Optimization Letters manuscript No.
(will be inserted by the editor)

Computational Performance Evaluation of Two Integer Linear Programming Models for the Minimum Common String Partition Problem

Christian Blum · Günther R. Raidl

Received: date / Accepted: date

Abstract In the minimum common string partition (MCSP) problem two related input strings are given. "Related" refers to the property that both strings consist of the same set of letters appearing the same number of times in each of the two strings. The MCSP seeks a minimum cardinality partitioning of one string into non-overlapping substrings that is also a valid partitioning for the second string. This problem has applications in bioinformatics e.g. in analyzing related DNA or protein sequences. For strings with lengths less than about 1000 letters, a previously published integer linear programming (ILP) formulation yields, when solved with a state-of-the-art solver such as CPLEX, satisfactory results. In this work, we propose a new, alternative ILP model that is compared to the former one. While a polyhedral study shows the linear programming relaxations of the two models to be equally strong, a comprehensive experimental comparison using real-world as well as artificially created benchmark instances indicates substantial computational advantages of the new formulation.

Keywords Minimum Common String Partition · Bioinformatics · Integer Linear Programming · Computational Comparision

1 Introduction

String problems related to DNA and/or protein sequences are abundant in bioinformatics. Well-known examples include the longest common subsequence

C. Blum
Ikerbasque, Basque Foundation for Science, Bilbao, Spain
University of the Basque Country UPV/EHU, San Sebastian, Spain
E-mail: christian.blum@ehu.es

G. R. Raidl
Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Vienna, Austria
E-mail: raidl@ads.tuwien.ac.at

problem and its variants [15,23], the shortest common supersequence problem [10], and string consensus problems such as the *far from most string* problem and the *close to most string* problem [21,20]. Many of these problems are strongly *NP*-hard [11] and also computationally very challenging.

This work deals with a string problem which is known as the *minimum common string partition* (MCSP) problem. The MCSP problem can technically be described as follows. Given are two *related* input strings s^1 and s^2 which are both of length n over a finite alphabet Σ. The term *related* refers to the fact that each letter appears the same number of times in each of the two input strings. Note that being related implies that s^1 and s^2 have the same length. A valid solution to the MCSP problem is obtained by partitioning s^1 (resp. s^2) into a set P^1 (resp. P^2) of non-overlapping substrings such that $P^1 = P^2$. The optimization goal consists in finding a valid solution such that $|P^1| = |P^2|$ is minimal.

Consider the following example. Given are sequences $s^1 = \mathbf{AGACTG}$ and $s^2 = \mathbf{ACTAGG}$. Obviously, s^1 and s^2 are related because **A** and **G** appear twice in both input strings, while **C** and **T** appear once. A trivial valid solution can be obtained by partitioning both strings into substrings of length one, that is, $P^1 = P^2 = \{\mathbf{A}, \mathbf{A}, \mathbf{C}, \mathbf{T}, \mathbf{G}, \mathbf{G}\}$. The objective value of this solution is six. However, the optimal solution, with objective value three, is $P^1 = P^2 = \{\mathbf{ACT}, \mathbf{AG}, \mathbf{G}\}$.

The MCSP problem has applications, for example, in the bioinformatics field. Chen et al. [3] point out that the MCSP problem is closely related to the problem of sorting by reversals with duplicates, a key problem in genome rearrangement.

1.1 History of Research for the MCSP Problem

The original definition of the MCSP problem by Chen et al. [3] was inspired by computational problems arising in the context of genome rearrangement such as: May a given DNA string possibly be obtained by reordering subsequences of another DNA string? In the meanwhile, the general version of the problem was shown to be *NP*-hard [12]. Other papers concerning problem hardness consider problem variants such as, for example, the k-MCSP problem in which each letter occurs at most k times in each input string. The 2-MCSP problem was shown to be APX-hard in [12]. Jiang et al. [16] proved that the decision version of the MCSP^c problem—where c indicates the size of the alphabet—is *NP*-complete when $c \geq 2$.

A lot of research has been done concerning the approximability of the problem. Cormode and Muthukrishnan [5], for example, proposed an $O(\log n \log^* n)$-approximation for the *edit distance with moves* problem, which is a more general case of the MCSP problem. Other approximation approaches were proposed in [22,19]. Chrobak et al. [4] studied a simple greedy approach for the MCSP problem, showing that the approximation ratio concerning the 2-MCSP problem is 3, and for the 4-MCSP problem the approximation ratio

is in $\Omega(\log n)$. In the case of the general MCSP problem, the approximation ratio lies between $\Omega(n^{0.43})$ and $O(n^{0.67})$, assuming that the input strings use an alphabet of size $O(\log n)$. Later Kaplan and Shafir [17] improved the lower bound to $\Omega(n^{0.46})$. Kolman proposed a modified version of the simple greedy algorithm with an approximation ratio of $O(k^2)$ for the k-MCSP [18]. Recently, Goldstein and Lewenstein [13] proposed a greedy algorithm for the MCSP problem that runs in $O(n)$ time. He [14] introduced another a greedy algorithm with the aim of obtaining better average results.

Damaschke [6] was the first one to study the fixed-parameter tractability (FPT) of the problem. Later, Jiang et al. [16] showed that both the k-MCSP and MCSP^c problems admit FPT algorithms when k and c are constant parameters. Fu et al. [9] proposed an $O(2^n n^{O(1)})$ time algorithm for the general case and an $O(n(\log n)^2)$ time algorithm applicable under certain constraints.

Finally, in recent years researchers have also focused on algorithms for deriving high quality solutions in practical settings. Ferdous and Sohel Rahman [7,8], for example, developed a $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System metaheuristic. Blum et al. [1] proposed a probabilistic tree search approach. Both works applied their algorithm to a range of artificial and real DNA instances from [7]. The first integer linear programming (ILP) model, as well as a heuristic approach on the basis of the proposed ILP model, was presented in [2]. The heuristic is a 2-phase approach which—in the first phase—aims at covering most of the input strings with few but long substrings, while—in the second phase—the so-far uncovered parts of the input strings are covered in the best way possible. Experimental results showed that for smaller problem instances with $n < 1000$ applying a solver such as CPLEX[1] to the proposed ILP is currently state-of-the-art. For larger problem instances, runtimes are typically too high and best results are usually obtained by the heuristic from [2].

1.2 Contribution of this Work

In this paper we introduce an alternative ILP model for solving the MCSP problem. We show that the LP-relaxations of both models are equally strong from a theoretical point of view. An extensive experimental comparison with the model from [2] shows, however, that CPLEX is able to derive feasible integer solutions much faster with the new model. Moreover, the results when given the same computation time as for solving the existing ILP model are significantly better.

1.3 Organization of the Paper

The remainder is organized as follows. In Section 2, the ILP model from [2] as well as the newly proposed ILP model are described. A polyhedral comparison of the two models is performed in Section 3. The experimental evaluation

[1] `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer`

on problem instances from the related literature as well as on newly generated problem instances is provided in Section 4. Finally, in Section 5 we draw conclusions and give an outlook on future work.

2 ILP Models for the MCSP

In the following we first review the existing ILP model for solving the MCSP as proposed in [2]. Subsequently, the new alternative model is presented.

2.1 Existing ILP Model

The existing ILP model from [2] is based on the notion of *common blocks*. Therefore we will henceforth refer to this model as the *common blocks model*. A common block b_i of input strings s^1 and s^2 is a triple (t_i, k_i^1, k_i^2) where t_i is a string which appears as substring in s^1 at position k_i^1 and in s^2 at position k_i^2, with $k_i^1, k_i^2 \in \{1, \ldots, n\}$. Let the length of a common block b_i be its string's length, i.e., $|t_i|$. Let us now consider the set $B = \{b_1, \ldots, b_m\}$ of all existing common blocks of s^1 and s^2. Any valid solution $\mathcal{S}$ to the MCSP problem can then be expressed as a subset of B, i.e., $\mathcal{S} \subset B$, such that:

1. $\sum_{b_i \in \mathcal{S}} |t_i| = n$, that is, the sum of the lengths of the common blocks in $\mathcal{S}$ is equal to the length of the input strings.
2. For any two common blocks $b_i, b_j \in \mathcal{S}$ it holds that their corresponding strings neither overlap in s^1 nor in s^2.

The ILP uses for each common block $b_i \in B$ a binary variable x_i indicating its selection in the solution. In other words, if $x_i = 1$, the corresponding common block b_i is selected for the solution, and if $x_i = 0$, common block b_i is not selected.

$$
\begin{aligned}
(\text{ILP}_{\text{cb}}) \quad \min \quad & \sum_{i=1}^{m} x_i & & & (1)\\
\text{s.t.} \quad & \sum_{i \in \{1,\ldots,m | k_i^1 \leq j < k_i^1 + |t_i|\}} x_i = 1 & \text{for } j = 1, \ldots, n & & (2)\\
& \sum_{i \in \{1,\ldots,m | k_i^2 \leq j < k_i^2 + |t_i|\}} x_i = 1 & \text{for } j = 1, \ldots, n & & (3)\\
& x_i \in \{0, 1\} & \text{for } i = 1, \ldots, m
\end{aligned}
$$

The objective function (1) minimizes the number of selected common blocks. Equations (2) ensure that each position $j = 1, \ldots, n$ of string s^1 is covered by exactly one selected common block and selected common blocks also do not overlap. Equations (3) ensure the same with respect to s^2. Note

that equations (2) (and also (3)) implicitly guarantee that the sum of the lengths of the selected blocks is n as

$$\sum_{i=1}^{m} |t_i| \cdot x_i = \sum_{i=1}^{m} \sum_{j=k_i^1}^{k_i^1+|t_i|-1} x_j = \sum_{j=1}^{n} \sum_{i \in \{1,\ldots,m | k_i^1 \leq j < k_i^1+|t_i|\}} x_i = n.$$

Finally, note that the number of variables in model ILP_{cb} is of order $O(n^3)$.

2.2 An Alternative ILP Model: The Common Substrings Model

An aspect which the above model does not effectively exploit is the fact that, frequently, some string appears multiple times at different positions as substring in s^1 and/or s^2. For example, assume that string **AC** appears five times in s^1 and four times in s^2. Model ILP_{cb} will then consider $5 \cdot 4 = 20$ different common blocks, one for each pairing of an occurrence in s^1 and in s^2. Especially when the cardinality of the alphabet is low and n large, it is likely that some smaller strings appear very often and induce a huge set of possible common blocks B. To overcome this disadvantage, we propose the following alternative modeling approach.

Let T denote the set of all (unique) strings that appear as substrings at least once in both s^1 and s^2. For each $t \in T$, let Q_t^1 and Q_t^2 denote the set of all positions between 1 and n at which t starts in input strings s^1 and s^2, respectively.

We now use binary variables $y_{t,k}^1$ for each $t \in T$, $k \in Q_t^1$, and $y_{t,k}^2$ for each $t \in T$, $k \in Q_t^2$. If and only if $y_{t,k}^i = 1$, the occurance of string $t \in T$ at position $k \in Q_t^i$ in input string s_i is selected for the solution (where $i \in \{1, 2\}$). The new alternative model, henceforth also referred to as the *common substrings model*, can then be expressed as follows.

$$(\text{ILP}_{\text{cs}}) \quad \min \quad \sum_{t \in T} \sum_{k \in Q_t^1} y_{t,k}^1 \tag{4}$$

$$\text{s.t.} \quad \sum_{t \in T} \sum_{k \in Q_t^1 | k \leq j < k+|t|} y_{t,k}^1 = 1 \qquad \text{for } j = 1, \ldots, n \tag{5}$$

$$\sum_{t \in T} \sum_{k \in Q_t^2 | k \leq j < k+|t|} y_{t,k}^2 = 1 \qquad \text{for } j = 1, \ldots, n \tag{6}$$

$$\sum_{k \in Q_t^1} y_{t,k}^1 = \sum_{k \in Q_t^2} y_{t,k}^2 \qquad \text{for } t \in T \tag{7}$$

$$y_{t,k}^1 \in \{0, 1\} \qquad \text{for } t \in T,\ k \in Q_t^1$$

$$y_{t,k}^2 \in \{0, 1\} \qquad \text{for } t \in T,\ k \in Q_t^2$$

The objective function (4) counts the number of chosen substrings; note that $\sum_{t\in T}\sum_{k\in Q_t^2} y_{t,k}^2$ would yield the same value. Equations (5) and (6) ensure that for each position $j = 1,\ldots,n$ of input string s^1 (respectively, s^2) exactly one covering substring is chosen. These equations consider for each position j all substrings $t \in T$ for which the starting position k is at most j and less than $k+|t|$. Equations (7) ensure that each string $t \in T$ is chosen the same number of times within s^1 and s^2. Similarly as in ILP_{cb}, the requirement that the sum of the lengths of the selected substrings has to sum up to n follows implicitly from (5) and (6).

Concerning the number of variables involved in model ILP_{cs}, the following can be observed. A string of length n has exactly $n(n-1)/2$ possibly partly equal substrings of size greater than zero. In the worst case, the model uses one y variable for each of these substrings for s^1 and s^2, respectively. In case some substring t appears multiple times at different positions, it will only be considered once in $|T|$, but nevertheless its different occurrences appear in Q_t^1 and Q_t^2 and thus the number of y variables stays $n(n-1)/2$. When some substring of s^1 does not appear in s^2 or vice versa, no respective y variable(s) are considered and the overall number of variables is smaller. Therefore, in the general case, the number of variables of the new model is bounded by $O(n^2)$ and there are also $O(n^2)$ constraints.

3 Polyhedral Comparison

We compare the two ILP models by projecting solutions of ILP_{cb} expressed in terms of variables x_i, $i = 1,\ldots,m$, into the space of variables $y_{t,k}^1$, $t \in T$, $k \in Q_t^1$, and $y_{t,k}^2$, $t \in T$, $k \in Q_t^2$, from ILP_{cs}. A corresponding solution is obtained by

$$y_{t,k}^1 = \sum_{i\in\{1,\ldots,m|t_i=t\wedge k_i^1=k\}} x_i \quad \text{and} \quad y_{t,k}^2 = \sum_{i\in\{1,\ldots,m|t_i=t\wedge k_i^2=k\}} x_i. \quad (8)$$

Let LP_{cb} and LP_{cs} be the linear programming relaxations of models ILP_{cb} and ILP_{cs}, respectively, obtained by relaxing the integrality conditions. In the following we show that both models describe the same polyhedron in the space of y-variables and are thus equally strong from a theoretical point.

Lemma 1 *The polyhedron defined by* LP_{cb} *is contained in* LP_{cs}.

Proof We show that for any feasible solution to LP_{cb}, the solution in terms of the y-variables obtained by (8) is also feasible in LP_{cs}. For equations (5) replacing $y_{t,k}^1$ yields

$$\sum_{t\in T}\sum_{k\in Q_t^1|k\le j<k+|t|}\sum_{i\in\{1,\ldots,m|t_i=t\wedge k_i^1=k\}} x_i = \sum_{i\in\{1,\ldots,m|k_i^1\le j<k_i^1+|t_i^1|\}} x_i, \quad (9)$$

which corresponds to the left side of (2) and is thus always equal to one. Equations (6) are correspondingly fulfilled. For constraints (7) we obtain for each $t \in T$

$$\sum_{k \in P_t^1} \sum_{i \in \{1,\ldots,m | t_i = t \wedge k_i^1 = k\}} x_i = \sum_{i \in \{1,\ldots,m | t_i = t\}} x_i = \sum_{k \in P_t^2} \sum_{i \in \{1,\ldots,m | t_i = t \wedge k_i^2 = k\}} x_i,$$

and they are therefore also always fulfilled. Last but not least, also $0 \leq y_{t,k}^1 \leq 1$ and $0 \leq y_{t,k}^2 \leq 1$ trivially hold due to (2) and (3).

Lemma 2 *The polyhedron defined by* LP_{cs} *is contained in* LP_{cb}.

Proof Due to the correspondence (9), equations (2) can be written in terms of the y-variables and therefore also hold for any feasible solution of LP_{cs}. Correspondingly, equations (3) are always fulfilled for any solution of LP_{cs}. If one is interested in a specific solution in terms of the x-variables for a feasible solution expressed by y-variables, it can be easily derived by considering each $t \in T$ and assigning values to variables x_i with $i \in \{1, \ldots, m \mid t_i = t\}$ in an iterative, greedy fashion so that relations (8) are fulfilled for any k_i^1 and k_i^2. A feasible assignment of such values must always exist as an individual x_i variable exists for each possible pair of positions Q_t^1 in s^1 and positions Q_t^2 in s^2, due to constraints (7), and the variable domains.

From the above results, we can directly conclude the following.

Theorem 1 LP_{cb} *corresponds to* LP_{cs} *when projected into the domain of* y*-variables, and therefore* ILP_{cb} *and* ILP_{cs} *yield the same LP-values and are equally strong.*

4 Experimental Evaluation

Both ILP_{cb} and ILP_{cs} were implemented using GCC 4.7.3 and IBM ILOG CPLEX V12.1. The experimental results were obtained on a cluster of PCs with 2933 MHz Intel(R) Xeon(R) 5670 CPUs having 12 nuclei and 32GB RAM. Moreover, CPLEX was configured for single-threaded execution.

4.1 Benchmark Instances

Two different benchmark sets were used for the experimental evaluation. The first one was introduced by Ferdous and Sohel Rahman in [7] for the evaluation of their ant colony optimization approach. This set contains in total 30 artificial instances and 15 real-life instances consisting of DNA sequences, that is, $|\Sigma| = 4$. Remember, in this context, that each problem instance consists of two related input strings. Moreover, the benchmark set consists of four subsets of instances. The first subset (henceforth labelled GROUP1) consists of 10 artificial instances in which the input strings have lengths up to 200. The second

Table 1 Results for the 10 instances of GROUP1.

id	n	ILP_{cb}					ILP_{cs}				
		value	**time (s)**	**gap**	**LP gap**	**# vars**	**value**	**time (s)**	**gap**	**LP gap**	**# vars**
1	114	*41	0/1	0.0%	3.3%	4299	*41	0/0	0.0%	3.3%	781
2	137	*47	1/2	0.0%	3.6%	6211	*47	0/0	0.0%	3.6%	928
3	158	*52	2/34	0.0%	5.7%	8439	*52	0/14	0.0%	5.7%	1172
4	113	*41	0/1	0.0%	2.0%	4299	*41	0/1	0.0%	2.0%	736
5	119	*40	1/1	0.0%	3.0%	4718	*40	0/1	0.0%	3.0%	833
6	115	*40	0/3	0.0%	4.2%	4435	*40	0/1	0.0%	4.2%	765
7	162	*55	2/38	0.0%	2.9%	8687	*55	0/18	0.0%	2.9%	1159
8	123	*43	1/2	0.0%	3.2%	4995	*43	0/2	0.0%	3.2%	816
9	118	*42	1/2	0.0%	3.7%	4995	*42	0/1	0.0%	3.7%	767
10	170	*54	1/51	0.0%	3.7%	9699	*54	0/16	0.0%	3.7%	1254
avg.		45.5	1/14	0.0%	3.5%	6029.3	45.5	0/5	0.0%	3.5%	921.1

subset (GROUP2) consists of 10 artificial instances with input string lengths in $(200, 400]$. In the third subset (GROUP3) the input strings of the 10 artificial instances have lengths in $(400, 600]$. Finally, the fourth subset (REAL) consists of 15 real-life instances of various lengths in $[200, 600]$. The second benchmark set that we used is new. It consists of 10 uniformly randomly generated instances for each combination of $n \in \{100, 200, \ldots, 1000\}$ and alphabet size $|\Sigma| \in \{4, 12, 20\}$. In total, this set thus consists of 300 benchmark instances.

4.2 Results for the instances from Ferdous and Sohel Rahman

The results for the four subsets of instances from the benchmark set by Ferdous and Sohel Rahman [7] are shown in Tables 1-4, in terms of one table per instance subset. The structure of these tables is as follows. The first and second columns provide the instance identifiers and the input string length, respectively. Then the results of ILP_{cb} and ILP_{cs} are shown by means of five columns each. The first column provides the objective values of the best solutions found within a limit of 3600 CPU seconds. In case optimality of the corresponding solution was proven by CPLEX, the value is marked by an asterisk. The second column provides computation times in the form X/Y, where X is the time at which CPLEX was able to find the first valid integer solution, and Y the time at which CPLEX found the best (possibly optimal) solution within the 3600s limit. The third column shows optimality gaps, which are the relative differences in percent between the values of the best feasible solutions and the lower bounds at the times of stopping the runs. The fourth column provides LP gaps, i.e., the relative differences between the LP relaxation values and the best (possibly optimal) integer solution values.[2] Finally, the last column lists the numbers of variables of the ILP models. The best result for each problem instance is marked by a grey background, and the last row of each table provides averages over the whole table.

The following observations can be made. First, apart from the instances of GROUP1 which are all solved with both models to optimality, the results for

[2] Note that we confirmed, in this context, that in all cases the values of the LP relaxations concerning ILP_{cb} and ILP_{cs} were equal.

Table 2 Results for the 10 instances of GROUP2.

id	n	ILP_{cb}					ILP_{cs}				
		value	**time (s)**	**gap**	**LP gap**	**# vars**	**value**	**time (s)**	**gap**	**LP gap**	**# vars**
1	337	98	50/2067	2.9%	4.1%	37743	98	1/1218	2.2%	4.1%	2740
2	376	106	80/1046	7.5%	7.8%	47174	103	1/2554	3.6%	5.2%	3191
3	334	97	35/1220	2.7%	3.7%	36979	*96	1/523	0.0%	2.7%	2776
4	351	102	48/891	4.9%	5.6%	40960	100	1/470	2.2%	3.7%	2914
5	398	116	83/2703	6.7%	7.5%	52697	114	1/903	4.5%	5.9%	3291
6	327	93	39/1476	5.6%	6.5%	35650	94	1/269	6.2%	7.5%	2694
7	303	88	31/3107	6.0%	7.7%	30839	87	1/1358	4.2%	6.7%	2494
8	358	104	61/3248	5.1%	6.2%	42668	104	1/72	5.7%	6.2%	2954
9	360	104	49/1563	5.2%	6.1%	42998	103	1/162	4.2%	5.2%	2924
10	306	89	27/1397	3.6%	4.9%	31169	*88	1/434	0.0%	3.8%	2423
avg.		99.7	50/1872	5.0%	6.0%	39887.7	98.7	1/796	3.3%	5.1%	2840.1

Table 3 Results for the 10 instances of GROUP3.

id	n	ILP_{cb}					ILP_{cs}				
		value	**time (s)**	**gap**	**LP gap**	**# vars**	**value**	**time (s)**	**gap**	**LP gap**	**# vars**
1	577	155	333/858	7.5%	7.7%	110973	154	2/1015	6.4%	6.5%	5230
2	556	155	345/693	7.7%	7.7%	102670	152	2/972	5.3%	5.9%	4849
3	599	166	462/2063	8.5%	8.6%	119287	160	2/643	4.8%	5.2%	5339
4	588	159	458/976	6.9%	7.1%	114975	159	2/1783	6.4%	7.1%	5251
5	547	150	279/682	9.7%	9.9%	99775	147	3/237	7.6%	8.1%	4917
6	517	147	239/573	9.1%	9.2%	88839	143	2/621	6.0%	6.7%	4441
7	535	149	253/620	9.8%	10.0%	95765	145	2/1572	6.7%	7.5%	4734
8	542	151	312/3591	6.7%	6.9%	97400	149	1/1092	5.0%	5.7%	4691
9	559	158	352/1022	10.9%	11.1%	104186	148	2/3418	4.2%	5.1%	5009
10	543	148	343/1334	9.1%	9.5%	98237	145	2/3316	6.7%	8.2%	4823
avg.		153.8	338/1241	8.6%	8.8%	103211.0	150.2	2/1467	5.9%	6.6%	4928.4

Table 4 Results for the 15 instances of set REAL.

id	n	ILP_{cb}					ILP_{cs}				
		value	**time (s)**	**gap**	**LP gap**	**# vars**	**value**	**time (s)**	**gap**	**LP gap**	**# vars**
1	252	*78	14/968	0.0%	3.9%	22799	*78	0/232	0.0%	3.9%	1966
2	487	139	196/441	9.2%	9.3%	80523	134	1/988	5.2%	5.9%	4330
3	363	104	61/3575	5.6%	6.4%	45869	102	1/115	3.9%	4.6%	3052
4	513	144	301/1353	6.5%	6.6%	91663	141	1/227	4.3%	4.7%	4467
5	559	150	379/1998	7.9%	8.2%	108866	148	2/3230	6.2%	7.0%	5068
6	451	128	170/3584	6.5%	7.0%	70655	124	1/1392	3.0%	4.0%	3836
7	458	121	180/1814	6.9%	7.6%	73502	119	1/2729	4.3%	6.1%	4187
8	433	116	127/3268	6.8%	7.6%	65560	115	1/607	5.5%	6.8%	3879
9	468	131	191/358	8.8%	8.9%	75833	127	1/844	5.2%	6.1%	4130
10	450	130	144/3429	6.1%	6.7%	69560	127	1/2669	3.1%	4.5%	3876
11	400	110	114/3591	4.8%	5.6%	56160	109	1/2309	3.3%	4.8%	3546
12	449	126	178/651	9.8%	10.2%	70861	122	1/562	6.3%	7.2%	3981
13	579	157	469/2236	7.1%	7.9%	115810	155	2/835	6.1%	6.7%	5251
14	458	130	161/3099	6.7%	7.2%	73449	129	1/581	5.5%	6.5%	3905
15	510	139	295/1430	7.7%	7.9%	91060	135	2/712	4.4%	5.2%	4556
avg.		126.9	212/2120	6.7%	7.4%	74163.9	124.3	1/1202	4.4%	5.6%	4002.0

subsets GROUP2, GROUP3 and REAL are clearly in favor of model ILP_{cs}. Only in one out of 35 cases (leaving GROUP1 aside) a better result is obtained with ILP_{cb}, and in further four cases the results obtained with ILP_{cs} are matched. In all remaining cases the solutions obtained with ILP_{cs} are better than those obtained with ILP_{cb}. This observation is confirmed by a study of the optimality gaps. They are significantly smaller for ILP_{cs} than for ILP_{cb}. One of the main reasons for the superiority of model ILP_{cs} over ILP_{cb} is certainly the difference in the number of the variables. For the instance of GROUP1, ILP_{cb} needs, on average, about 6.5 times more variables than ILP_{cs}. This factor seems to grow

with growing instance size. Concerning instances of GROUP2, ILP_{cb} requires, on average, about 14.0 times more variables. The corresponding number for GROUP3 is about 20.9. Another reason for the advantage of ILP_{cs} over ILP_{cb} is that symmetries are avoided. Finally, a last observation concerns the computation times: the first feasible integer solution is found for ILP_{cs}, on average, in about 0.7% of the time that is needed in the case of ILP_{cb}.

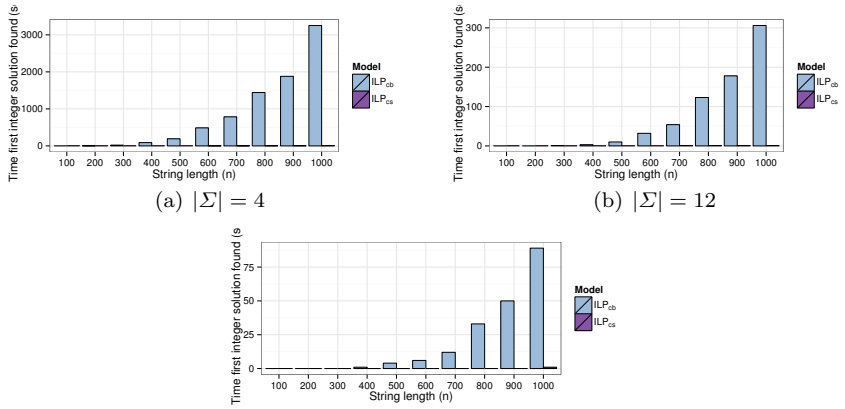
4.3 Results for the New Instance Set

The results for the new set of problem instances are presented in Table 5. Each line provides the results of both ILP_{cb} and ILP_{cs} averaged over the 10 instances for a combination between n and $|\Sigma|$. The results are presented for each ILP model by means of six table columns. The first five represent the same information as was provided in the case of the first benchmark set. An additional sixth column (with heading **# opt**) indicates for each row how many (out of 10) instances were solved to optimality. The additional last table column (with heading **Impr. in** %) indicates the average improvement in solution quality of ILP_{cs} over ILP_{cb}. The results permit, basically, to draw the same conclusions as in the case of the results for the instance set treated in the previous subsection. The application of CPLEX to ILP_{cs} outperforms the application of CPLEX to ILP_{cb} both in final solution quality and in the computation time needed to find the first feasible integer solution. These differences in results become more pronounced with increasing input string length and with decreasing alphabet size. In the case of $|\Sigma| = 4$, for example, the solutions provided by ILP_{cs} are on average 5.0% better than those provided by ILP_{cb}. The superiority of ILP_{cs} over ILP_{cb} is also indicated by the number of instances that were solved to optimality: 160 out of 300 in the case of ILP_{cb}, and 183 out of 300 in the case of ILP_{cs}.

In order to facilitate the study of the computation times at which the first integer solutions were found, these times are graphically shown for different values of $|\Sigma|$ in three different barplots in Figure 1. The charts clearly show that the advantages of ILP_{cs} over ILP_{cb} are considerable. In fact, the numbers concerning ILP_{cs} are so small (in comparison to the ones concerning ILP_{cb}) that the bars are not visible in these plots. Moreover, these advantages seem to grow with increasing alphabet size. This means that, even though the differences in solution quality are negligible when $|\Sigma| = 20$, the first integer solutions are found much faster in the case of ILP_{cs}. The average gap sizes concerning the quality of the best solutions found and the best lower bounds at the time of termination are plotted in the same way in the three charts of Figure 2. These charts clearly show that, for all combinations of n and $|\Sigma|$, the average gap is smaller in the case of ILP_{cs}. Finally, Figure 3 shows the evolution of the number of variables needed by the two models for instances of different sizes.

Table 5 Average results for the 300 instances of the newly generated benchmark set.

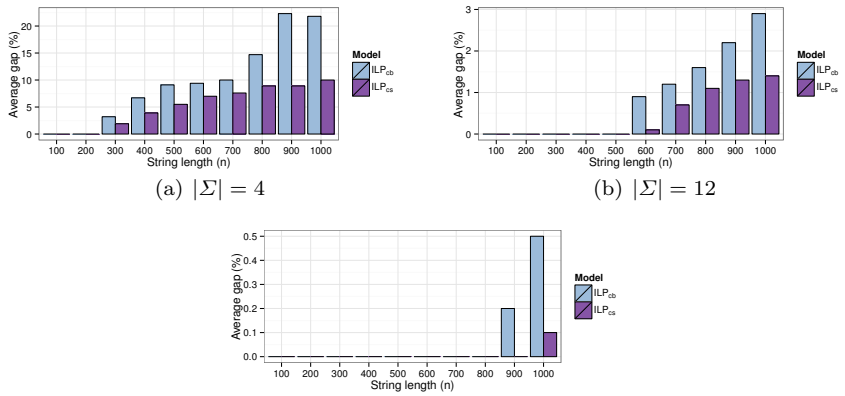
n	$\lvert\Sigma\rvert$	ILP$_{\text{cb}}$						ILP$_{\text{cs}}$						**Impr.**
		value	**time (s)**	**# opt**	**gap**	**LP gap**	**# vars**	**value**	**time (s)**	**# opt**	**gap**	**LP gap**	**# vars**	**in %**
100	4	37.3	0/0	10/10	0.0%	2.8%	3425.6	37.3	0/0	10/10	0.0%	2.8%	649.7	0.0%
	12	68.5	0/0	10/10	0.0%	0.2%	993.3	68.5	0/0	10/10	0.0%	0.2%	324.0	0.0%
	20	79.8	0/0	10/10	0.0%	0.0%	622.4	79.8	0/0	10/10	0.0%	0.0%	264.2	0.0%
200	4	63.5	3/101	10/10	0.0%	3.5%	13498.5	63.5	0/34	10/10	0.0%	3.5%	1473.8	0.0%
	12	119.2	0/0	10/10	0.0%	0.5%	3824.6	119.2	0/0	10/10	0.0%	0.5%	762.8	0.0%
	20	146.2	0/0	10/10	0.0%	0.0%	2301.1	146.2	0/0	10/10	0.0%	0.0%	591.6	0.0%
300	4	88.5	21/2358	1/10	3.2%	4.7%	30398.5	88.1	0/448	4/10	1.9%	4.3%	2412.5	0.5%
	12	165.3	1/3	10/10	0.0%	0.8%	8478.6	165.3	0/1	10/10	0.0%	0.8%	1249.1	0.0%
	20	206.7	0/0	10/10	0.0%	0.02%	5029.6	206.7	0/0	10/10	0.0%	0.02%	967.0	0.0%
400	4	115.5	89/2159	0/10	6.7%	7.2%	53658.5	113.0	0/1277	0/10	3.9%	5.2%	3369.8	2.2%
	12	208.9	3/47	10/10	0.0%	0.9%	14887.2	208.9	0/3	10/10	0.0%	0.9%	1742.1	0.0%
	20	261.5	1/1	10/10	0.0%	0.1%	8932.0	261.5	0/0	10/10	0.0%	0.1%	1366.8	0.0%
500	4	139.3	192/870	0/10	9.1%	9.3%	84004.2	134.7	0/793	0/10	5.5%	6.2%	4411.8	3.4%
	12	249.0	10/328	10/10	0.0%	0.9%	23173.1	249.0	0/26	10/10	0.0%	0.9%	2266.2	0.0%
	20	312.2	4/4	10/10	0.0%	0.2%	13761.0	312.2	0/0	10/10	0.0%	0.2%	1803.3	0.0%
600	4	162.2	487/1893	0/10	9.4%	9.5%	120795.1	159.0	2/2043	0/10	7.0%	7.9%	5451.3	2.0%
	12	291.0	32/1202	2/10	0.9%	1.2%	33372.6	290.5	0/151	9/10	0.1%	1.1%	2780.3	0.2%
	20	362.3	6/12	10/10	0.0%	0.3%	19543.8	362.3	0/1	10/10	0.0%	0.3%	2253.2	0.0%
700	4	187.7	785/2856	0/10	10.0%	10.2%	164116.2	183.4	3/1680	0/10	7.6%	7.8%	6459.3	2.3%
	12	331.0	54/1811	0/10	1.2%	1.4%	45303.9	330.2	1/962	2/10	0.7%	1.2%	3312.0	0.2%
	20	408.9	12/120	10/10	0.0%	0.4%	26588.5	408.9	0/4	10/10	0.0%	0.4%	2729.3	0.0%
800	4	221.6	1442/3432	0/10	14.7%	15.3%	213956.1	207.1	5/2052	0/10	8.9%	9.4%	7555.9	7.0%
	12	368.7	123/2460	0/10	1.6%	1.8%	59026.8	367.6	1/943	0/10	1.1%	1.5%	3871.0	0.3%
	20	456.1	33/669	10/10	0.0%	0.5%	34451.6	456.1	0/14	10/10	0.0%	0.5%	3180.1	0.0%
900	4	266.3	1880/2314	0/10	22.3%	22.5%	271158.3	227.3	6/2607	0/10	8.9%	9.4%	8682.5	17.2%
	12	408.5	178/2406	0/10	2.2%	2.3%	74372.5	405.5	1/1350	0/10	1.3%	1.5%	4440.8	0.7%
	20	501.5	50/1625	6/10	0.2%	0.6%	43543.4	501.3	0/238	10/10	0.0%	0.6%	3649.8	0.04%
1000	4	288.7	3253/3739	0/10	21.8%	22.1%	334125.1	250.5	9/1465	0/10	10.0%	10.0%	9825.4	15.2%
	12	449.2	306/3147	0/10	2.9%	2.9%	91955.2	443.2	1/1324	0/10	1.4%	1.7%	5017.2	1.4%
	20	546.9	89/2182	1/10	0.5%	0.7%	53736.0	546.1	1/844	8/10	0.1%	0.6%	4106.7	0.1%



(a) $|\Sigma| = 4$ (b) $|\Sigma| = 4$

(c) $|\Sigma| = 20$

Fig. 1 Evolution of the average computation time the first integer solution is found.



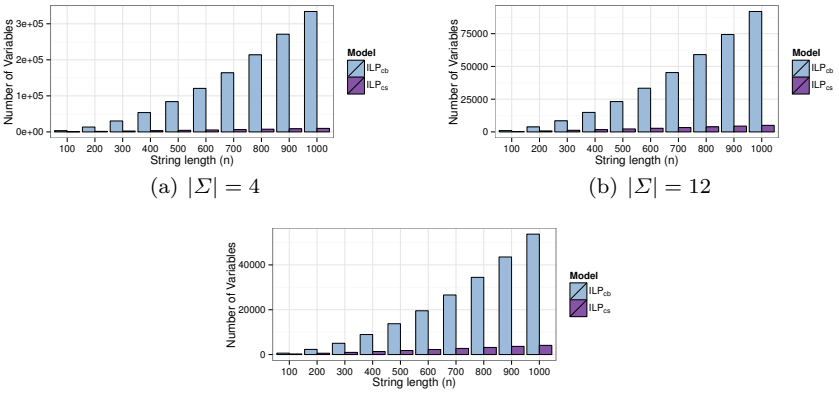
(a) $|\Sigma| = 4$ (b) $|\Sigma| = 12$

(c) $|\Sigma| = 20$

Fig. 2 Evolution of the average optimality gap size (in percent).

5 Conclusions and Future Work

While (meta-)heuristic approaches are the state-of-the-art for approximately solving large instances of the MCSP, instances with string lengths of less than about 1000 letters can be well solved with an ILP model in conjunction with a state-of-the-art solver like CPLEX. In this work we have proposed the model based on *common substrings* that reduces symmetries appearing in the formerly suggested *common blocks* model. While our polyhedral analysis indicated that both models are equally strong w.r.t. their linear programming relaxations, there are significant differences in the computational difficulties to solve these models. The new formulation allows for finding feasible solutions of already reasonable quality in substantially less time and also yields better final solutions in most cases where proven optimal solutions could not



(a) $|\Sigma| = 4$ (b) $|\Sigma| = 12$

(c) $|\Sigma| = 20$

Fig. 3 Evolution of the number of variables used by the two ILP models.

be identified within the time limit. An important reason for this is to be found in the number of variables needed by the two models. While the existing model from the literature requires $O(n^3)$ variables (where n is the length of the input strings), the new model only requires $O(n^2)$ variables.

In future work it would be interesting to consider extended variants of the MCSP, in particular such where the input strings need not to be related. In biological applications this would give a greater flexibility as sequences that were also affected by other kinds of mutations can be compared in terms of their reordering of subsequences. Another interesting generalization would be to consider more than two input strings. The newly proposed ILP model appears to be a promising basis also for these variants.

Acknowledgements C. Blum acknowledges support by grant TIN2012-37930-02 of the Spanish Government. In addition, support is acknowledged from IKERBASQUE (Basque Foundation for Science). Our experiments have been executed in the High Performance Computing environment managed by RDlab (`http://rdlab.lsi.upc.edu`) and we would like to thank them for their support.

References

1. Blum, C., Lozano, J.A., Pinacho Davidson, P.: Iterative probabilistic tree search for the minimum common string partition problem. In: M.J. Blesa, C. Blum, S. Voss (eds.) Proceedings of HM 20104– 9th International Workshop on Hybrid Metaheuristics, *Lecture Notes in Computer Science*, vol. 8457, pp. 154–154. Springer Verlag, Berlin, Germany (2014)
2. Blum, C., Lozano, J.A., Pinacho Davidson, P.: Mathematical programming strategies for solving the minimum common string partition problem. European Journal of Operational Research **242**(3), 769–777 (2015)
3. Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S., Jiang, T.: Assignment of orthologous genes via genome rearrangement. IEEE/ACM Transations on Computional Biology and Bioinformatics **2**(4), 302–315 (2005)

4. Chrobak, M., Kolman, P., Sgall, J.: The greedy algorithm for the minimum common string partition problem. In: K. Jansen, S. Khanna, J.D.P. Rolim, D. Ron (eds.) Proceedings of APPROX 2004 – 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, *Lecture Notes in Computer Science*, vol. 3122, pp. 84–95. Springer Berlin Heidelberg (2004)
5. Cormode, G., Muthukrishnan, S.: The string edit distance matching problem with moves. ACM Transactions on Algorithms **3**(2), 1–19 (2007)
6. Damaschke, P.: Minimum common string partition parameterized. In: K.A. Crandall, J. Lagergren (eds.) Proceedings of WABI 2008 – 8th International Workshop on Algorithms in Bioinformatics, *Lecture Notes in Computer Science*, vol. 5251, pp. 87–98. Springer Berlin Heidelberg (2008)
7. Ferdous, S.M., Rahman, M.S.: Solving the minimum common string partition problem with the help of ants. In: Y. Tan, Y. Shi, H. Mo (eds.) Proceedings of ICSI 2013 – 4th International Conference on Advances in Swarm Intelligence, *Lecture Notes in Computer Science*, vol. 7928, pp. 306–313. Springer Berlin Heidelberg (2013)
8. Ferdous, S.M., Rahman, M.S.: A MAX-MIN ant colony system for minimum common string partition problem. CoRR **abs/1401.4539** (2014). `http://arxiv.org/abs/1401.4539`
9. Fu, B., Jiang, H., Yang, B., Zhu, B.: Exponential and polynomial time algorithms for the minimum common string partition problem. In: W. Wang, X. Zhu, D.Z. Du (eds.) Proceedings of COCOA 2011 – 5th International Conference on Combinatorial Optimization and Applications, *Lecture Notes in Computer Science*, vol. 6831, pp. 299–310. Springer Berlin Heidelberg (2011)
10. Gallardo, J.E.: A multilevel probabilistic beam search algorithm for the shortest common supersequence problem. PLOS ONE **7**(12) (2012)
11. Garey, M.R., Johnson, D.S.: Computers and intractability; a guide to the theory of *NP*-completeness. W. H. Freeman (1979)
12. Goldstein, A., Kolman, P., Zheng, J.: Minimum common string partition problem: Hardness and approximations. In: R. Fleischer, G. Trippen (eds.) Proceedings of ISAAC 2004 – 15th International Symposium on Algorithms and Computation, *Lecture Notes in Computer Science*, vol. 3341, pp. 484–495. Springer Berlin Heidelberg (2005)
13. Goldstein, I., Lewenstein, M.: Quick greedy computation for minimum common string partitions. In: R. Giancarlo, G. Manzini (eds.) Proceedings of CPM 2011 – 22nd Annual Symposium on Combinatorial Pattern Matching, *Lecture Notes in Computer Science*, vol. 6661, pp. 273–284. Springer Berlin Heidelberg (2011)
14. He, D.: A novel greedy algorithm for the minimum common string partition problem. In: I. Mandoiu, A. Zelikovsky (eds.) Proceedings of ISBRA 2007 – Third International Symposium on Bioinformatics Research and Applications, *Lecture Notes in Computer Science*, vol. 4463, pp. 441–452. Springer Berlin Heidelberg (2007)
15. Hsu, W.J., Du, M.W.: Computing a longest common subsequence for a set of strings. BIT Numerical Mathematics **24**(1), 45–59 (1984). DOI 10.1007/BF01934514
16. Jiang, H., Zhu, B., Zhu, D., Zhu, H.: Minimum common string partition revisited. Journal of Combinatorial Optimization **23**(4), 519–527 (2012)
17. Kaplan, H., Shafrir, N.: The greedy algorithm for edit distance with moves. Information Processing Letters **97**(1), 23–27 (2006)
18. Kolman, P.: Approximating reversal distance for strings with bounded number of duplicates. In: J. Jedrzejowicz, A. Szepietowski (eds.) Proceedings of MFCS 2005 – 30th International Symposium on Mathematical Foundations of Computer Science, *Lecture Notes in Computer Science*, vol. 3618, pp. 580–590. Springer Berlin Heidelberg (2005)
19. Kolman, P., Waleń, T.: Reversal distance for strings with duplicates: Linear time approximation using hitting set. In: T. Erlebach, C. Kaklamanis (eds.) Proceedings of WAOA 2007 – 4th International Workshop on Approximation and Online Algorithms, *Lecture Notes in Computer Science*, vol. 4368, pp. 279–289. Springer Berlin Heidelberg (2007)
20. Meneses, C., Oliveira, C., Pardalos, P.: Optimization techniques for string selection and comparison problems in genomics. IEEE Engineering in Medicine and Biology Magazine **24**(3), 81–87 (2005)
21. Mousavi, S., Babaie, M., Montazerian, M.: An improved heuristic for the far from most strings problem. Journal of Heuristics **18**, 239–262 (2012)

22. Shapira, D., Storer, J.A.: Edit distance with move operations. In: A. Apostolico, M. Takeda (eds.) Proceedings of CPM 2002 – 13th Annual Symposium on Combinatorial Pattern Matching, *Lecture Notes in Computer Science*, vol. 2373, pp. 85–98. Springer Berlin Heidelberg (2002)
23. Smith, T., Waterman, M.: Identification of common molecular subsequences. Journal of Molecular Biology **147**(1), 195–197 (1981)