.70** | **0.00** | 0.02 | **30** | 65.68 | **19.70** | **0.00** | <0.01 | **<0.01** | **30/30** | 79.50 |
| | 256 | **29.97** | **29.97** | **0.00** | 0.04 | **30** | 72.89 | **29.97** | **0.00** | 0.01 | **0.01** | **30/30** | 84.41 |
| | 512 | **44.57** | **44.57** | **0.00** | 0.26 | **30** | 77.45 | **44.57** | **0.00** | 0.03 | **0.03** | **30/30** | 88.28 |
| | 1024 | **65.20** | **65.20** | **0.00** | 0.53 | **30** | 83.86 | **65.20** | **0.00** | 0.26 | **0.26** | **30/30** | 92.07 |
| | 2048 | **94.67** | **94.67** | **0.00** | 1.45 | **30** | 88.57 | **94.67** | **0.00** | 0.51 | **0.51** | **30/30** | 94.18 |
| | 4096 | **136.77** | **136.77** | **0.00** | 6.06 | **30** | 90.14 | **136.77** | **0.00** | 4.19 | **4.19** | **30/30** | 95.22 |
| $7n/8$ | 32 | **7.67** | **7.67** | **0.00** | 0.01 | **30** | 47.37 | **7.67** | **0.00** | <0.01 | **<0.01** | **30/30** | 64.92 |
| | 64 | **11.57** | **11.57** | **0.00** | 0.01 | **30** | 56.15 | **11.57** | **0.00** | <0.01 | **<0.01** | **30/30** | 73.63 |
| | 128 | **18.40** | **18.40** | **0.00** | 0.02 | **30** | 63.40 | **18.40** | **0.00** | <0.01 | **<0.01** | **30/30** | 76.54 |
| | 256 | **27.80** | **27.80** | **0.00** | 0.03 | **30** | 74.05 | **27.80** | **0.00** | 0.01 | **0.01** | **30/30** | 84.25 |
| | 512 | **40.60** | **40.60** | **0.00** | 0.09 | **30** | 80.25 | **40.60** | **0.00** | 0.03 | **0.03** | **30/30** | 87.14 |
| | 1024 | **60.57** | **60.57** | **0.00** | 0.47 | **30** | 85.41 | **60.57** | **0.00** | 0.11 | **0.11** | **30/30** | 91.16 |
| | 2048 | **88.00** | **88.00** | **0.00** | 2.54 | **30** | 85.93 | **88.00** | **0.00** | 0.63 | **0.63** | **30/30** | 91.89 |
| | 4096 | **127.37** | **127.37** | **0.00** | 4.76 | **30** | 91.37 | **127.37** | **0.00** | 2.17 | **2.17** | **30/30** | 94.71 |

**Table 2.** Results on SET2 instances.

| $\|\Sigma\|$ | reps | so far | UB+CPLEX | | | | | MDD+CPLEX | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | best | obj | gap [%] | $t$ [s] | #opt | red [%] | obj | gap [%] | $t_{\text{prep}}$ [s] | $t_{\text{tot}}$ [s] | #opt | red [%] |
| 4 | 3 | **3.47** | **3.47** | **0.00** | **<0.01** | **30** | 34.16 | **3.47** | **0.00** | <0.01 | **<0.01** | **30/30** | 65.78 |
| | 4 | **3.77** | **3.77** | **0.00** | **<0.01** | **30** | 32.24 | **3.77** | **0.00** | <0.01 | **<0.01** | **30/30** | 76.59 |
| | 5 | **3.83** | **3.83** | **0.00** | **<0.01** | **30** | 35.31 | **3.83** | **0.00** | <0.01 | **<0.01** | **30/30** | 81.44 |
| | 6 | **3.90** | **3.90** | **0.00** | 0.01 | **30** | 25.62 | **3.90** | **0.00** | <0.01 | **<0.01** | **30/30** | 85.92 |
| | 7 | **3.97** | **3.97** | **0.00** | 0.02 | **30** | 18.34 | **3.97** | **0.00** | <0.01 | **<0.01** | **30/30** | 88.59 |
| | 8 | **3.97** | **3.97** | **0.00** | 0.02 | **30** | 19.01 | **3.97** | **0.00** | <0.01 | **<0.01** | **30/30** | 89.54 |
| 8 | 3 | **6.23** | **6.23** | **0.00** | **<0.01** | **30** | 38.40 | **6.23** | **0.00** | <0.01 | **<0.01** | **30/30** | 66.94 |
| | 4 | **6.87** | **6.87** | **0.00** | 0.01 | **30** | 34.77 | **6.87** | **0.00** | <0.01 | **<0.01** | **30/30** | 71.70 |
| | 5 | **7.40** | **7.40** | **0.00** | 0.02 | **30** | 33.12 | **7.40** | **0.00** | <0.01 | **<0.01** | **30/30** | 80.10 |
| | 6 | **7.53** | **7.53** | **0.00** | 0.02 | **30** | 25.51 | **7.53** | **0.00** | <0.01 | **<0.01** | **30/30** | 79.22 |
| | 7 | **7.70** | **7.70** | **0.00** | 0.03 | **30** | 21.51 | **7.70** | **0.00** | <0.01 | **<0.01** | **30/30** | 80.36 |
| | 8 | **7.77** | **7.77** | **0.00** | 0.07 | **30** | 19.09 | **7.77** | **0.00** | <0.01 | **<0.01** | **30/30** | 80.65 |
| 16 | 3 | **9.70** | **9.70** | **0.00** | 0.01 | **30** | 43.91 | **9.70** | **0.00** | <0.01 | **<0.01** | **30/30** | 71.78 |
| | 4 | **11.57** | **11.57** | **0.00** | 0.02 | **30** | 42.73 | **11.57** | **0.00** | <0.01 | **<0.01** | **30/30** | 79.44 |
| | 5 | **12.93** | **12.93** | **0.00** | 0.04 | **30** | 28.84 | **12.93** | **0.00** | <0.01 | **<0.01** | **30/30** | 79.97 |
| | 6 | **14.00** | **14.00** | **0.00** | 0.12 | **30** | 23.82 | **14.00** | **0.00** | <0.01 | **<0.01** | **30/30** | 83.96 |
| | 7 | **14.93** | **14.93** | **0.00** | 0.29 | **30** | 21.32 | **14.93** | **0.00** | 0.01 | **0.01** | **30/30** | 84.81 |
| | 8 | **14.80** | **14.80** | **0.00** | 0.46 | **30** | 19.26 | **14.80** | **0.00** | 0.01 | **0.01** | **30/30** | 86.09 |
| 32 | 3 | **16.13** | **16.13** | **0.00** | 0.02 | **30** | 57.94 | **16.13** | **0.00** | <0.01 | **<0.01** | **30/30** | 78.26 |
| | 4 | **19.00** | **19.00** | **0.00** | 0.05 | **30** | 48.26 | **19.00** | **0.00** | <0.01 | **<0.01** | **30/30** | 81.46 |
| | 5 | **21.63** | **21.63** | **0.00** | 0.52 | **30** | 33.72 | **21.63** | **0.00** | 0.04 | **0.04** | **30/30** | 85.76 |
| | 6 | **23.73** | **23.73** | **0.00** | 1.39 | **30** | 25.52 | **23.73** | **0.00** | 0.10 | **0.10** | **30/30** | 85.91 |
| | 7 | **25.57** | **25.57** | **0.00** | 3.65 | **30** | 21.18 | **25.57** | **0.00** | 0.61 | **0.61** | **30/30** | 89.21 |
| | 8 | **27.50** | **27.50** | **0.00** | 5.19 | **30** | 18.22 | **27.50** | **0.00** | 0.99 | **0.99** | **30/30** | 88.80 |
| 64 | 3 | **25.43** | **25.43** | **0.00** | 0.04 | **30** | 65.65 | **25.43** | **0.00** | <0.01 | **<0.01** | **30/30** | 82.34 |
| | 4 | **30.37** | **30.37** | **0.00** | 0.22 | **30** | 57.79 | **30.37** | **0.00** | 0.04 | **0.04** | **30/30** | 86.45 |
| | 5 | **34.93** | **34.93** | **0.00** | 3.36 | **30** | 44.23 | **34.93** | **0.00** | 0.74 | **0.74** | **30/30** | 86.26 |
| | 6 | **39.13** | **39.13** | **0.00** | 12.93 | **30** | 37.10 | **39.13** | **0.00** | 2.03 | **2.03** | **30/30** | 90.35 |
| | 7 | **43.63** | **43.63** | **0.00** | 28.43 | **30** | 29.84 | **43.63** | **0.00** | 7.51 | **7.92** | **30/29** | 89.54 |
| | 8 | **45.53** | **45.53** | **0.00** | 84.45 | **30** | 24.83 | **45.53** | **0.00** | 14.14 | **27.62** | **30/25** | 84.73 |
| 128 | 3 | **36.77** | **36.77** | **0.00** | 0.25 | **30** | 70.96 | **36.77** | **0.00** | 0.02 | **0.02** | **30/30** | 85.18 |
| | 4 | **45.03** | **45.03** | **0.00** | 3.20 | **30** | 60.94 | **45.03** | **0.00** | 0.37 | **0.37** | **30/30** | 87.78 |
| | 5 | **53.43** | **53.43** | **0.00** | 13.48 | **30** | 54.18 | **53.43** | **0.00** | 3.30 | **3.30** | **30/30** | 90.42 |
| | 6 | **61.53** | **61.53** | **0.00** | 47.21 | **30** | 48.04 | **61.53** | **0.00** | 8.17 | **8.17** | **30/30** | 92.79 |
| | 7 | **68.47** | **68.47** | **0.00** | 337.66 | **30** | 39.01 | **68.47** | **0.00** | 29.22 | **36.33** | **30/28** | 91.40 |
| | 8 | **74.60** | 74.43 | 1.43 | 1941.38 | 20 | 33.90 | **74.60** | **0.11** | 74.85 | **1010.01** | **28/12** | 68.99 |
| 256 | 3 | **55.03** | **55.03** | **0.00** | 0.66 | **30** | 77.45 | **55.03** | **0.00** | 0.07 | **0.07** | **30/30** | 89.31 |
| | 4 | **68.93** | **68.93** | **0.00** | 4.99 | **30** | 74.05 | **68.93** | **0.00** | 1.82 | **1.82** | **30/30** | 92.27 |
| | 5 | **81.43** | **81.43** | **0.00** | 41.75 | **30** | 63.63 | **81.43** | **0.00** | 12.95 | **12.95** | **30/30** | 93.83 |
| | 6 | **93.60** | **93.60** | **0.00** | 406.78 | **30** | 56.99 | **93.60** | **0.00** | 45.63 | **48.11** | **30/28** | 93.00 |
| | 7 | **104.50** | 104.40 | 0.80 | 1764.49 | 24 | 52.06 | **104.50** | **0.00** | 129.27 | **369.43** | **30/20** | 87.91 |
| | 8 | **115.07** | 110.77 | 8.91 | 3562.90 | 1 | 43.12 | 115.03 | **2.70** | 375.68 | **3167.78** | **10/2** | 62.22 |
| 512 | 3 | **81.63** | **81.63** | **0.00** | 0.83 | **30** | 86.31 | **81.63** | **0.00** | 0.58 | **0.58** | **30/30** | 92.84 |
| | 4 | **101.13** | **101.13** | **0.00** | 10.56 | **30** | 80.23 | **101.13** | **0.00** | 9.01 | **9.01** | **30/30** | 93.71 |
| | 5 | **121.03** | **121.03** | **0.00** | 162.42 | **30** | 72.44 | **121.03** | **0.00** | 37.06 | **37.06** | **30/30** | 95.16 |
| | 6 | 138.40 | 137.13 | 1.81 | 2040.66 | 21 | 62.80 | **138.60** | **0.00** | 143.51 | **148.66** | **30/27** | 94.86 |
| | 7 | 155.17 | 126.53 | 23.97 | 3570.00 | 1 | 55.24 | **156.00** | **0.70** | 679.41 | **2115.36** | **20/10** | 82.00 |
| | 8 | 173.07 | 19.67 | 90.03 | **3633.19** | 0 | 47.95 | **174.23** | **3.25** | 1221.36 | 4499.14 | **3/1** | 64.24 |

## 8 Conclusions

In this work we approached the RFLCS problem by transforming an instance into a maximum independent set (MIS) problem instance as this is done by Blum et al. [6]. The MIS problem is subsequently solved by the ILP solver CPLEX. Our major contribution is to heavily reduce the conflict graph of the MIS problem by means of relaxed MDDs. This has multiple advantages: (1) reducing the conflict graph leads to an improved performance of CPLEX s.t. more instances could be solved faster to proven optimality, (2) the compiled relaxed MDDs present a discrete relaxation of the RFLCS problem meaning that upper bounds can be additionally derived and (3) it is also possible to quickly derive heuristic solutions from the MDDs. In many cases it was not necessary anymore to solve the ILP for the MIS problem since the upper bound from the MDD corresponded to the length of the derived heuristic solution and thus optimality was already proven. Overall, for many benchmark instances new state-of-the-art results could be obtained.

In the literature there are works where relaxed decision diagrams are successfully embedded into a branch-and-bound algorithm s.t. branching is done over nodes in the relaxed decision diagram. Since relaxed MDDs provide also strong upper bonds for the RFLCS problem it may be promising future work to develop such a branch-and-bound algorithm for the RFLCS problem to solve even larger instances to optimality. Moreover, it seems promising to apply relaxed decision diagrams also on other LCS-related problems.

## References

1. Adi, S.S., et al.: Repetition-free longest common subsequence. Discret. Appl. Math. **158**(12), 1315–1324 (2010)
2. Aho, A., Hopcroft, J., Ullman, J.: Data Structures and Algorithms. Addison-Wesley, Boston (1983)
3. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.N.: Decision diagrams for optimization. In: OŚullivan, B., Wooldridge, M. (eds.) Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42849-9
4. Blum, C., Blesa, M.J.: Construct, merge, solve and adapt: application to the repetition-free longest common subsequence problem. In: Chicano, F., Hu, B., García-Sánchez, P. (eds.) EvoCOP 2016. LNCS, vol. 9595, pp. 46–57. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30698-8_4
5. Blum, C., Blesa, M.J.: A comprehensive comparison of metaheuristics for the repetition-free longest common subsequence problem. J. Heuristics **24**(3), 551–579 (2017). https://doi.org/10.1007/s10732-017-9329-x
6. Blum, C., et al.: Solving longest common subsequence problems via a transformation to the maximum clique problem. Technical report. AC-TR-20-003 (2020). Submitted to Computers & Operations Research
7. Bonizzoni, P., Vedova, G.D., Dondi, R., Pirola, Y.: Variants of constrained longest common subsequence. Inf. Process. Lett. **110**(20), 877–881 (2010)

8. Brisk, P., Kaplan, A., Sarrafzadeh, M.: Area-efficient instruction set synthesis for reconfigurable system-on-chip designs. In: Proceedings of DAC 2004 - The 41st Annual Design Automation Conference. pp. 395–400. IEEE Press (2004)
9. Castelli, M., Beretta, S., Vanneschi, L.: A hybrid genetic algorithm for the repetition free longest common subsequence problem. Oper. Res. Lett. **41**(6), 644–649 (2013)
10. Chen, Y.C., Chao, K.M.: On the generalized constrained longest common subsequence problems. J. Comb. Optim. **21**(3), 383–392 (2011)
11. Chowdhury, S.R., Hasan, M.M., Iqbal, S., Rahman, M.S.: Computing a longest common palindromic subsequence. Fundamenta Informaticae **129**(4), 329–340 (2014)
12. Cire, A.A., Hoeve, W.V.: Multivalued decision diagrams for sequencing problems. Oper. Res. **61**(6), 1411–1428 (2013)
13. Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge (1997)
14. 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
15. Jiang, T., Lin, G.-H., Ma, B., Zhang, K.: The longest common subsequence problem for arc-annotated sequences. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. LNCS, vol. 1848, pp. 154–165. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45123-4_15
16. Jiang, T., Lin, G., Ma, B., Zhang, K.: A general edit distance between RNA structures. J. Comput. Biol. **9**(2), 371–388 (2002)
17. Kruskal, J.B.: An overview of sequence comparison: time warps, string edits, and macromolecules. SIAM Rev. **25**(2), 201–237 (1983)
18. Smith, T., Waterman, M.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1981)
19. Storer, J.A.: Data Compression: Methods and Theory. Computer Science Press, Inc., Rockville (1987)
20. Tsai, Y.T.: The constrained longest common subsequence problem. Inf. Process. Lett. **88**(4), 173–176 (2003)