Computers and Operations Research 125 (2021) 105089

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/caor

Solving longest common subsequence problems via a transformation to the maximum clique problem

Christian Blum [a,*], Marko Djukanovic [b], Alberto Santini [c], Hua Jiang [d], Chu-Min Li [e,f], Felip Manyà [a], Günter R. Raidl [b]

[a] Artificial Intelligence Research Institute (IIIA-CSIC), Campus UAB, Bellaterra, Spain
[b] Institute of Logic and Computation, TU Wien, Vienna, Austria
[c] Universitat Pompeu Fabra, Barcelona, Spain
[d] Yunnan University, Kunming, China
[e] University of Picardie Jules Verne, Amiens, France
[f] Huazhong University of Science and Technology, Wuhan, China

ARTICLE INFO

Article history:
Received 28 January 2020
Revised 14 August 2020
Accepted 19 August 2020
Available online 10 September 2020

Keywords:
Longest common subsequence
Transformation to maximum clique
Exact and heuristic solvers

ABSTRACT

Longest common subsequence problems find various applications in bioinformatics, data compression and text editing, just to name a few. Even though numerous heuristic approaches were published in the related literature for many of the considered problem variants during the last decades, solving these problems to optimality remains an important challenge. This is particularly the case when the number and the length of the input strings grows. In this work we define a new way to transform instances of the classical longest common subsequence problem and of some of its variants into instances of the maximum clique problem. Moreover, we propose a technique to reduce the size of the resulting graphs. Finally, a comprehensive experimental evaluation using recent exact and heuristic maximum clique solvers is presented. Numerous, so-far unsolved problem instances from benchmark sets taken from the literature were solved to optimality in this way.

1. Introduction

One of the common measures when comparing two (or more) strings is the length of their longest common subsequence (Iliopoulos and Sohel Rahman, 2009; Castelli et al., 2013). A subsequence is a string obtained by possibly deleting characters from another string. For example, AGT is a subsequence of ADDAGTA obtained by deleting the two occurrences of letter D and the last two occurrences of letter A.

The classical *longest common subsequence* (LCS) problem asks to find the longest subsequence common to a given set of strings. The LCS problem is one of the central problems in bioinformatics, often with strings representing segments of RNA or DNA (Gusfield, 1997; Smith and Waterman, 1981; Jiang et al., 2002). Other applications arise in computer science, in the fields of data compression, text editing (Kruskal, 1983), the production of circuits in field programmable gate arrays (Brisk et al., 2004) and file comparison (Storer, 1988; Aho et al., 1983).

The LCS problem is $\mathcal{NP}$-hard for an arbitrary number of input strings (Maier, 1978). If the number of strings is a constant, the problem is polynomially solvable by dynamic programming (Gusfield, 1997). Standard dynamic programming approaches for this problem require a time of $O(n^m)$ where n is the length of the longest input string and m is the number of strings. This complexity requirement makes the LCS problem hard to solve in practice with exact methods.

Real-life applications also require the solution of variants of the LCS problem in which additional constraints are imposed on the solutions. Examples concern the repetition-free longest common subsequence (RFLCS) problem (Adi et al., 2010), the constrained longest common subsequence (C-LCS) problem (Tsai, 2003), and the generalized constrained longest common subsequence (GC-LCS) problem (Chen and Chao, 2011). Others are mentioned in survey papers such as (Bonizzoni et al., 2010). Henceforth, we refer to the variants of the classical LCS problem, in general, as LCS-type problems.

Despite LCS-type problems being present in the literature for almost forty years, their computational difficulty causes that

* Corresponding author.
E-mail addresses: christian.blum@iiia.csic.es (C. Blum), djukanovic@ac.tuwien.ac.at (M. Djukanovic), alberto.santini@upf.edu (A. Santini), huajiang@ynu.edu.cn (H. Jiang), chu-min.li@u-picardie.fr (C.-M. Li), felip@iiia.csic.es (F. Manyà), raidl@ac.tuwien.ac.at (G.R. Raidl).

https://doi.org/10.1016/j.cor.2020.105089
0305-0548/

research is still active on this topic. In particular, in this work we present a new approach to solve various LCS-type problems by transforming them into instances of the *maximum clique* (MC) problem (Bomze et al., 1999). The central idea of the transformation is to construct, for each instance, a *conflict graph* (Lee et al., 2006). Hereby, an independent set in the conflict graph corresponds to a common subsequence concerning the original LCS instance. Moreover, a maximum independent set in the conflict graphs corresponds to a longest common subsequence of the LCS instance. Note that finding a *maximum independent set* (MIS) in the conflict graph is equivalent to finding a largest clique on the complement graph of the conflict graph. Therefore, an LCS problem instance can be solved by finding a largest clique in the complement of the conflict graph.

想法

The advantages of this approach are twofold. First, because of a steady improvement of the solvers for the MC problem, we have high-performing algorithms at our disposal that may make solving an MC problem on the complement of the conflict graph faster than solving the original LCS problem with known exact algorithms. Second, we will show that our transformation—in addition to the classical LCS problem—can be used to tackle other LCS-type problems from the literature, thus providing a unified approach for different LCS-type problems.

In the rest of this section we provide a short review of recent approaches for the classical LCS problem, with a focus on exact methods. In Section 2 we provide a description of the LCS-type problems considered in this work, detailing the transformations required to build the conflict graphs and providing further literature references for solution methods tailored to each specific variant. We provide computational evidence of the validity of our approach in Section 4. To this end, we compare the following three techniques with specialized algorithms for each of the considered LCS-type problems: (1) the Integer Linear Programming (ILP) solver CPLEX applied to solve the MIS problem in the conflict graph; (2) LMC (Jiang et al., 2016; Li et al., 2017), nowadays one of the best available exact MC solvers; and (3) LSCC-BMS (Wang et al., 2016), nowadays one of the best available heuristic MC solvers.

1.1. LCS-focused literature review

During the last decade, new and efficient heuristic approaches were proposed in the literature for tackling LCS-type problems (Blum and Festa, 2016), but it still remains an important challenge to solve these problems to optimality. The dynamic programming approach of Gusfield (1997), which was mentioned above, becomes impractical when the number m of input strings grows. At the same time, real-life applications of LCS-type problems also involve long strings with large values of n, making a run time of $O(n^m)$ impractical. Another approach is to model LCS-type problems—if possible—in terms of Integer Linear Programs (ILPs), which can then be solved by general-purpose ILP solvers, such as CPLEX or GUROBI. Computational experiments by Lee et al. (2009) in the context of two ILP models for the classical LCS problem showed that this approach turns impractical already for small values of m. Other specialized exact algorithms for the classical LCS problem are the following ones. Chan et al. (2016) proposed FAST_LCS, which is a parallel search algorithm. Pruning operations are utilized to reduce the computational effort. While the algorithm is effective for a small number of input strings, it also struggles for larger m. Wang et al. (2011) also proposed parallel algorithm labeled QUICK-DP. This algorithm is based on the dominant point approach and employs a fast divide-and-conquer technique to compute the so-called dominant points. More recently, Li et al. (2016) introduced the *Top_MLCS* algorithm, which is based on a directed acyclic layered-graph model (called irredundant common subsequence graph) and parallel topological sorting strategies used to filter out paths representing suboptimal solutions. Moreover, the authors showed that the earlier dominant-point-based algorithms do not scale well to larger LCS instances. Both a sequential and a parallel version of *Top_MLCS* was proposed. A space efficient algorithm based on a graph model, called the LEVELED-DAG, was described by Peng and Wang (2017). In the experimental comparison, LEVELED-DAG and TOP_MLCS solved the same number of benchmark instances to proven optimality, but LEVELED-DAG consumed less memory. Despite these recent advances, solving practically relevant instances to proven optimality remains a substantial challenge in terms of memory and computation time, even when utilizing many parallel threads.

研究趨勢

A recent branch of work on exact techniques for LCS-type problems concerns the development of extensions of the classical A* algorithm (Hart et al., 1968). One advantage of A* is that it can be hybridized with heuristic algorithms (Wang et al., 2010; Djukanovic et al., 2019). Djukanovic et al. (2019) presented a related study about A*-based anytime algorithms for the classical LCS problem.

Another branch of work concerns approximation algorithms. Jiang and Li (1995) suggested a simple Long Run (LR) algorithm that finds an LCS consisting of a single letter, with a $|\Sigma|$-approximation ratio. Bonizzoni et al. (2001) developed the so-called Expansion Algorithm (EA), which is also a $|\Sigma|$-approximation. The EA generally outperforms the LR algorithm. Tsai et al. (2002) introduced an improvement of the EA algorithm. Finally, two additional Σ-approximation algorithms—Enhanced Long Run (ELR) and Best-Next for Maximal Available Symbols (BNMAS)—were proposed in Huang et al. (2004).

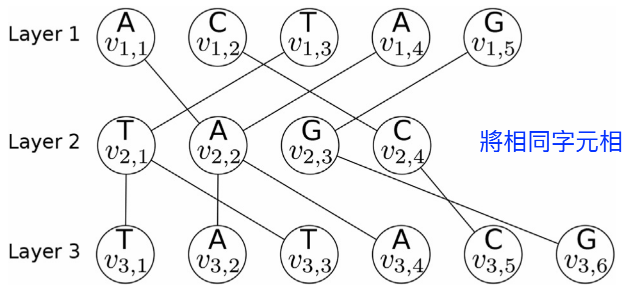
2. Considered problems and transformations

We start by defining a way to transform an instance of the classical LCS problem into a conflict graph in which a maximum independent set corresponds to a longest common subsequence of the original problem instance. Henceforth, an LCS problem instance is described by a pair (S, Σ) in which $S = \{s_1, \ldots, s_m\}$ is a set of input strings over a finite alphabet Σ. We denote the length of string $s_i \in S$ as $|s_i|$ and the element at position j in string s_i as $s_i[j], i = 1, \ldots, m$. Given such an instance, we construct an undirected multi-layered graph $G = (V, E)$ whose vertex set V is partitioned into sets $\{V_1, \ldots, V_m\}$. Each V_i is called a *layer* and consists of $|s_i|$ vertices. Note that each layer represents exactly one input string and each vertex of the layer represents a position in the string. More specifically, $V_i = \{\ _{i,1}, \ldots, \ _{i,|s_i|}\}$, where vertex $\mathbf{V}_{i,j}$ represents the j-th position of input string s_i.

We also partition the edge set E of the multi-layered graph G into sets $\{E_1, \ldots, E_{m-1}\}$, where E_i is the set of edges between layers V_i and V_{i+1}. Set E_i contains an edge connecting vertices $\mathbf{V}_{i,j}$ and $\mathbf{V}_{i+1,k}$ if and only if $s_i[j] = s_{i+1}[k]$, i.e., if the letter at position j of input string s_i is equal to the letter at position k of input string s_{i+1}. Figure 1 shows an example of this graph construction for three strings over an alphabet of size four.

Any sequence $p = (\ _{1,j_1},\ _{2,j_2}, \ldots,\ _{m,j_m})$ of m vertices with the i-th vertex of p being from the i-th layer of G is called a *complete path* in G if and only if it fulfills the condition that the letters at the positions of the input strings corresponding to the m vertices are all the same, that is, $s_1[j_1] = s_2[j_2] = \ldots = s_n[j_m]$. Note that if p fulfills this condition, there is—by definition—an edge between each pair of consecutive vertices of p. Given a complete path $p = (\ _{1,j_1},\ _{2,j_2}, \ldots,\ _{m,j_m})$, the common letter at positions $j_1, \ldots, j_m$ of the m input strings is also called the *letter of* p. We denote it by $\ell(p)$.

每一層代表一條輸入字串 v(i, j): i(row), j(column)



將相同字元相連

Fig. 1. The undirected multi-layered graph G obtained from the LCS instance ($S = \{s_1 = \text{ACTAG}, s_2 = \text{TAGC}, s_3 = \text{ATACG}\}, \Sigma = \{\text{A, C, T, G}\}$).

Two complete paths p and q, with $p = (\ _{1,j_1},\ _{2,j_2},\dots,\ _{m,j_m})$ and $q = (\ _{1,k_1},\ _{2,k_2},\dots,\ _{m,k_m})$, are said to *cross* if and only if there is at least one index $l \in \{1,\dots,m\}$ such that $j_l \leqslant k_l$ and at least one index $r \in \{1,\dots m\}, r \neq l$, such that $j_r \geqslant k_r$. To make the concept of crossing paths clearer, refer to Figure 2 which shows two examples based on the instance depicted in Figure 1. In the left figure, the solid and dashed paths are crossing because they contain crossing edges between layers 1 and 2. In the right figure, they cross because they contain a common vertex in layer 2.

重要 LCS -> MIS

Given these notations, the classical LCS problem can be transformed into the maximum independent set (MIS) problem as follows. First, note that solving the classical LCS problem amounts to finding the largest set of non-crossing complete paths in the respective multi-layered graph G. Based on G we can create the conflict graph $G^c = (V^c, E^c)$ with a vertex for each complete path of G and an edge between two paths iff they cross. Then, solving the LCS problem is equivalent to solving the MIS problem in G^c which, in turn, is equivalent to solving the MC problem in the complement $\overline{G^c}$ of graph G^c.

In the rest of this section we consider three LCS-type problems and show how analogous transformations allow us to reduce each problem to a MC problem on the complement of a conflict graph.
maximum clique

2.1. *Repetition-free longest common subsequence problem* RFLCS 無重複最長共同子序列問題

The repetition-free longest common subsequence (RFLCS) problem (Adi et al., 2010) is an LCS variant in which valid solutions are further constrained to contain each possible letter at most once. It was introduced as a comparison measure for sequences of different biological origin. In the related literature, this problem is generally considered for the case $m = 2$, that is, for two input strings. Note that even for $m = 2$ the problem is APX-hard (which implies it is $\mathcal{NP}$-hard), as shown by Adi et al. (2010). It is still an open question whether the RFLCS admits a constant factor approximation. A fixed parameter tractable (FTP) algorithm was presented in Blin et al. (2012).

Blum and Blesa (2018a) proposed the current best specialized algorithm for this problem: a construct, merge, solve and adapt (CMSA) approach in which the authors initialize the reduced sub-instance by beam search. In Blum and Blesa (2018a), the authors show how their algorithm outperforms other metaheuristics and the application of CPLEX to an ILP model of the problem.

To generate the conflict graph for the RFLCS problem, we first build the multi-layered graph G concerning the two input strings, just like in the case of the classical LCS problem. Note that, due to the two input strings, G will have two layers. Two complete paths p and q of G are in conflict if they fulfill at least one of the following two conditions:

1. p and q cross each other.
2. p and q have the same letter: $\ell(p) = \ell(q)$.

Note that the second condition ensures that no letter appears more than once in a solution.

2.2. *Longest arc-preserving common subsequence problem* LAPCS 最長保弧共同子序列問題

The second considered LCS variant is known as the longest arc-preserving common subsequence (LAPCS) problem (Evans, 1999a). As in the case of the RFLCS problem, the LAPCS problem is studied for two input strings/sequences in the literature. Note that, in the case of the LAPCS problem, the input strings are arc-annotated. An *arc annotation* of a string s is a pair of positions in s, say (i_1, i_2) with $i_1, i_2 \in \{1,\dots,|s|\}$ and $i_1 < i_2$. An arc-annotated sequence is a pair (s, P_s) where s is a string over some finite alphabet Σ and P_s is the set of arc annotations of s. Given two arc-annotated sequences (s_1, P_1) and (s_2, P_2), the two-layered multigraph G is constructed for s_1 and s_2 in the same way as shown before. Any set S of non-crossing complete paths in P is a feasible LAPCS solution if the following additional condition is fulfilled. For any pair $p = (\ _{1,j_1},\ _{2,j_2}) \neq q = (\ _{1,k_1},\ _{2,k_2})$ of non-crossing complete paths from S with $j_1 < k_1$ it must hold that if (j_1, k_1) is an arc annotation of s_1—that is, if $(j_1, k_1) \in P_1$—then (j_2, k_2) must be an arc annotation of s_2, and vice versa. The optimization goal is to find a largest feasible solution S.

Arc-annotated sequences are useful for the structural comparison of RNA sequences. Fig. 3 shows an example of an arc-annotated RNA sequence in which the arc annotations are indicated as solid lines linking the nucleobases ACGU. Evans (1999b, a) introduced the LACPS problem and showed that it is $\mathcal{NP}$-hard already for two strings. Researchers have also focused on special cases of the problem and developed polynomial time algorithms, approximation algorithms and fixed parameter tractability results for some of these cases (Evans, 1999c; Jiang et al., 2004; Gramm et al., 2006; Gorbenko and Popov, 2012).

Blum and Blesa (2018b) proposed the best specialized algorithms for the LAPCS. Depending on the problem instance charac-

LCS找最大非交叉的完全路徑，因為如果有交叉，代表相同元素的順序相反，但這已違反LCS

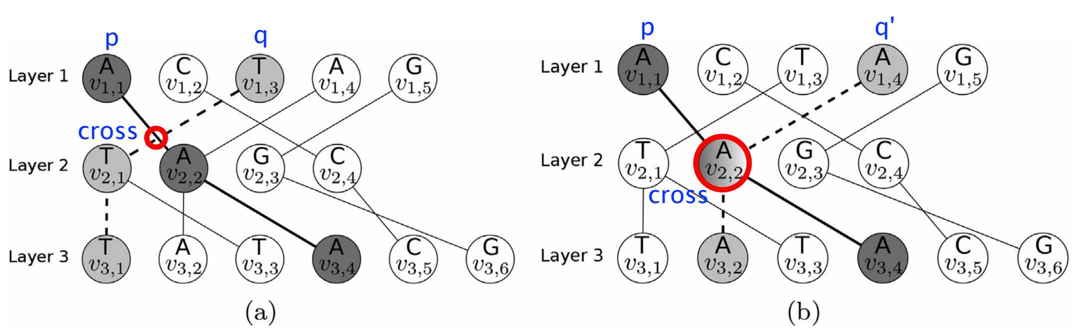


Fig. 2. Two examples of complete paths that cross, based on the LCS instance from Fig. 1. (a) Paths $p = (\ _{1,1},\ _{2,2},\ _{3,4})$ and $q = (\ _{1,3},\ _{2,1},\ _{3,1})$ cross because their corresponding edges between layers 1 and 2 cross. (b) Paths $p = (\ _{1,1},\ _{2,2},\ _{3,4})$ and $q = (\ _{1,4},\ _{2,2},\ _{3,2})$ cross because they both include vertex $V_{2,2}$ from the second layer.

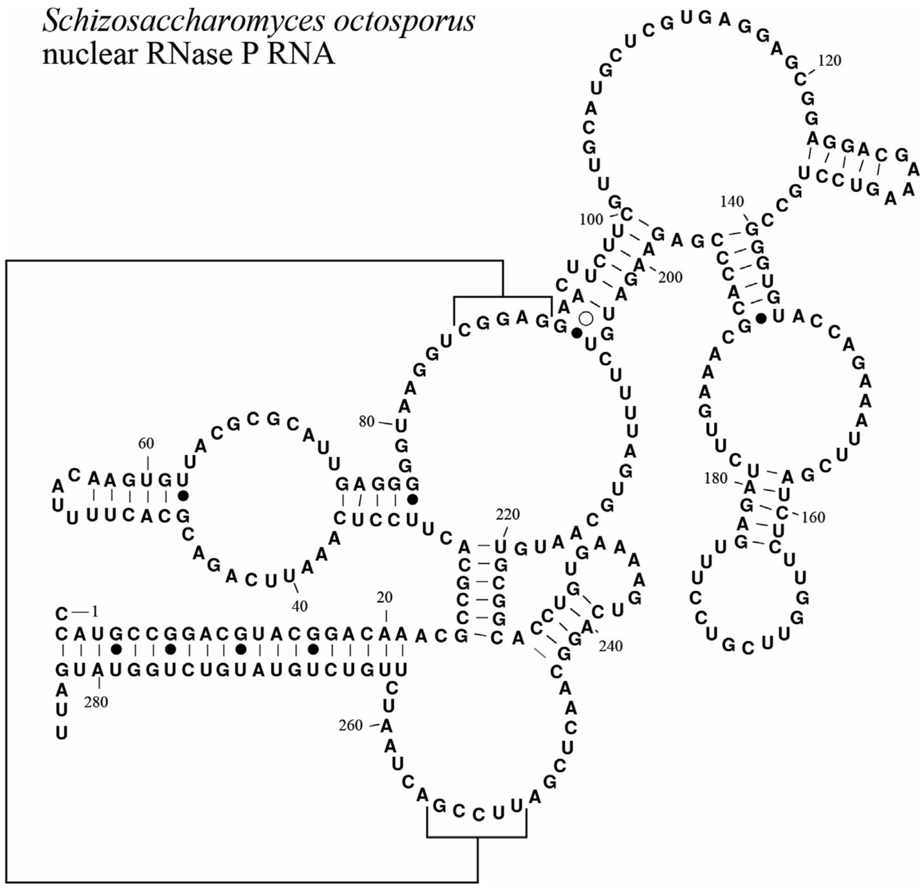


Fig. 3. Example of an arc-annotated sequence (RNA of *Schizosaccharomyces octosporus*). The connections between different positions of the RNA sequence, indicated by short lines, are the members of the arc annotation set. Note that this graphic was obtained from the RNase P Database (Brown, 1999).

teristics, the state-of-the-art algorithm is either a heuristic based on problem reduction, or an iterative probabilistic algorithm, both of which solve reduced ILP models. The authors compared these algorithms with the application of CPLEX to solve the MIS problem in the corresponding conflict graphs.

To generate the conflict graph for a LAPCS problem instance consisting of (s_1, P_1) and (s_2, P_2), we first construct the two-layered multi-graph G based on s_1 and s_2, as done in the classical LCS problem case. Two complete paths $p = (\ _{1,j_1},\ \ _{2,j_2}) \neq q = (\ _{1,k_1},\ \ _{2,k_2})$ with $j_1 < k_1$ are in conflict if and only if they fulfill at least one of the following two conditions:

1. p and q cross each other.
2. p and q violate the arc preservation constraints. This happens when either $(j_1, k_1) \in P_1$ and $(j_2, k_2) \notin P_2$, or $(j_2, k_2) \in P_2$ and $(j_1, k_1) \notin P_1$.

Fig. 4 shows an example LAPCS instance. The solution depicted with dashed lines is infeasible because it matches $\mathbf{V}_{1,2}$ and $\mathbf{V}_{1,4}$ in s_1 with, respectively, $\mathbf{V}_{2,4}$ and $\mathbf{V}_{2,5}$ in s_2. An arc annotation links the positions in s_1 but not in s_2, thus violating condition 2 above. The solution depicted with straight, solid lines, in contrast, is feasible.

3. Conflict graph reduction

The size of the conflict graphs (in terms of the number of vertices) mainly depends on the length and on the number of input strings. Let $n_{\max} := \max_{i=1,\ldots,m}\{|s_i|\}$. Then, the sizes of the conflict

n: length, m: number of the input，條數為conflict graph的維度

graphs can be expressed as follows: $\mathcal{O}(n_{\max}^m)$ in the case of the classical LCS problem, and $\mathcal{O}(n_{\max}^2)$ in the case of the RFLCS and LAPCS problems. In fact, during preliminary experiments we realized that the conflict graphs are too large, even for rather small problem instances from the literature, in the case of the classical LCS problem. Therefore, we henceforth focus exclusively on the RFLCS and LAPCS problems. However, even for these two problems, the conflict graphs are very large when large-scale problem instances are concerned. Therefore, we decided to investigate into techniques for reducing the size of the conflict graphs. Note that there are potentially two strategies for reducing the size of a given conflict graph G^c: (1) making use of problem-specific information relative to the respective LCS-type problems, and (2) analyzing and reducing G^c from the point of view of the MC problem. However, the latter strategy has proven ineffective in preliminary computational experiments. This is because solver LMC (the state-of-the-art exact MC problem solver that we used (Jiang et al., 2016; Li et al., 2017)) already implements powerful graph reduction procedures which were not able to reduce G^c. Therefore we make use of LCS-specific information to reduce the conflict graphs in a novel way.

The main idea for our reduction of the conflict graphs is based on making use of a high-quality primal (lower) bound value lb for the tackled problem, that is, the value of a high-quality solution. The value of the best-known solution from the literature can be taken for this purpose, for example. Before we proceed, the following notation is required: given a string t and two indices $l, r \in \{1, \ldots, |t|\}$ with $l \leqslant r$, $t[l, r]$ denotes the substring of t starting at position l and ending at position r. Now, on the basis of the primal bound lb, it can be decided for every complete path

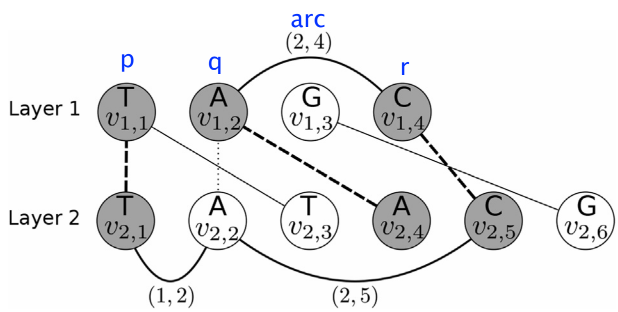


Fig. 4. This example shows the undirected multi-layered graph G obtained from the LAPCS instance consisting of $(s_1 = \mathrm{TAGC}, P_1 = \{(2,4)\})$ and $(s_2 = \mathrm{TATACG}, P_2 = \{(1,2),(2,5)\})$. The solution in dashed lines $\{p = (\ _{1,1},\ _{2,1}), q = (\ _{1,2},\ _{2,4}), r = (\ _{1,4},\ _{2,5})\}$ is not valid because arc $(2,4) \in P_1$ connects two chosen positions in s_1, while the corresponding chosen positions in s_2 – that is, positions 4 and 5 – are not connected by an arc from P_2. The solution in straight, solid lines is feasible.

$p = \ _{1,j_1}, \ldots, \ _{m,j_m}$ of the multi-layered graph, if the corresponding vertex $\mathbf{V}_p$ can be removed from the conflict graph G^c without loosing an optimal solution.[1] This is done as follows. First, note that the complete path under consideration splits each input string s_i into two parts: $s_i[1, j_i - 1]$ (the left-hand side) and $s_i[j_i + 1, |s_i|]$ (the right-hand side). Henceforth we denote the set of left-hand sides corresponding to a complete path p by S_p^{L}, and the set of right-hand sides by S_p^{R}. More formally: 將字串依path分成兩部分

左半部 $S_p^{\mathrm{L}} = \{s_i[1, j_i - 1] \mid i = 1, \ldots, m\}$

右半部 $S_p^{\mathrm{R}} = \{s_i[j_i + 1, |s_i|] \mid i = 1, \ldots, m\}$ j(i) = Vp 被移除了

Note that both S_p^{L} and S_p^{R} are subinstances of the original problem instance. Therefore, any upper bound function UB() known for the problem (RFLCS, respectively LAPCS) can be used for (over)-estimating the quality of the length of an optimal solution in S_p^{L} and S_p^{R}. Given such an upper bound function UB(), vertex $\mathbf{V}_p$ and all corresponding edges can be deleted from the conflict graph G^c iff

$$\mathrm{UB}\left(S_p^{\mathrm{L}}\right) + \overset{\text{Vp}}{1} + \mathrm{UB}\left(S_p^{\mathrm{R}}\right) < lb. \quad \text{左半上限+1+右半上限 < lb} \qquad (1)$$

For the following discussion, bear in mind that any upper bound for the classical LCS problem is also an upper bound for the RFLCS and LAPCS problems. This is, because these two problems correspond to classical LCS problems with additional constraints. In other words, the set of valid solutions of a RFLCS problem instance, respectively a LAPCS problem instance, is a subset of the set of valid solutions of the instance if solved as a classical LCS problem. Therefore, upper bound functions developed for the classical LCS problem are candidates to be used for UB() in Eq. 1.

upper bound

Blum et al. (2009), for example, introduced an upper bound function henceforth labeled $\mathrm{UB}_1^{\mathrm{LCS}}()$ for the classical LCS problem (which is a tightened version of a bound originally introduced by Fraser (1995)). Given a problem instance (S, Σ), for each input string $s_i \in S$ and each letter $a \in \Sigma$, let $|s_i|_a$ be the number of occurrences of a in s_i and let $c_a(S) = \min_{s_i \in S} |s_i|_a$. Then, $\mathrm{UB}_1^{\mathrm{LCS}}()$ is defined as follows: a在第1~n字串出現的最少次數

$$\mathrm{UB}_1^{\mathrm{LCS}}(S) = \sum_{a \in \Sigma} c_a(S) \quad \text{所有字母 在所有字串中出現的最少次數總和}$$

Let $\delta(a, S)$ for $a \in \Sigma$ evaluate to one, if letter a appears at least once in each input string from S, and otherwise to zero. As each letter from Σ can mostly appear once in a valid RFLCS solution, $\mathrm{UB}_1^{\mathrm{LCS}}()$

delta(a, S): a字母是否都出現過在每個輸入字串中

from above reduces to the following upper bound function in the context of the RFLCS problem:

$$\mathrm{UB}_1^{\mathrm{RFLCS}}(S) = \sum_{a \in \Sigma} \delta(a, S) \quad \text{所有字母 在所有字串中是否都有出現過的總和}$$

Finally, when used for our purposes—that is, for obtaining an upper bound for (sub-) instances S_p^{L} and S_p^{R} in Equation 1 in the context of an RFLCS instance—we can even exclude letter $l(p)$ (the letter of path p) from the sum. This results in:

$$\mathrm{UB}_1^{\mathrm{RFLCS}}(S, p) = \sum_{a \in \Sigma \setminus \{l(p)\}} \delta(a, S). \quad \text{path代表一個字母，排除了一些字母}$$

Wang et al. (2011) proposed another upper bound function for the classical LCS problem, henceforth labeled $\mathrm{UB}_2()$, which is based on dynamic programming (DP). This function is defined as follows:

$$\mathrm{UB}_2(S) = \min_{i=1,\ldots,m-1} \mathrm{LCS}(s_i, s_{i+1}), \quad \text{兩兩一組最小的LCS長度}$$

where $\mathrm{LCS}(s_i, s_{i+1})$ refers to the length of the longest common subsequence of input strings s_i and s_{i+1}. Using the DP recursion of Wang et al. (2010) we can obtain this bound in $\mathcal{O}(m)$ time by using an appropriate preprocessing data structure known as the *scoring matrix* (Wang et al., 2012; Inenaga and Hyyrö, 2018). In particular, note that in the context of the RFLCS and LAPCS problems, the preprocessing is done in $\mathcal{O}(n^2)$ time.

In summary, for the conflict graph reduction in the context of the RFLCS problem, UB() is defined as $\min\left\{\mathrm{UB}_1^{\mathrm{RFLCS}}(), \mathrm{UB}_2()\right\}$; and in the context of the LAPCS problem $\mathrm{UB}() := \mathrm{UB}_2()$, because $\mathrm{UB}_2() < \mathrm{UB}_1^{\mathrm{LCS}}()$ in all cases.

4. Experimental evaluation

The aim of the computational experiments is to compare two strategies to solve LCS problems: (1) their direct solution using a specialized state-of-the-art algorithm, and (2) their transformation to the MIS, respectively the MC, problems and the subsequent solution by CPLEX[2] (in case of the MIS problem) or by the following MC solvers:

- LMC. This exact MC solver was introduced by Jiang et al. (2016, 2017). It is currently one of the best exact solvers available for the MC problem. It combines an aggressive preprocessing of the graph with a MaxSAT solver (Li et al., 2009) in a branch-and-bound scheme.
- LSCC-BMS. This is one of the best-performing heuristic algorithms for the MC problem. Wang et al. (2016) introduced this local-search-based algorithm, whose main strengths are a configuration checking procedure that reduces the probability of cycling during local search, and a low-complexity vertex swap neighborhood which is fast even on massive graphs[3].

Note that both CPLEX and LSCC-BMS were executed on a cluster of 12-core Intel Xeon 5670 CPUs at 2.9 GHz and at least 40 GB of RAM. LMC was executed on a cluster with 8-core Intel Xeon E5-2680 CPUs at 2.4 GHz and with 128 GB of memory. In both cases, the memory consumption of each process was limited to 16 GB.

RFLCS benchmark instances. Two sets of problem instances can be found in the related literature. The first set, henceforth denoted

[1] Note that the conflict graph reduction will be described for a general case of n input strings, even though we only have two input strings in the cases of the RFLCS and LAPCS problems.

[2] IBM ILOG CPLEX is an optimization software package that includes state-of-the-art exact techniques for solving integer linear programming models, among others. It is available for free for academic purposes. For more information, we refer the interested reader to http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html. In this work we made use of version 12.7.

[3] We downloaded the code of LSCC-BMS from http://ai.nenu.edu.cn/wangyy/Yiyuan-data/LocalSearchforMWCP.htm on April 29, 2019.

Table 1
Characteristics of real instances from set LAPCS-REAL. All 20 arc-annotated RNA sequences were taken from the RNase P Database (Brown, 1999).

Instance	First String			Second string		
	RNA	n	n_{arcs}	RNA	n	n_{arcs}
Real_1	*Allochromatium vinosum*	369	119	*Haemophilus influenza*	377	124
Real_2	*Bacteroides thetaiotaomicron*	361	121	*Porphyromonas gingivalis*	398	131
Real_3	*Halococcus morrhuae*	475	154	*Haloferax volcanii*	433	142
Real_4	*Klebsiella pneumoniae*	383	127	*Escherichia coli*	377	124
Real_5	*Methanococcus jannaschii*	252	75	*Archaeoglobus fulgidus*	229	67
Real_6	*Methanosarcina barkeri*	371	115	*Pyrococcus abyssi*	330	100
Real_7	*Mycoplasma genitalium*	384	119	*Mycoplasma pneumoniae*	369	112
Real_8	*Saccharomyces kluveri*	336	90	*Schizosaccharomyces octosporus*	281	71
Real_9	*Serratia marcescens*	378	125	*Shewanella putrefaciens*	354	115
Real_10	*Streptomyces bikiniensis*	398	135	*Streptomyces lividans*	405	138

by RFLCS-SET1, consists of 30 randomly generated problem instances for each combination of the input sequence length $n \in \{32, 64, 128, 256, 512, 1024, 2048, 4096\}$ and the alphabet size $|\Sigma| \in \{\frac{n}{8}, \frac{n}{4}, \frac{3n}{8}, \frac{n}{2}, \frac{5n}{8}, \frac{3n}{4}, \frac{7n}{8}\}$. This results in a total of 1680 instances. The second set, henceforth denoted by RFLCS-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 letter, $\text{reps} \in \{3, 4, 5, 6, 7, 8\}$. In total, set RFLCS-SET2 contains 1440 instances.

LAPCS benchmark instances. The recent literature on the LAPCS problem considers both artificial instances (benchmark set LAPCS-ARTI) and real RNA instances (benchmark set LAPCS-REAL). Each artificial instance consists of two randomly generated RNA strings of length $n \in \{100, 200, \ldots, 900, 1000\}$. Moreover, each input string has $n_{arcs} \in \{\frac{n}{10}, \frac{n}{5}, \frac{n}{2}\}$ randomly generated unique arc annotations. Set LAPCS-ARTI consists of 30 instances for each combination of n and n_{arcs}, which makes a total of 900 problem instances. Set LAPCS-REAL consists of 10 problem instances that are composed of arc-annotated RNA sequences downloaded from the RNase P Database (Brown, 1999). Note that the alphabet size in all cases is equal to four. Table 1 summaries the characteristics of these instances.

Due to the fact that the amount of reduction of the conflict graphs from Section 3 depends on the quality of the used primal bound per instance, we used the currently best-known solution values from the literature for all considered instances. In the case of the RFLCS problem, these values were taken from Blum and Blesa (2018c), and in the case of the LAPCS problem from Blum and Blesa (2018b).

4.1. *Results without conflict graph reduction*

All three methods—CPLEX, LMC, and LSCC-BMS—were applied with a computation time limit of 3600 s (1 h) and a memory limit of 16 GB per run to all RFLCS and LAPCS problem instances. The results are presented in numerical form in Tables 2 and 3 concerning the RFLCS problem, and in Tables 4 and 5 concerning the LAPCS problem. The first two columns in Tables 2–4 indicate the problem instance characteristics, while the third column provides the currently best known results from the literature. In the case of the RFLCS problem, the best known results were obtained by the current state-of-the-art method—a hybrid CMSA algorithm—from Blum and Blesa (2018c). The best-known results for the LAPCS problem were obtained by two state-of-the-art algorithms—a hybrid evolutionary algorithm (HYB-EA) and an ILP-based heuristic—from Blum and Blesa (2018b). Note that, in this way, the results of our transformation-based approaches are compared to the current state-of-the-art methods. Each table row provides results averaged over 30 problem instances of the same type. Table 5 is slightly different. The first column provides the instance name, while the second column indicates the best-known results from the literature. Moreover, each table row only covers one single problem instance. In the case of the LAPCS problem, the best-known results from the literature are additionally marked either by an *a*, indicating that an ILP-based heuristic has produced this result, or by a *b*, which indicates that the HYB-EA algorithm has generated this result. In Tables 2–4, the results of CPLEX and LSM are each provided in four columns. The first one (with heading **result**) contains the average solution quality obtained for the 30 problem instances. The second column (with heading $\bar{t}$) indicates the average computation time at which the best solution of a run was found, while the third column (with heading $\bar{t}_{\mathbf{opt}}$) provides the average computation time at which optimality was proven. Finally, the fourth table column contains the number of instances that could be solved to optimality. This fourth table column is not provided in Table 5, as it only deals with one instance per table row. Furthermore, the results of LSCC-BMS are given in two columns in all cases, providing the (average) result and the (average) computation time. Note that a value in the columns with heading **result** is indicated in bold font if the value is at least as good as the best known one from the literature. Moreover, a value is marked by an asterisk in case it corresponds to a new best-known result. Finally, results of CPLEX and LMC are marked by a gray background if they correspond to provenly optimal results.

The following observations can be made in the case of the RFLCS problem:

- While both LMC and LSCC-BMS are able to provide feasible solutions for all problem instances from both sets (RFLCS-SET1 and RFLCS-SET2), CPLEX suffers from a sharp phase transition when the conflict graphs become too large. Observe, for example, the case ($|\Sigma| = n/8, n = 256$) in Table 2 in comparison to the next larger case ($|\Sigma| = n/8, n = 512$). While CPLEX is able to solve all instances of the first case to optimality, it only provides very short solutions in the second case.
- Concerning the comparison of the two exact solvers, we can state that LMC (the MC solver) clearly outperforms CPLEX. LMC is able to solve 1282 RFLCS-SET1 instances and 1237 RFLCS-SET2 instances to optimality, while CPLEX can only solve 1221 RFLCS-SET1 instance and 1181 RFLCS-SET2 instances to optimality. Moreover, LMC does not suffer from the above-mentioned phase transition for the remaining instances, and it requires generally less computation time. More specifically, while LSM requires—on average—41.7 s for proving optimality (if possible) of RFLCS-SET1 instances, CPLEX requires 187.2 s; respectively 34.07 and 127.14 s in the case of the RFLCS-SET2 instances.
- The heuristic MC solver LSCC-BMS is especially successful in those cases in which the exact techniques start to fail. See, for example, cases ($|\Sigma| = n/8, n \in \{512, 1024\}$) in Table 2 and cases ($|\Sigma| = 256, \text{reps} \in \{6, 7, 8\}$) in Table 3. LSCC-BMS can be seen as

Table 2
Experimental results for RFLCS instances RFLCS-SET1.

$\lvert\Sigma\rvert$	n	**Spec.**	CPLEX				LMC				LSCC+BMS	
		Tech.	**result**	$\bar{t}$	$\bar{t}_{\text{opt}}$	#opt	**result**	$\bar{t}$	$\bar{t}_{\text{opt}}$	#opt	**result**	$\bar{t}$
$n/8$	32	4.00	**4.00**	0.09	0.09	30	**4.00**	0.00	0.01	30	**4.0**	0.01
	64	8.00	**8.00**	0.81	0.81	30	**8.00**	0.00	0.07	30	**8.0**	0.00
	128	16.00	**16.00**	8.12	8.12	30	**16.00**	0.00	49.61	30	**16.0**	0.01
	256	31.97	**31.97**	188.31	188.31	30	31.90	20.54	– –	0	**31.97**	0.09
	512	63.27	5.17	625.34	– –	0	62.50	485.59	– –	0	**63.90***	68.84
	1024	111.57	0.03	1461.74	– –	0	112.53	818.57	– –	0	**116.10***	1297.00
	2048	182.67	– –	– –	– –	0	182.40	1331.53	– –	0	181.67	1394.27
	4096	283.33	– –	– –	– –	0	281.37	1037.61	– –	0	261.37	1510.89
$n/4$	32	7.83	**7.83**	0.03	0.03	30	**7.83**	0.00	0.00	30	**7.83**	0.00
	64	14.67	**14.67**	0.29	0.29	30	**14.67**	0.00	0.01	30	**14.67**	0.00
	128	25.77	**25.93***	2.02	2.50	30	**25.93***	0.01	0.09	30	**25.93***	0.02
	256	43.70	**43.97***	30.92	51.17	30	**43.97***	0.12	0.80	30	**43.97***	0.22
	512	67.90	68.50	582.53	1622.77	27	**68.57***	75.61	185.15	30	**68.57***	7.57
	1024	103.00	0.00	240.97	– –	0	103.77	386.81	– –	0	**104.87***	877.29
	2048	154.33	0.00	1398.78	– –	0	152.87	438.52	– –	0	151.33	1485.85
	4096	226.67	– –	– –	– –	0	223.57	780.50	– –	0	207.03	1984.69
$3n/8$	32	8.77	**8.77**	0.02	0.02	30	**8.77**	0.00	0.00	30	**8.77**	0.00
	64	15.53	**15.53**	0.10	0.10	30	**15.53**	0.00	0.00	30	**15.53**	0.00
	128	24.90	**24.90**	1.75	1.79	30	**24.90**	0.00	0.03	30	**24.90**	0.01
	256	39.97	**39.97**	5.25	5.90	30	**39.97**	0.02	0.20	30	**39.97**	0.13
	512	59.77	**59.97***	106.42	133.02	30	**59.97***	0.46	1.83	30	**59.97***	1.99
	1024	90.50	90.67	2204.06	2263.32	23	**90.73***	5.71	30.67	30	**90.73***	145.24
	2048	130.57	0.00	547.50	– –	0	129.67	233.36	105.92	1	129.13	1578.88
	4096	191.37	– –	– –	– –	0	188.30	311.61	– –	0	179.73	1670.85
$n/2$	32	8.87	**8.87**	0.01	0.01	30	**8.87**	0.00	0.00	30	**8.87**	0.00
	64	14.80	**14.80**	0.06	0.06	30	**14.80**	0.00	0.00	30	**14.80**	0.00
	128	22.93	**22.93**	0.76	0.78	30	**22.93**	0.00	0.01	30	**22.93**	0.00
	256	35.10	**35.20***	2.18	2.27	30	**35.20***	0.02	0.09	30	**35.20***	0.09
	512	53.10	**53.13***	31.82	34.03	30	**53.13***	0.08	0.66	30	**53.13***	0.71
	1024	79.03	**79.13***	627.90	701.13	30	**79.13***	6.04	11.56	30	**79.13***	30.80
	2048	115.30	0.00	248.56	– –	0	115.07	432.97	598.59	19	114.87	1517.02
	4096	167.47	0.00	1295.77	– –	0	165.87	390.18	– –	0	159.37	1490.48
$5n/8$	32	8.60	**8.60**	0.01	0.01	30	**8.60**	0.00	0.00	30	**8.60**	0.00
	64	13.30	**13.30**	0.03	0.03	30	**13.30**	0.00	0.00	30	**13.30**	0.00
	128	21.20	**21.20**	0.36	0.37	30	**21.20**	0.00	0.01	30	**21.20**	0.00
	256	32.53	**32.53**	4.21	4.36	30	**32.53**	0.01	0.05	30	**32.53**	0.04
	512	47.83	**47.83**	13.06	13.15	30	**47.83**	0.04	0.33	30	**47.83**	0.28
	1024	70.03	**70.20***	208.55	215.63	30	**70.20***	1.43	4.12	30	**70.20***	8.70
	2048	103.80	48.33	2306.93	3328.76	1	**103.97***	63.19	158.21	30	103.87	936.80
	4096	150.00	0.00	878.84	– –	0	148.53	302.72	1607.66	2	145.77	1423.49
$3n/4$	32	8.17	**8.17**	0.00	0.00	30	**8.17**	0.00	0.00	30	**8.17**	0.00
	64	12.53	**12.53**	0.02	0.02	30	**12.53**	0.00	0.00	30	**12.53**	0.00
	128	19.70	**19.70**	0.17	0.18	30	**19.70**	0.00	0.00	30	**19.70**	0.00
	256	29.97	**29.97**	2.25	2.32	30	**29.97**	0.00	0.03	30	**29.97**	0.02
	512	44.53	**44.57***	4.90	4.94	30	**44.57***	0.03	0.19	30	**44.57***	0.29
	1024	65.07	**65.20***	96.77	97.46	30	**65.20***	0.75	2.11	30	**65.20***	3.39
	2048	94.53	**94.67***	1829.86	1862.21	30	**94.67***	4.57	18.69	30	94.63	638.75
	4096	136.57	0.00	500.41	– –	0	135.73	355.77	682.50	13	133.53	1617.99
$7n/8$	32	7.67	**7.67**	0.00	0.00	30	**7.67**	0.00	0.00	30	**7.67**	0.00
	64	11.57	**11.57**	0.01	0.01	30	**11.57**	0.00	0.00	30	**11.57**	0.00
	128	18.40	**18.40**	0.12	0.12	30	**18.40**	0.00	0.00	30	**18.40**	0.00
	256	27.80	**27.80**	1.21	1.22	30	**27.80**	0.00	0.02	30	**27.80**	0.01
	512	40.57	**40.60***	2.93	3.01	30	**40.60***	0.02	0.12	30	**40.60***	0.10
	1024	60.50	**60.57***	79.74	79.76	30	**60.57***	0.28	1.19	30	**60.57***	3.55
	2048	88.00	**88.00**	831.15	896.78	30	**88.00**	4.13	18.68	30	**88.00**	114.45
	4096	127.20	0.00	361.39	– –	0	126.50	212.34	478.99	17	125.47	1608.56

the most successful one among the techniques, providing new best-known results in 35 cases (considering both instance sets together), while LMC provides new best-known results in 30 cases and CPLEX in 24 cases.

All in all we can state that the idea of solving the RFLCS problem by means of the transformation to the MC problem is very successful, even before trying to reduce the size of the conflict graphs.

Let us now turn towards the LAPCS problem. In some aspects, the observations that can be made in the context of the artificial instances (LAPCS-ARTI; Table 4) are similar to the ones made for the RFLCS problem. CPLEX suffers from a sharp phase transition. In fact, it is only able to provide solutions for the case of the smallest problem instances ($n = 100$). LMC does not suffer from this phase transition and is able to provide feasible solutions of reasonable quality until instances with input strings of length $n = 500$. Both LMC are

Table 3
Experimental results RFLCS instances RFLCS-SET2.

$\lvert\Sigma\rvert$	reps	Spec. Tech.	Cplex result	$\bar{t}$	$\bar{t}_{\text{opt}}$	#opt	Lmc result	$\bar{t}$	$\bar{t}_{\text{opt}}$	#opt	Lscc-Bmc result	$\bar{t}$
4	3	3.47	**3.47**	0.00	0.00	30	**3.47**	0.00	0.00	30	**3.47**	0.00
	4	3.77	**3.77**	0.00	0.00	30	**3.77**	0.00	0.00	30	**3.77**	0.00
	5	3.83	**3.83**	0.00	0.00	30	**3.83**	0.00	0.00	30	**3.83**	0.00
	6	3.90	**3.90**	0.00	0.00	30	**3.90**	0.00	0.00	30	**3.90**	0.00
	7	3.97	**3.97**	0.01	0.01	30	**3.97**	0.00	0.00	30	**3.97**	0.00
	8	3.97	**3.97**	0.01	0.01	30	**3.97**	0.00	0.00	30	**3.97**	0.00
8	3	6.23	**6.23**	0.00	0.00	30	**6.23**	0.00	0.00	30	**6.23**	0.00
	4	6.87	**6.87**	0.00	0.00	30	**6.87**	0.00	0.00	30	**6.87**	0.00
	5	7.40	**7.40**	0.02	0.02	30	**7.40**	0.00	0.00	30	**7.40**	0.00
	6	7.53	**7.53**	0.02	0.02	30	**7.53**	0.00	0.00	30	**7.53**	0.00
	7	7.70	**7.70**	0.06	0.06	30	**7.70**	0.00	0.00	30	**7.70**	0.00
	8	7.77	**7.77**	0.05	0.05	30	**7.77**	0.00	0.00	30	**7.77**	0.00
16	3	9.70	**9.70**	0.01	0.01	30	**9.70**	0.00	0.00	30	**9.70**	0.00
	4	11.57	**11.57**	0.03	0.03	30	**11.57**	0.00	0.00	30	**11.57**	0.00
	5	12.93	**12.93**	0.06	0.06	30	**12.93**	0.00	0.00	30	**12.93**	0.00
	6	14.00	**14.00**	0.15	0.16	30	**14.00**	0.00	0.01	30	**14.00**	0.00
	7	14.93	**14.93**	0.30	0.30	30	**14.93**	0.00	0.02	30	**14.93**	0.02
	8	14.80	**14.80**	0.37	0.38	30	**14.80**	0.00	0.02	30	**14.80**	0.00
32	3	16.13	**16.13**	0.08	0.08	30	**16.13**	0.00	0.00	30	**16.13**	0.00
	4	19.00	**19.00**	0.27	0.27	30	**19.00**	0.00	0.01	30	**19.00**	0.00
	5	21.63	**21.63**	0.83	0.85	30	**21.63**	0.00	0.02	30	**21.63**	0.01
	6	23.73	**23.73**	1.57	1.65	30	**23.73**	0.00	0.04	30	**23.73**	0.01
	7	25.53	**25.57***	2.23	2.34	30	**25.57***	0.02	0.10	30	**25.57***	0.03
	8	27.40	**27.50***	4.59	4.71	30	**27.50***	0.06	0.23	30	**27.50***	0.07
64	3	25.43	**25.43**	0.88	0.91	30	**25.43**	0.00	0.01	30	**25.43**	0.00
	4	30.37	**30.37**	2.65	2.80	30	**30.37**	0.01	0.05	30	**30.37**	0.02
	5	34.87	**34.93***	3.57	4.66	30	**34.93***	0.02	0.13	30	**34.93***	0.07
	6	39.07	**39.13***	13.36	17.37	30	**39.13***	0.05	0.34	30	**39.13***	0.18
	7	43.50	**43.63***	28.44	55.76	30	**43.63***	0.16	0.92	30	**43.63***	0.40
	8	45.17	**45.53***	58.39	116.58	30	**45.53***	1.38	5.41	30	**45.53***	0.75
128	3	36.70	**36.77***	2.39	2.44	30	**36.77***	0.01	0.09	30	**36.77***	0.14
	4	44.90	**45.03***	12.95	15.22	30	**45.03***	0.06	0.37	30	**45.03***	0.39
	5	53.23	**53.43***	48.50	64.03	30	**53.43***	0.15	1.08	30	**53.43***	1.12
	6	61.07	**61.53***	183.29	300.56	30	**61.53***	4.55	7.76	30	**61.53***	4.42
	7	67.90	68.40	749.39	1377.40	25	**68.47***	8.25	54.19	30	**68.47***	5.13
	8	73.57	74.37	1288.16	1932.99	11	74.30	524.20	474.27	13	**74.60***	22.68
256	3	54.97	**55.03***	46.81	48.61	30	**55.03***	0.08	0.69	30	**55.03***	1.06
	4	68.70	**68.93***	247.83	268.90	30	**68.93***	0.31	2.90	30	**68.93***	8.53
	5	81.00	**81.43***	917.97	1182.86	30	**81.43***	9.65	21.74	30	**81.43***	45.01
	6	93.10	73.83	2951.48	3090.98	2	93.17	239.22	418.66	17	**93.53***	162.94
	7	103.50	0.00	308.34	– –	0	103.13	132.52	499.03	3	**104.40***	734.99
	8	113.70	0.00	501.06	– –	0	113.10	298.94	– –	0	**114.70***	1300.54
512	3	81.57	**81.63***	524.51	536.33	30	**81.63***	0.72	5.38	30	**81.63***	41.71
	4	100.83	78.63	2899.04	3142.25	3	101.10	157.68	230.29	29	**101.13***	602.19
	5	120.43	0.00	404.86	– –	0	118.70	539.36	851.21	5	119.60	1147.39
	6	137.03	0.00	681.76	– –	0	135.50	483.24	– –	0	136.00	1894.44
	7	154.57	0.00	1218.70	– –	0	152.33	784.72	– –	0	150.63	1784.08
	8	172.10	– –	– –	– –	0	169.90	698.89	– –	0	166.47	1428.87

CPLEX are able to solve 80 problem instances to optimality. And finally, the heuristic MC solver LSCC-BMS is again very successful in those cases in which LMC and CPLEX start to fail proving optimality (see the instances with $n = 200$). Concerning the results obtained for the real instances (LAPCS-REAL; Table 5), we can state that LSM is, by far, the most successful algorithm. While CPLEX is not able to derive any feasible solutions and LSCC-BMS never matches the best results from the literature, LSM matches the best results from the literature in three cases and obtains new best-known solutions in two additional cases. Nevertheless, we can state that the results—obtained before trying to reduce the size of the conflict graphs—are rather unsatisfactory in the context of the LAPCS problem. The main reason for this is the increased size of the conflict graphs in comparison to the RFLCS problem, which is due to the small alphabet size of four.

4.2. Results after conflict graph reduction

After reducing all the conflict graphs with the method described in Section 3, we first measured the amount of reduction that was achieved. This reduction is displayed for all RFLCS and LAPCS problem instances by means of boxplots in Figs. 5–8. More specifically,

Table 4
Experimental results for LAPCS instances LAPCS-ARTI.

n	n_{arcs}	Spec. Tech.	Cplex result	Cplex $\bar{t}$	Cplex $\bar{t}_{\text{opt}}$	Cplex #opt	Lmc result	Lmc $\bar{t}$	Lmc $\bar{t}_{\text{opt}}$	Lmc #opt	Lscc-Bms result	Lscc-Bms $\bar{t}$
100	10	60.17^a	**60.20****	304.84	326.59	30	**60.20***	80.25	144.93	30	**60.20***	3.25
	20	58.13^a	**58.20***	341.35	485.49	30	**58.20***	89.40	318.07	29	**58.20***	2.87
	50	51.87^a	52.03	826.12	1990.85	20	52.07	147.26	1112.64	21	**52.10***	4.57
200	20	121.70^b	– –	– –	– –	0	120.27	672.79	– –	0	121.23	977.41
	40	116.70^b	– –	– –	– –	0	115.87	698.50	– –	0	**117.67***	1306.38
	100	104.57^a	– –	– –	– –	0	104.30	540.59	– –	0	**106.50***	1122.56
300	30	181.30^a	– –	– –	– –	0	178.10	781.43	– –	0	173.47	1811.81
	60	174.97^a	– –	– –	– –	0	171.80	498.21	– –	0	169.57	1646.61
	150	157.13^a	– –	– –	– –	0	155.93	988.86	– –	0	156.10	2010.08
400	40	242.70^b	– –	– –	– –	0	239.53	790.89	– –	0	220.67	1772.27
	80	233.23^a	– –	– –	– –	0	226.97	583.45	– –	0	215.10	1617.85
	200	208.77^a	– –	– –	– –	0	205.23	931.87	– –	0	199.43	1949.73
500	50	302.27^b	– –	– –	– –	0	295.90	746.95	– –	0	262.10	1490.81
	100	291.23^a	– –	– –	– –	0	284.67	936.73	– –	0	256.03	1678.48
	250	259.50^a	– –	– –	– –	0	255.83	1161.76	– –	0	240.17	1943.09
600	60	366.03^b	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	120	350.97^a	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	300	309.20^a	– –	– –	– –	0	– –	– –	– –	0	– –	– –
700	70	418.40^b	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	140	400.60^a	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	350	362.74^a	– –	– –	– –	0	– –	– –	– –	0	– –	– –
800	80	484.43^b	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	160	462.60^b	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	400	414.33^a	– –	– –	– –	0	– –	– –	– –	0	– –	– –
900	90	542.07^b	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	180	522.40^a	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	450	463.27^a	– –	– –	– –	0	– –	– –	– –	0	– –	– –
1000	100	605.10^b	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	200	583.30^a	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	500	514.80^b	– –	– –	– –	0	– –	– –	– –	0	– –	– –

the boxplots show the percentage reduction concerning the number of vertices of the original conflict graphs. If the reduction for an instance is at 60%, for example, this means that the reduction technique was able to remove 60% of the vertices of the original conflict graph. In the context of the RFLCS instances, we can state that the percentage reduction tends to grow with a growing string length and a growing alphabet size. Note that for long strings on large alphabets we were able to achieve reduction percentages of more than 90%. Concerning the LAPCS problem, it can be observed that the reduction percentages grow with an increasing number of arc annotations. However, they slightly increase with a growing input string length. This is due to the small alphabet size of four. Finally, it is worth mentioning that in the case of the real problem instances (set LAPCS-ARTI; Figure 8) we were able to achieve very high reduction percentages, sometimes well over 90%. This indicates the difference in structure between artificial and real problem instances.

The numerical results obtained by the three considered techniques after conflict graph reduction are provided in Tables A.9, A.10,A.11,A.12 that can be found in Appendix A. The structure of these tables is very similar to the one of Tables 2–5, which was described at the beginning of Section 4.1. The only difference is

Table 5
Experimental results for LAPCS instances LAPCS-REAL.

Inst. Name	Spec. Tech.	CPLEX result	CPLEX t	CPLEX t_{opt}	LMC result	LMC t	LMC t_{opt}	LSCC-BMS result	LSCC-BMS t
Real_1	268^b	– –	– –	– –	259	2691.58	– –	231	3504.69
Real_2	291^b	– –	– –	– –	283	637.94	– –	216	1088.45
Real_3	294^b	– –	– –	– –	284	104.13	– –	234	1580.28
Real_4	374^b	– –	– –	– –	**374**	34.59	– –	366	2148.66
Real_5	178^b	– –	– –	– –	**179***	6.04	– –	170	2336.97
Real_6	209^b	– –	– –	– –	206	30.64	– –	197	2181.59
Real_7	330^b	– –	– –	– –	**330**	43.61	– –	251	1461.38
Real_8	177^b	– –	– –	– –	175	3309.91	– –	173	448.26
Real_9	302^b	– –	– –	– –	**304***	44.36	– –	226	49.66
Real_10	361^a	– –	– –	– –	**361**	71.14	– –	272	496.70

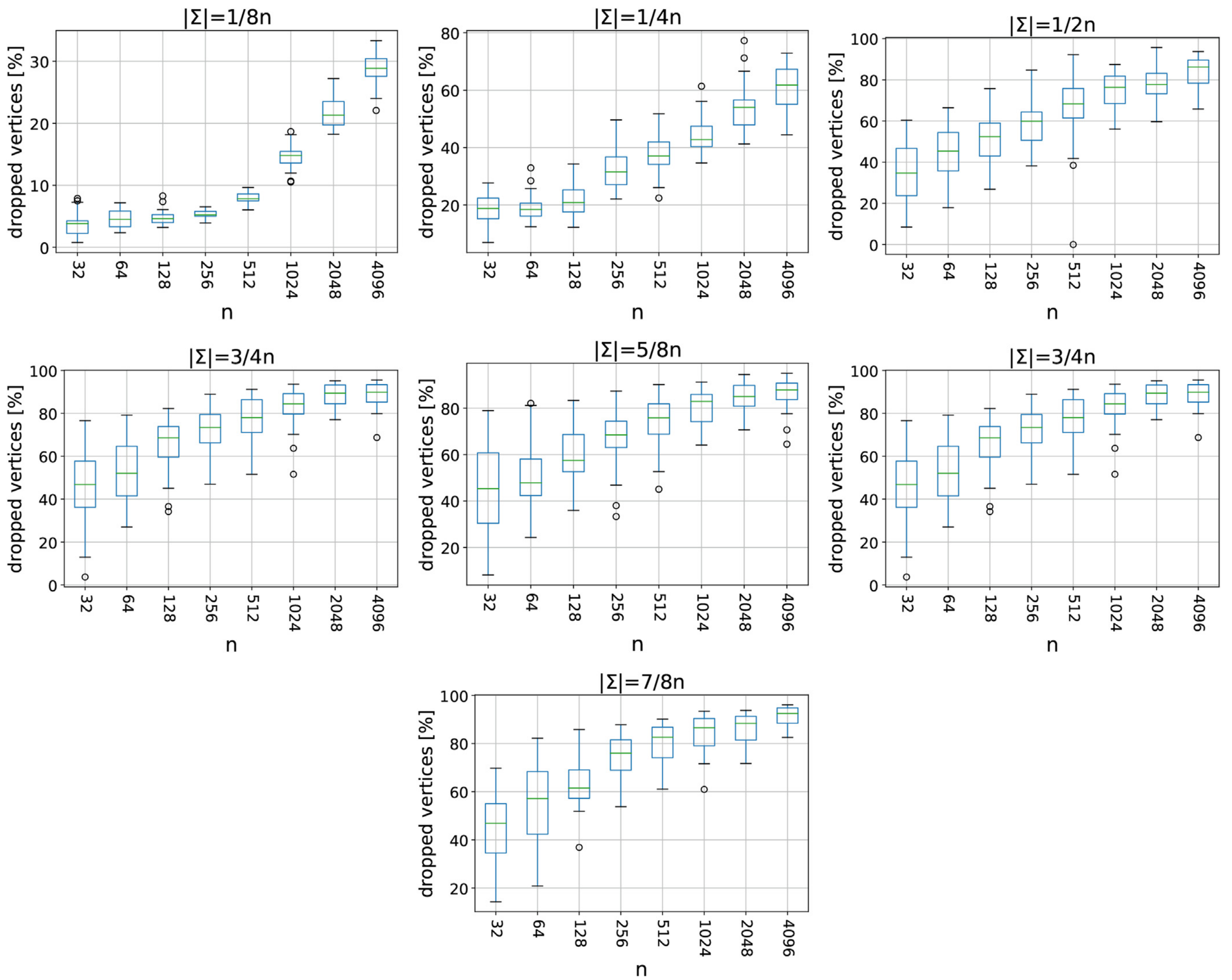


Fig. 5. Graph reduction (in %) for RFLCS instances from set RFLCS-SET1.

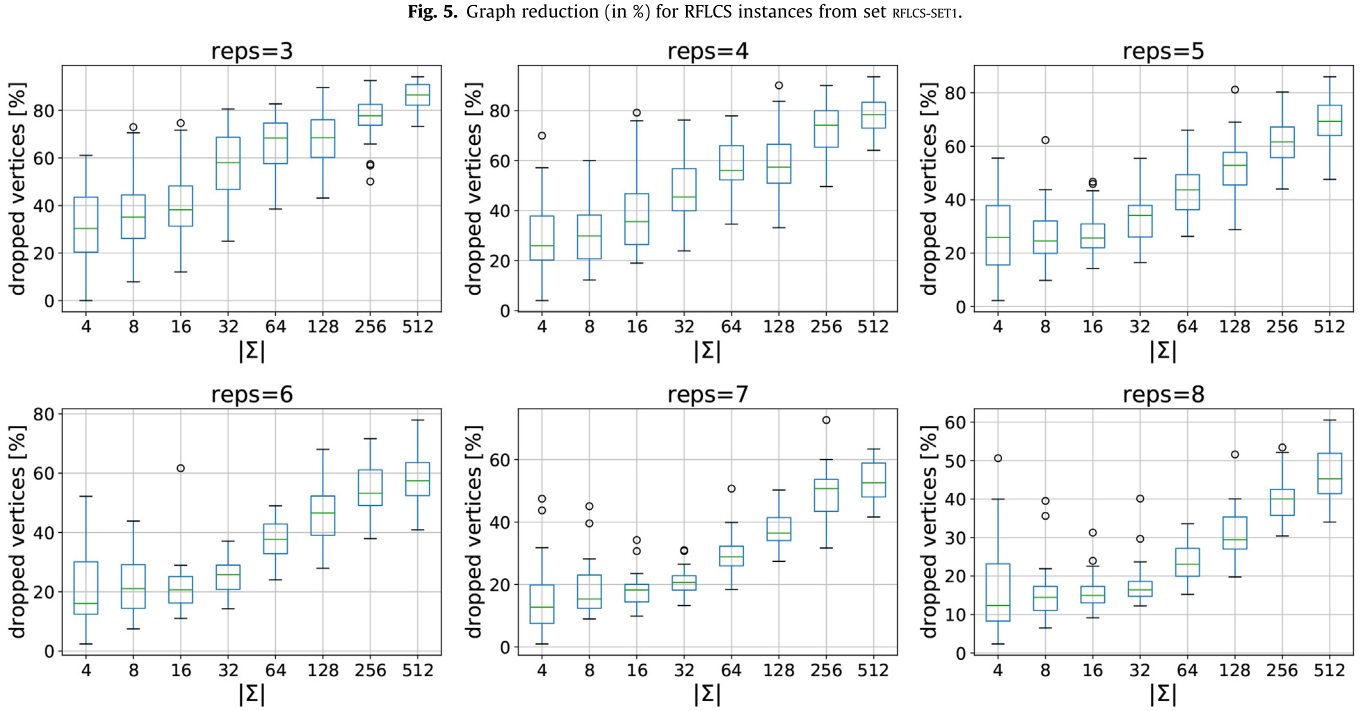


Fig. 6. Graph reduction (in %) for RFLCS instances from set RFLCS-SET2.

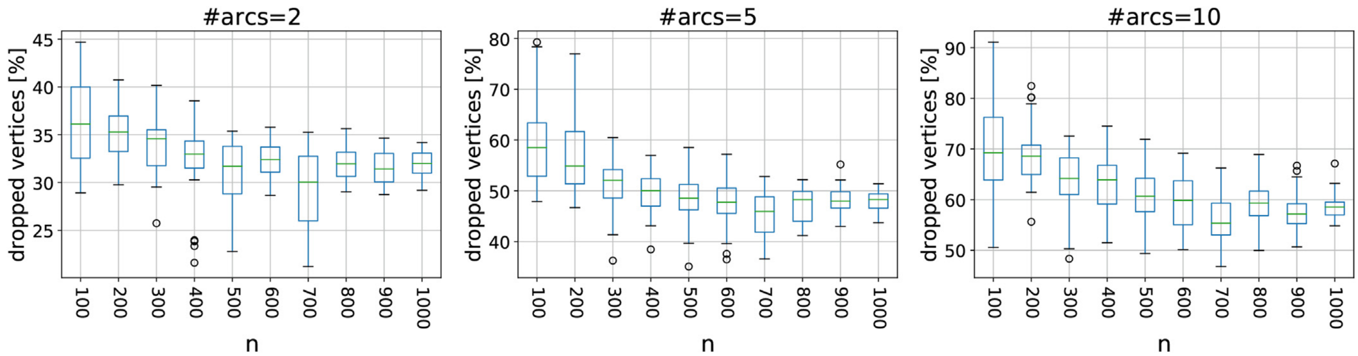


Fig. 7. Graph reduction (in %) for LAPCS instances from set LAPCS-ARTI.

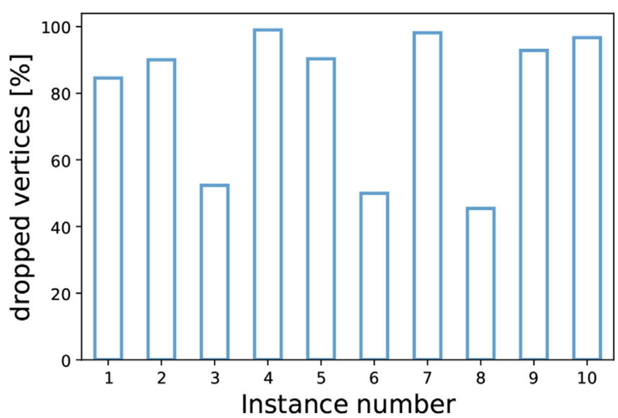


Fig. 8. Graph reduction (in %) for LAPCS instances from set LAPCS-REAL.

that the part on the state-of-the-art results—see the columns with heading "Spec. Tech."—is now extended by the corresponding computation times, that is, the times at which these results were obtained by the respective techniques.[4] The computation time of the fastest technique in each table row is underlined. Moreover, in order to relate the two sets of results, the values in the columns with heading **result** are marked in a different way. More specifically, values marked by a preceding =-symbol are equal to the values obtained by the same technique before graph reduction. Furthermore, values marked in *italic* font and by a preceding –-symbol are worse than the values obtained by the same technique before graph reduction, and values marked in **bold** font and by a preceding +-symbol are better than the corresponding values before conflict graph reduction. In order to relate the performance of a technique before graph reduction with its performance after graph reduction, we also computed a set of measures that are provided in Table 6 for CPLEX and LMC, and in Table 7 for LSCC-BMS. The measures regarding the exact techniques (see Table 6) are as follows.

1. **Measure E-M1** refers to those instances that were solved to optimality, both concerning the original conflict graph and the reduced conflict graph. In particular, it provides the average time saved for finding the best solution of a run (in seconds) after reducing the respective graph.
2. **Measure E-M2** is very similar, just that it refers to the average time saving for proving optimality.
3. **Measure E-M3** indicates the number of instances additionally solved to optimality after graph reduction.
4. **Measure E-M4** indicates the average improvement in solution quality (in percent) for all those instances for which feasible solutions can be found both before and after graph reduction, but for which optimality cannot be proven.
5. Finally, **measure E-M5** reports on the number of instances for which a feasible (and possibly optimal) solution can be found after graph reduction, and for which no feasible solution could be found before graph reduction.

In the context of the heuristic MC solver LSCC-BMS (see Table 7), measures H-M1–H-M4 can be described as follows.

1. In all those cases in which the same result is obtained by LSCC-BMS before and after conflict graph reduction, **measure H-M1** refers to the average time saving per instance (in seconds) for achieving this result.
2. **Measure H-M2** indicates the number of instances for which the result of LSCC-BMS improves after graph reduction.
3. **Measure H-M3** refers to the number of cases in which the result gets worse.
4. Finally, **measure H-M4** counts the number of instances for which LSCC-BMS can find a feasible solution after graph reduction, while before graph reduction LSCC-BMS was not able to find any feasible solution.

Remarks concerning the results for the RFLCS problem:

- The great beneficiary of the applied conflict graph reduction is CPLEX. CPLEX is now able to solve 1478 RFLCS-SET1 instances (out of 1680) and 1314 RFLCS-SET2 instances (out of 1440) to optimality, while LMC now solves 1366 RFLCS-SET1 instances and 1247 RFLCS-SET2 instances to optimality. Nevertheless, CPLEX still suffers from a sharp phase transition which, due to the graph reduction, has been moved to larger problem instances. Also the time savings achieved for finding the best solutions of a run and for proving optimality are much higher in the case of CPLEX when compared to those of LMC (see Table 6).
- The heuristic MC solver LSCC-BMS is also able to profit from the graph reduction. It provides an improved result for 296 RFLCS-SET1 instances and for 114 RFLCS-SET2 instances, while worse results are only produced in 27, respectively 9, cases. Moreover, in those cases in which LSCC-BMS obtains the same result before and after graph reduction, the average time saving per instance is approx. 77 s for the RFLCS-SET1 instances, and approx. 39 s for the RFLCS-SET2 instances.

After studying the results obtained for the LAPCS instances, the following observations can be made:

[4] Note that the state-of-the-art techniques for both the RFLCS and the LAPCS problem were executed on the same computers as CPLEX and LSCC-BMS.

Table 6
Differences in performance of the exact methods (CPLEX and LMC) summarized for the four different data sets. The five measures (E-M1–E-M5) are described in the text.

Data set	CPLEX					LMC				
	E-M1	E-M2	E-M3	E-M4	E-M5	E-M1	E-M2	E-M3	E-M4	E-M5
RFLCS-SET1	144.29	172.77	257	61.29	60	9.33	32.99	84	0.24	0
RFLCS-SET2	78.90	105.19	133	82.34	30	5.58	14.90	10	0.20	0
LAPCS-ARTI	330.89	526.45	30	−0.39	120	29.70	20.02	0	−0.0004	150
LAPCS-REAL	– –	– –	7	– –	9	– –	– –	5	0.0	0

- Concerning the set of artificial problem instances (LAPCS-ARTI), it can be observed that all three techniques are now able to provide solutions for some of the larger instances. CPLEX, for example, can now provide solutions for the instances with $n = 200$ and for the case $(n = 300, n_{\text{arcs}} = 30)$, for which no result was obtained before conflict graph reduction. However, while LSCC-BMS is able to improve its results for many instances (see cases $n \in \{200, \ldots, 500\}$), LSM is again not able to take much profit from the graph reduction. In fact, the results of LSM after graph reduction are sometimes even worse than before; see case $n = 200$ and $n_{\text{arcs}} \in \{20, 40\}$, for example. On average, LSM is not able to improve its results for those instances for which a feasible solution was obtained before and after conflict graph reduction, but for which optimality could not be proven; see measure E-H4 in Table 6. CPLEX is now able to solve 110 problem instances to optimality, while LSM can solve 80 problem instances, the same ones that it was able to solve before conflict graph reduction. Again, LSCC-BMS performs best when the performance of CPLEX and LSM starts to decline (see the cases with $n = 200$).
- Finally, the results—in particular those of CPLEX—for the real-life instances of set LAPCS-REAL are quite pleasing. CPLEX is able to solve seven out of 10 instances to optimality. In three of these cases, the best-known result from the literature is improved. LSM, on the other side, obtains exactly the same results as before conflict graph reduction, with the difference that optimality can be proven now for five out of the 10 problem instances. LSCC-BMS is again able to take profit from the graph reduction, improving its results in 9 out of 10 cases.

A summary of the obtained results in comparison to the current state of the art is provided in Table 8. Concerning the instances of set RFLCS-SET1, for example, our approaches were able to improve the current state-of-the-art algorithm from Blum and Blesa (2018c) in 23 out of 56 cases, the results were matched in 28 cases, and in only five cases our results were inferior to the state of the art. In general, Table 8 shows that our transformation-based approaches are very successful in the context of the RFLCS problem, while they only succeeded for the smaller instances of set LAPCS-ARTI and the real-life instances from set LAPCS-REAL.

Finally, the following observations can be made concerning the comparison of the computation times of our approaches with those of the state of the art. In the context of the RFLCS problem, our approaches are generally faster than the current state-of-the-art approaches, especially for instances of RFLCS-SET1 with alphabets of medium and large size. The state-of-the-art approaches are only faster for RFLCS-SET1 instances with small alphabet sizes and rather long input strings. The same happens for RFLCS-SET2 instances with large alphabet sizes and many repetitions. The computation time comparison concerning the LAPCS problem reflects the analysis from above concerning solution quality. In particular, our approaches—especially LSCC-BMS—are only faster than the current state of the art for the smallest LAPCS-ARTI instances with input string length $n = 100$. Starting from LAPCS-ARTI instances with $n = 200$, our approaches require considerably more computation time than state-of-the-art techniques. On the other side, in the context of the real life instances of set LAPCS-REAL our approaches are faster in eight out of 10 cases, reflecting the good results obtained for this instance set.

Table 7
Differences in performance of the heuristic method (LSCC-BMS) summarized for the four different data sets. The four measures (H-M1–H-M4) are described in the text.

Data set	LSCC-BMS			
	H-M1	H-M2	H-M3	H-M4
RFLCS-SET1	77.34	296	27	0
RFLCS-SET2	39.41	114	9	0
LAPCS-ARTI	176.21	280	33	30
LAPCS-REAL	– –	9	1	0

Table 8
Number of instances for which better, equally good, and worse solutions were obtained in comparison to the current state of the art.

	RFLCS-SET1	RFLCS-SET2	LAPCS-ARTI	LAPCS-REAL
Better	23	22	6	3
Equal	28	24	0	4
Worse	5	2	24	3

5. Conclusions and future work

In this work we proposed a way to transform longest common subsequence problem instances into instances of the maximum clique problem. Moreover, we defined a technique for the reduction of the resulting graphs, based on high-quality primal bounds. The benefits of this approach were experimentally studied in the context of two longest common subsequence variants: (1) the repetition-free longest common subsequence (RFLCS) problem and (2) the longest arc-preserving common subsequence (LAPCS) problem. Both problem variants are known to be $\mathcal{NP}$-hard even for two input strings. We compared the application of CPLEX for solving the maximum independent set problem, which is the complimentary problem of the maximum clique problem, with the application of recent heuristic and exact maximum clique solvers. The three approaches were applied both before and after graph reduction. The best results were obtained after graph reduction, even though the impact of graph reduction was very different for the three solvers. Summarizing, we were able to solve 2613 of the 3120 RFLCS instances to optimality. Moreover, 110 out of 900 artificially created LAPCS problem instances were solved to optimality. In the context of the LAPCS problem, it was especially pleasing to see seven out of 10 real-life instances solved to optimality for the first time.

Concerning future work, we plan to study further techniques for graph reduction in order to be able to apply the utilized solvers to even larger problem instances, in particular regarding the LAPCS problem. Moreover, our current graph reduction technique relies on the existence of a heuristic solution of reasonable quality to any tackled problem instance. In some cases, such a solution might not be available. In future work we plan to study the use of relaxed decision diagrams (Bergman et al., 2016) for the purpose of graph reduction in such cases. Finally, we also intend to study additional variants of the longest common subsequence problem.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Christian Blum: Methodology, Writing - original draft, Writing - review & editing, Supervision. **Marko Djukanovic:** Software, Conceptualization, Writing - original draft, Writing - review & editing. **Alberto Santini:** Conceptualization, Methodology. **Hua Jiang:** Software, Data curation. **Chu-Min Li:** Validation, Writing - review & editing. **Felip Manyà:** Methodology, Writing - review & editing. **Günter R. Raidl:** Methodology, Writing - review & editing, Supervision.

Acknowledgements

This work was supported by project CI-SUSTAIN funded by the Spanish Ministry of Science and Innovation (PID2019-104156GB-I00).

Appendix A. Tables showing the results after graph reduction

Tables A.9,A.10,A.11,A.12

Table A.9
Results obtained after graph reduction (RFLCS instances of RFLCS-SET1).

$\lvert\Sigma\rvert$	n	Spec. Tech.		CPLEX				LMC				LSCC+BMS	
		result	$\bar{t}$	**result**	$\bar{t}$	$\bar{t}_{\text{opt}}$	**#opt**	**result**	$\bar{t}$	$\bar{t}_{\text{opt}}$	**#opt**	**result**	$\bar{t}$
$n/8$	32	4.00	0.00	$^{=}$4.00	0.07	0.07	30	$^{=}$4.00	0.00	0.03	30	$^{=}$4.0	0.00
	64	8.00	0.00	$^{=}$8.00	0.52	0.52	30	$^{=}$8.00	0.00	0.11	30	$^{=}$8.0	0.00
	128	16.00	0.00	$^{=}$16.00	6.10	6.10	30	$^{=}$16.00	0.00	76.21	30	$^{=}$16.0	0.00
	256	31.97	0.03	$^{=}$31.97	202.49	202.49	30	$^{=}$31.90	18.82	– –	0	$^{=}$31.97	0.03
	512	63.27	38.89	$^{+}$**30.97**	1889.32	– –	0	$^{-}$*62.40*	347.07	– –	0	$^{=}$63.90	69.94
	1024	111.57	146.30	$^{=}$0.03	1062.21	– –	0	$^{=}$112.53	923.56	– –	0	$^{+}$**116.30***	1099.26
	2048	182.67	278.15	– –	– –	– –	0	$^{-}$*182.37*	1415.51	– –	0	$^{+}$**182.40**	1443.72
	4096	283.33	67.15	– –	– –	– –	0	$^{=}$281.37	1114.40	– –	0	$^{+}$**263.83**	2079.95
$n/4$	32	7.83	0.01	$^{=}$7.83	0.01	0.01	30	$^{=}$7.83	0.00	0.02	30	$^{=}$7.83	0.00
	64	14.67	0.08	$^{=}$14.67	0.08	0.08	30	$^{=}$14.67	0.00	0.03	30	$^{=}$14.67	0.00
	128	25.77	3.47	$^{=}$25.93	1.78	2.34	30	$^{=}$25.93	0.01	0.12	30	$^{=}$25.93	0.01
	256	43.70	10.39	$^{=}$43.97	9.20	19.41	30	$^{=}$43.97	0.13	0.67	30	$^{=}$43.97	0.12
	512	67.90	35.97	$^{+}$**68.57**	233.30	489.73	30	$^{=}$68.57	95.78	104.92	29	$^{=}$68.57	3.20
	1024	103.00	50.18	$^{+}$**104.53**	1169.51	1969.63	14	$^{-}$*103.73*	487.95	– –	0	$^{+}$**105.00***	482.73
	2048	154.33	150.72	$^{+}$**117.47**	1986.41	1001.01	3	$^{-}$*152.73*	513.24	– –	0	$^{+}$**153.93**	1229.79
	4096	226.67	452.60	$^{+}$**23.17**	648.79	– –	0	$^{+}$**224.63**	573.22	– –	0	$^{+}$**215.00**	1510.06
$3n/8$	32	8.77	0.01	$^{=}$8.77	0.00	0.00	30	$^{=}$8.77	0.00	0.02	30	$^{=}$8.77	0.00
	64	15.53	0.05	$^{=}$15.53	0.03	0.03	30	$^{=}$15.53	0.00	0.02	30	$^{=}$15.53	0.00
	128	24.90	0.45	$^{=}$24.90	0.33	0.34	30	$^{=}$24.90	0.00	0.04	30	$^{=}$24.90	0.01
	256	39.97	2.68	$^{=}$39.97	0.81	0.98	30	$^{=}$39.97	0.02	0.08	30	$^{=}$39.97	0.06
	512	59.77	12.80	$^{=}$59.97	9.15	14.02	30	$^{=}$59.97	0.14	0.58	30	$^{=}$59.97	0.39
	1024	90.50	34.73	$^{+}$**90.73**	30.62	49.38	30	$^{=}$90.73	2.93	19.30	30	$^{=}$90.73	5.81
	2048	130.57	51.96	$^{+}$**131.07***	323.31	369.50	28	$^{+}$**130.57**	117.06	410.36	13	$^{+}$**131.00**	726.79
	4096	191.37	98.06	$^{+}$**186.27**	739.49	653.72	23	$^{+}$**190.90**	399.25	– –	0	$^{+}$**189.07**	1441.64
$n/2$	32	8.87	0.01	$^{=}$8.87	0.00	0.00	30	$^{=}$8.87	0.00	0.02	30	$^{=}$8.87	0.00
	64	14.80	0.01	$^{=}$14.80	0.01	0.01	30	$^{=}$14.80	0.00	0.02	30	$^{=}$14.80	0.00
	128	22.93	0.03	$^{=}$22.93	0.05	0.06	30	$^{=}$22.93	0.00	0.03	30	$^{=}$22.93	0.00
	256	35.10	1.23	$^{=}$35.20	0.30	0.34	30	$^{=}$35.20	0.01	0.04	30	$^{=}$35.20	0.03
	512	53.10	5.70	$^{=}$53.13	1.41	1.78	30	$^{=}$53.13	0.03	0.15	30	$^{=}$53.13	0.16
	1024	79.03	12.43	$^{=}$79.13	4.46	5.16	30	$^{=}$79.13	0.24	0.60	30	$^{=}$79.13	3.37
	2048	115.30	53.49	$^{+}$**115.70***	26.72	29.61	30	$^{+}$**115.67**	116.09	350.85	28	$^{+}$**115.67**	174.36
	4096	167.47	89.48	$^{+}$**167.97***	114.74	158.87	30	$^{+}$**167.60**	36.16	208.53	18	$^{+}$**167.43**	664.34
$5n/8$	32	8.60	0.01	$^{=}$8.60	0.00	0.00	30	$^{=}$8.60	0.00	0.02	30	$^{=}$8.60	0.00
	64	13.30	0.08	$^{=}$13.30	0.00	0.00	30	$^{=}$13.30	0.00	0.02	30	$^{=}$13.30	0.00
	128	21.20	0.03	$^{=}$21.20	0.01	0.01	30	$^{=}$21.20	0.00	0.02	30	$^{=}$21.20	0.00
	256	32.53	0.34	$^{=}$32.53	0.08	0.09	30	$^{=}$32.53	0.00	0.03	30	$^{=}$32.53	0.00
	512	47.83	1.81	$^{=}$47.83	0.65	0.65	30	$^{=}$47.83	0.02	0.06	30	$^{=}$47.83	0.01
	1024	70.03	2.62	$^{=}$70.20	1.29	1.32	30	$^{=}$70.20	0.04	0.17	30	$^{=}$70.20	0.25
	2048	103.80	24.71	$^{+}$**103.97**	4.28	4.32	30	$^{=}$103.97	0.49	3.18	30	$^{+}$**103.97**	2.77
	4096	150.00	152.30	$^{+}$**150.57***	50.27	50.36	30	$^{+}$**150.43**	281.64	219.92	24	$^{+}$**150.40**	365.03
$3n/4$	32	8.17	0.01	$^{=}$8.17	0.00	0.00	30	$^{=}$8.17	0.00	0.02	30	$^{=}$8.17	0.00
	64	12.53	0.01	$^{=}$12.53	0.00	0.00	30	$^{=}$12.53	0.00	0.02	30	$^{=}$12.53	0.00
	128	19.70	0.07	$^{=}$19.70	0.00	0.00	30	$^{=}$19.70	0.00	0.02	30	$^{=}$19.70	0.00
	256	29.97	0.13	$^{=}$29.97	0.02	0.02	30	$^{=}$29.97	0.00	0.02	30	$^{=}$29.97	0.00
	512	44.53	0.59	$^{=}$44.57	0.17	0.19	30	$^{=}$44.57	0.00	0.04	30	$^{=}$44.57	0.02
	1024	65.07	7.84	$^{=}$65.20	0.52	0.52	30	$^{=}$65.20	0.02	0.09	30	$^{=}$65.20	0.07
	2048	94.53	13.11	$^{=}$94.67	0.87	0.90	30	$^{=}$94.67	0.04	0.16	30	$^{+}$**94.67**	1.29
	4096	136.57	60.44	$^{+}$**136.77***	11.63	11.80	30	$^{+}$**136.47**	171.71	3.79	26	$^{+}$**136.73**	26.59
$7n/8$	32	7.67	0.00	$^{=}$7.67	0.00	0.00	30	$^{=}$7.67	0.00	0.02	30	$^{=}$7.67	0.00
	64	11.57	0.00	$^{=}$11.57	0.00	0.00	30	$^{=}$11.57	0.00	0.02	30	$^{=}$11.57	0.00
	128	18.40	0.01	$^{=}$18.40	0.01	0.01	30	$^{=}$18.40	0.00	0.02	30	$^{=}$18.40	0.00
	256	27.80	0.04	$^{=}$27.80	0.01	0.01	30	$^{=}$27.80	0.00	0.02	30	$^{=}$27.80	0.00
	512	40.57	4.65	$^{=}$40.60	0.06	0.06	30	$^{=}$40.60	0.00	0.02	30	$^{=}$40.60	0.00
	1024	60.50	7.01	$^{=}$60.57	0.34	0.35	30	$^{=}$60.57	0.01	0.04	30	$^{=}$60.57	0.09
	2048	88.00	22.42	$^{=}$88.00	2.56	2.59	30	$^{=}$88.00	0.05	8.89	30	$^{=}$88.00	0.33
	4096	127.20	37.41	$^{+}$**127.37***	2.93	2.97	30	$^{+}$**127.37***	0.15	1.50	30	$^{+}$**127.37***	2.17

Table A.10
Results obtained after graph reduction (RFLCS instances of RFLCS-SET2).

$\lvert\Sigma\rvert$	*reps*	Spec. Tech.		Cplex				Lmc				Lscc-Bms	
		result	$\bar{t}$	**result**	$\bar{t}$	$\bar{t}_{\mathbf{opt}}$	**#opt**	**result**	$\bar{t}$	$\bar{t}_{\mathbf{opt}}$	**#opt**	**result**	$\bar{t}$
4	3	3.47	0.00	$^{=}$3.47	0.00	0.00	30	$^{=}$3.47	0.00	0.02	30	$^{=}$3.47	0.00
	4	3.77	0.00	$^{=}$3.77	0.00	0.00	30	$^{=}$3.77	0.00	0.02	30	$^{=}$3.77	0.00
	5	3.83	0.00	$^{=}$3.83	0.00	0.00	30	$^{=}$3.83	0.00	0.02	30	$^{=}$3.83	0.00
	6	3.90	0.00	$^{=}$3.90	0.00	0.00	30	$^{=}$3.90	0.00	0.02	30	$^{=}$3.90	0.00
	7	3.97	0.00	$^{=}$3.97	0.00	0.00	30	$^{=}$3.97	0.00	0.02	30	$^{=}$3.97	0.00
	8	3.97	0.00	$^{=}$3.97	0.01	0.01	30	$^{=}$3.97	0.00	0.02	30	$^{=}$3.97	0.00
8	3	6.23	0.00	$^{=}$6.23	0.00	0.00	30	$^{=}$6.23	0.00	0.02	30	$^{=}$6.23	0.00
	4	6.87	0.00	$^{=}$6.87	0.00	0.00	30	$^{=}$6.87	0.00	0.02	30	$^{=}$6.87	0.00
	5	7.40	0.00	$^{=}$7.40	0.00	0.00	30	$^{=}$7.40	0.00	0.02	30	$^{=}$7.40	0.00
	6	7.53	0.01	$^{=}$7.53	0.01	0.01	30	$^{=}$7.53	0.00	0.02	30	$^{=}$7.53	0.00
	7	7.70	0.00	$^{=}$7.70	0.02	0.02	30	$^{=}$7.70	0.00	0.02	30	$^{=}$7.70	0.00
	8	7.77	0.00	$^{=}$7.77	0.02	0.02	30	$^{=}$7.77	0.00	0.02	30	$^{=}$7.77	0.00
16	3	9.70	0.01	$^{=}$9.70	0.00	0.00	30	$^{=}$9.70	0.00	0.02	30	$^{=}$9.70	0.00
	4	11.57	0.01	$^{=}$11.57	0.01	0.01	30	$^{=}$11.57	0.00	0.02	30	$^{=}$11.57	0.00
	5	12.93	0.01	$^{=}$12.93	0.02	0.02	30	$^{=}$12.93	0.00	0.02	30	$^{=}$12.93	0.00
	6	14.00	0.01	$^{=}$14.00	0.05	0.05	30	$^{=}$14.00	0.00	0.02	30	$^{=}$14.00	0.00
	7	14.93	0.22	$^{=}$14.93	0.10	0.10	30	$^{=}$14.93	0.00	0.04	30	$^{=}$14.93	0.00
	8	14.80	0.39	$^{=}$14.80	0.17	0.17	30	$^{=}$14.80	0.00	0.05	30	$^{=}$14.80	0.01
32	3	16.13	0.02	$^{=}$16.13	0.01	0.01	30	$^{=}$16.13	0.00	0.02	30	$^{=}$16.13	0.00
	4	19.00	0.07	$^{=}$19.00	0.02	0.03	30	$^{=}$19.00	0.00	0.02	30	$^{=}$19.00	0.00
	5	21.63	0.37	$^{=}$21.63	0.28	0.30	30	$^{=}$21.63	0.00	0.03	30	$^{=}$21.63	0.01
	6	23.73	0.48	$^{=}$23.73	0.62	0.70	30	$^{=}$23.73	0.00	0.06	30	$^{=}$23.73	0.01
	7	25.53	0.78	$^{=}$25.57	1.79	1.93	30	$^{=}$25.57	0.02	0.13	30	$^{=}$25.57	0.02
	8	27.40	5.02	$^{=}$27.50	2.49	2.67	30	$^{=}$27.50	0.07	0.29	30	$^{=}$27.50	0.05
64	3	25.43	0.06	$^{=}$25.43	0.02	0.02	30	$^{=}$25.43	0.00	0.02	30	$^{=}$25.43	0.00
	4	30.37	0.77	$^{=}$30.37	0.24	0.26	30	$^{=}$30.37	0.00	0.03	30	$^{=}$30.37	0.01
	5	34.87	1.09	$^{=}$34.93	1.46	2.21	30	$^{=}$34.93	0.01	0.09	30	$^{=}$34.93	0.04
	6	39.07	10.66	$^{=}$39.13	5.46	8.21	30	$^{=}$39.13	0.04	0.25	30	$^{=}$39.13	0.11
	7	43.50	24.28	$^{=}$43.63	11.06	24.01	30	$^{=}$43.63	0.16	0.79	30	$^{=}$43.63	0.18
	8	45.17	35.41	$^{=}$45.53	35.48	84.08	30	$^{=}$45.53	1.69	6.06	30	$^{=}$45.53	0.44
128	3	36.70	0.67	$^{=}$36.77	0.18	0.18	30	$^{=}$36.77	0.00	0.03	30	$^{=}$36.77	0.02
	4	44.90	4.83	$^{=}$45.03	1.99	2.67	30	$^{=}$45.03	0.02	0.11	30	$^{=}$45.03	0.15
	5	53.23	17.58	$^{=}$53.43	7.90	10.76	30	$^{=}$53.43	0.12	0.42	30	$^{=}$53.43	0.30
	6	61.07	34.68	$^{=}$61.53	28.13	46.99	30	$^{=}$61.53	4.27	6.76	30	$^{=}$61.53	0.98
	7	67.90	52.14	$^{+}$**68.47**	125.08	421.63	30	$^{=}$68.47	9.47	57.64	30	$^{=}$68.47	1.98
	8	73.57	105.79	$^{+}$**74.50**	554.65	1321.01	18	$^{=}$74.30	608.14	544.04	13	$^{=}$74.60	10.65
256	3	54.97	0.30	$^{=}$55.03	0.74	0.79	30	$^{=}$55.03	0.02	0.06	30	$^{=}$55.03	0.04
	4	68.70	1.74	$^{=}$68.93	5.61	6.63	30	$^{=}$68.93	0.10	1.64	30	$^{=}$68.93	0.79
	5	81.00	29.22	$^{=}$81.43	28.37	40.16	30	$^{=}$81.43	3.51	7.68	30	$^{=}$81.43	3.13
	6	93.10	36.48	$^{+}$**93.60***	227.62	476.17	30	$^{+}$**93.17**	124.54	309.18	17	$^{+}$**93.60***	47.19
	7	103.50	98.94	$^{+}$**104.27**	667.00	1338.10	24	$^{-}$*103.13*	156.41	176.14	3	$^{+}$**104.47***	262.17
	8	113.70	176.80	$^{+}$**112.07**	2277.45	1367.53	1	$^{=}$113.10	344.01	– –	0	$^{+}$**115.00***	1009.44
512	3	81.57	29.98	$^{=}$81.63	0.52	0.54	30	$^{=}$81.63	0.02	0.35	30	$^{=}$81.63	0.12
	4	100.83	14.56	$^{+}$**101.13**	10.22	10.99	30	$^{+}$**101.10**	17.70	23.64	30	$^{=}$101.13	4.44
	5	120.43	92.05	$^{+}$**121.03***	147.62	209.48	30	$^{+}$**120.03**	249.84	866.25	14	$^{+}$**121.03***	226.89
	6	137.03	128.19	$^{+}$**136.97**	1335.77	771.58	11	$^{+}$**136.47**	499.84	47.78	1	$^{+}$**137.80***	1064.07
	7	154.57	295.09	$^{+}$**111.40**	2118.90	– –	0	$^{-}$*152.27*	823.67	– –	0	$^{+}$**153.33**	1619.19
	8	172.10	356.01	$^{+}$**13.97**	577.60	– –	0	$^{-}$*169.73*	620.92	– –	0	$^{+}$**168.70**	1557.06

Table A.11
Results for LAPCS instances of set LAPCS-ARTI after graph reduction.

n	n_{arcs}	Spec. Tech result	$\bar{t}$	Cplex result	$\bar{t}$	$\bar{t}_{\text{opt}}$	#opt	Lmc result	$\bar{t}$	$\bar{t}_{\text{opt}}$	#opt	Lscc-Bms result	$\bar{t}$
	10	60.17^a	39.01	$^=$60.20	6.57	8.56	30	$^=$60.20	38.48	85.52	30	$^=$60.20	0.46
100	20	58.13^a	52.11	$^=$58.20	12.46	22.41	30	$^=$58.17	15.96	361.00	29	$^=$58.20	0.80
	50	51.87^a	50.78	$^=$52.03	637.48	1150.86	24	$^=$52.07	145.51	1105.66	21	$^=$52.10	2.48
	20	121.70^b	38.28	$^+$**122.60**	956.11	1072.96	22	$^-$*120.20*	635.33	– –	0	$^+$**122.63***	671.81
200	40	116.70^b	61.36	$^+$**111.80**	2475.25	1972.97	4	$^-$*115.80*	627.34	– –	0	$^+$**118.40***	748.94
	100	104.57^a	143.72	$^+$**0.07**	302.02	– –	0	$^=$104.30	539.94	– –	0	$^+$**106.87***	1048.04
	30	181.30^a	178.30	$^+$**22.10**	659.71	– –	0	$^+$**178.23**	643.25	– –	0	$^+$**178.63**	1495.86
300	60	174.97^a	157.58	– –	– –	– –	0	$^=$171.80	507.46	– –	0	$^+$**172.20**	1338.14
	150	157.13^a	262.94	– –	– –	– –	0	$^+$**155.97**	991.12	– –	0	$^+$**156.97**	1410.26
	40	242.70^b	191.72	– –	– –	– –	0	$^+$**239.73**	650.89	– –	0	$^+$**230.77**	1616.86
400	80	233.23^a	322.25	– –	– –	– –	0	$^=$226.97	578.27	– –	0	$^+$**221.80**	1657.25
	200	208.77^a	378.41	– –	– –	– –	0	$^-$*205.17*	705.84	– –	0	$^+$**202.27**	2035.29
	50	302.27^b	250.46	– –	– –	– –	0	$^-$*295.83*	847.81	– –	0	$^+$**278.47**	1819.17
500	100	291.23^a	181.52	– –	– –	– –	0	$^+$**284.70**	891.25	– –	0	$^+$**266.70**	1470.81
	250	259.50^a	498.75	– –	– –	– –	0	$^-$*255.80*	1116.97	– –	0	$^+$**242.57**	1686.56
	60	366.03^b	324.61	– –	– –	– –	0	$^+$**356.83**	773.68	– –	0	$^+$**323.13**	1598.87
600	120	350.97^a	580.70	– –	– –	– –	0	$^+$**341.40**	987.06	– –	0	– –	– –
	300	309.20^a	370.43	– –	– –	– –	0	$^+$**306.63**	906.20	– –	0	– –	– –
	70	418.40^b	372.52	– –	– –	– –	0	$^+$**386.93**	708.66	– –	0	– –	– –
700	140	400.60^a	6.17	– –	– –	– –	0	$^+$**40.23**	169.30	– –	0	– –	– –
	350	362.74^a	698.11	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	80	484.43^b	420.71	– –	– –	– –	0	– –	– –	– –	0	– –	– –
800	160	462.60^b	572.70	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	400	414.33^a	797.52	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	90	542.07^b	516.09	– –	– –	– –	0	– –	– –	– –	0	– –	– –
900	180	522.40^a	534.96	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	450	463.27^a	897.39	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	100	605.10^b	535.42	– –	– –	– –	0	– –	– –	– –	0	– –	– –
1000	200	583.30^a	889.74	– –	– –	– –	0	– –	– –	– –	0	– –	– –
	500	514.80^b	664.53	– –	– –	– –	0	– –	– –	– –	0	– –	– –

Table A.12
Results for LAPCS instances of set LAPCS-REAL after graph reduction.

Inst. Name	Spec. Tech. results	t	Cplex result	t	t_{opt}	Lmc result	t	t_{opt}	Lscc-Bms result	t
Real_1	268^b	80.92	$^+$**273***	339.70	339.76	$^=$259	2489.55	– –	$^+$**272**	847.60
Real_2	291^b	84.93	$^+$**291**	21.84	21.85	$^=$283	416.57	– –	$^+$**291**	903.86
Real_3	294^b	171.78	– –	– –	– –	$^=$284	49.69	– –	$^+$**263**	1360.83
Real_4	374^b	0.02	$^+$**374**	0.02	0.02	$^=$374	0.01	0.07	$^+$**374**	0.00
Real_5	178^b	58.96	$^+$**179**	4.62	4.63	$^=$179	1.38	351.62	$^+$**179**	11.32
Real_6	209^b	131.35	$^+$**0**	2376.75	– –	$^=$206	15.19	– –	$^+$**204**	2441.29
Real_7	330^b	7.40	$^+$**330**	0.05	0.05	$^=$330	0.73	1.12	$^+$**330**	758.07
Real_8	177^b	174.40	$^+$**1**	1281.87	– –	$^=$175	2760.20	– –	$^-$*172*	3025.38
Real_9	302^b	24.89	$^+$**304**	1.47	1.81	$^=$304	16.41	54.18	$^+$**304**	498.12
Real_10	361^a	0.49	$^+$**361**	0.23	0.23	$^=$361	1.49	10.50	$^+$**361**	74.89

Appendix B. Supplementary data

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.cor.2020.105089.

References

Iliopoulos, C.S., Sohel Rahman, M., 2009. A new efficient algorithm for computing the longest common subsequence. Theory Comput. Syst. 45, 355–371.

Castelli, M., Beretta, S., Vanneschi, L., 2013. A hybrid genetic algorithm for the repetition free longest common subsequence problem. Oper. Res. Lett. 41, 644–649.

Gusfield, D., 1997. Algorithms on Strings, Trees, and Sequences, Computer Science and Computational Biology. Cambridge University Press, Cambridge.

Smith, T., Waterman, M., 1981. Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197.

Jiang, T., Lin, G., Ma, B., Zhang, K., 2002. A general edit distance between RNA structures. J. Comput. Biol. 9, 371–388.

Kruskal, J.B., 1983. An overview of sequence comparison: time warps, string edits, and macromolecules. SIAM Rev. 25, 201–237.

Brisk, P., Kaplan, A., Sarrafzadeh, M., 2004. Area-efficient instruction set synthesis for reconfigurable system-on-chip design, in: Proceedings of DAC 2004 – The 41st Design Automation Conference, IEEE press, pp. 395–400..

Storer, J., 1988. Data Compression: Methods and Theory. Computer Science Press, MD.

Aho, A., Hopcroft, J., Ullman, J., 1983. Data Structures and Algorithms. Addison-Wesley, Reading, MA.

Maier, D., 1978. The complexity of some problems on subsequences and supersequences. J. ACM 25, 322–336.
Adi, S.S., Braga, M.D.V., Fernandes, C.G., Ferreira, C.E., Martinez, F.V., Sagot, M.-F., Stefanes, M.A., Tjandraatmadja, C., Wakabayashi, Y., 2010. Repetition-free longest common subsquence. Discr. Appl. Math. 158, 1315–1324.
Tsai, Y.-T., 2003. The constrained longest common subsequence problem. Inf. Process. Lett. 88, 173–176.
Chen, Y.-C., Chao, K.-M., 2011. On the generalized constrained longest common subsequence problems. J. Combinat. Optim. 21, 383–392.
Bonizzoni, P., Della Vedova, G., Dondi, R., Pirola, Y., 2010. Variants of constrained longest common subsequence. Inf. Process. Lett. 110, 877–881.
Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M., 1999. The maximum clique problem. In: Handbook of Combinatorial Optimization. Springer, pp. 1–74.
Lee, E.K., Easton, T., Gupta, K., 2006. Novel evolutionary models and applications to sequence alignment problems. Ann. Oper. Res. 148, 167–187.
Jiang, H., Li, C.-M., Manyà, F., 2016. Combining efficient preprocessing and incremental MaxSAT reasoning for MaxClique in large graphs, in: Proceedings of ECAI 2016–22nd European Conference on Artificial Intelligence, pp. 939–947..
Li, C.-M., Jiang, H., Manyà, F., 2017. On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem. Comput. Oper. Res. 84, 1–15.
Wang, Y., Cai, S., Yin, M., 2016. Two efficient local search algorithms for maximum weight clique problem., in. In: Proceedings of AAAI 2016 – Conference on Artificial Intelligence, pp. 805–811.
Blum, C., Festa, P., 2016. Metaheuristics for String Problems in Bio-informatics. John Wiley & Sons.
Lee, E.K., Gupta, K., 2009. Algorithms for genomic analysis. In: Floudas, C.A., Pardalos, P.M. (Eds.), Encyclopedia of Optimization. Springer, US, pp. 33–54.
Chan, H.-T., Yang, C.-B., Peng, Y.-H., 2016. The generalized definitions of the two-dimensional largest common substructure problems, in: Proceedings of the 33rd Workshop on Combinatorial Mathematics and Computation Theory, National Taiwan University, Department of Mathematics, pp. 1–12..
Wang, Q., Korkin, D., Shang, Y., 2011. A fast multiple longest common subsequence (MLCS) algorithm. IEEE Trans. Knowl. Data Eng. 23, 321–334.
Li, Y., Wang, Y., Zhang, Z., Wang, Y., Ma, D., Huang, J., 2016. A novel fast and memory efficient parallel MLCS algorithm for long and large-scale sequences alignments, in: IEEE 32nd International Conference on Data Engineering, pp. 1170–1181..
Peng, Z., Wang, Y., 2017. A novel efficient graph model for the multiple longest common subsequences (mlcs) problem. Front. Genet. 8, 104.
Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4, 100–107.
Wang, Q., Pan, M., Shang, Y., Korkin, D., 2010. A fast heuristic search algorithm for finding the longest common subsequence of multiple strings, in: Fox, M., Poole, D. (Eds.), Proceedings of AAAI 2010 – Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI press, pp. 1287–1292..
Djukanovic, M., Raidl, G.R., Blum, C., 2019. Finding Longest Common Subsequences: New Hybrid A* Search Results, Technical Report AC-TR-19-008. Algorithms and Complexity Group, TU Wien. http://www.ac.tuwien.ac.at/files/tr/ac-tr-19-008.pdf.
Jiang, T., Li, M., 1995. On the approximation of shortest common supersequences and longest common subsequences. SIAM J. Comput. 24, 1122–1139.
Bonizzoni, P., Della Vedova, G., Mauri, G., 2001. Experimenting an approximation algorithm for the lcs. Discr. Appl. Math. 110, 13–24.
Tsai, Y., Hsu, J., 2002. An approximation algorithm for multiple longest common subsequence problems, in: Proceeding of the 6th World Multiconference on Systemics, Cybernetics and Informatics, SCI, pp. 456–460..
Huang, K., Yang, C.-B., Tseng, K.-T., et al., 2004. Fast algorithms for finding the common subsequence of multiple sequences. In: Proceedings of the International Computer Symposium. IEEE press, pp. 1006–1011.
Blin, G., Bonizzoni, P., Dondi, R., Sikora, F., 2012. On the parameterized complexity of the repetition free longest common subsequence problem. Inf. Process. Lett. 112, 272–276.
Blum, C., Blesa, M.J., 2018a. A comprehensive comparison of metaheuristics for the repetition-free longest common subsequence problem. J. Heurist. 24, 551–579.
Evans, P.A., 1999a. Finding common subsequences with arcs and pseudoknots, in: Crochemore, M., Paterson, M. (Eds.), Proceedings of CPM 1999–10th Annual Symposium on Combinatorial Pattern Matching, Volume 1645 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, pp. 270–280..
Evans, P.A., 1999b. Algorithms and Complexity for Annotated Sequence Analysis, Ph. D. thesis, University of Victoria..
Evans, P.A., 1999c. Finding common subsequences with arcs and pseudoknots, in: Annual Symposium on Combinatorial Pattern Matching, Springer, pp. 270–280..
Jiang, T., Lin, G., Ma, B., Zhang, K., 2004. The longest common subsequence problem for arc-annotated sequences. J. Discr. Algorithms 2, 257–270.
Gramm, J., Guo, J., Niedermeier, R., 2006. Pattern matching for arc-annotated sequences. ACM Trans. Algorithms 2, 44–65.
Gorbenko, A., Popov, V., 2012. The c-fragment longest arc-preserving common subsequence problem. IAENG Int. J. Comput. Sci. 39, 231–238.
Blum, C., Blesa, M.J., 2018b. Hybrid techniques based on solving reduced problem instances for a longest common subsequence problem. Appl. Soft Comput. 62, 15–28.
Brown, J.W., 1999. The ribonuclease P database. Nucl. Acids Res. 27, 314. 314.
Blum, C., Blesa, M.J., López-Ibáñez, M., 2009. Beam search for the longest common subsequence problem. Comput. Oper. Res. 36, 3178–3186.
Fraser, C.B., 1995. Subsequences and supersequences of strings, Ph.D. thesis, University of Glasgow..
Wang, Q., Korkin, D., Shang, Y., 2011. A fast multiple longest common subsequence (MLCS) algorithm. IEEE Trans. Knowl. Data Eng. 23, 321–334.
Wang, Q., Pan, M., Shang, Y., Korkin, D., 2010. A fast heuristic search algorithm for finding the longest common subsequence of multiple strings, in: Proceedings of AAAI 2010 – Conference on Artificial Intelligence, pp. 1287–1292..
Wang, L., Wang, S.-Y., Xu, Y., 2012. An effective hybrid EDA-based algorithm for solving multidimensional knapsack problem. Expert Syst. Appl. 39, 5593–5599.
Inenaga, S., Hyyrö, H., 2018. A hardness result and new algorithm for the longest common palindromic subsequence problem. Inf. Process. Lett. 129, 11–15.
C. M. Li, F. Manya, Maxsat, hard and soft constraints., in: Handbook of satisfiability, volume 185, 2009, pp. 613–631..
Blum, C., Blesa, M.J., 2018c. A comprehensive comparison of metaheuristics for the repetition-free longest common subsequence problem. J. Heurist. 24, 551–579.
Bergman, D., Cire, A.A., Van Hoeve, W.-J., Hooker, J., 2016. Decision Diagrams for Optimization, vol. 1. Springer.