Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Mathematical programming strategies for solving the minimum common string partition problem

Christian Blum[a,b,*], José A. Lozano[a], Pinacho Davidson[a,c]

[a] Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, San Sebastian, Spain
[b] IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
[c] Escuela de Informática, Universidad Santo Tomás, Concepción, Chile

ARTICLE INFO

Article history:
Received 8 April 2014
Accepted 23 October 2014
Available online 4 November 2014

Keywords:
Minimum common string partition
Integer linear programming
Heuristic

ABSTRACT

The minimum common string partition problem is an NP-hard combinatorial optimization problem with applications in computational biology. In this work we propose the first integer linear programming model for solving this problem. Moreover, on the basis of the integer linear programming model we develop a deterministic 2-phase heuristic which is applicable to larger problem instances. The results show that provenly optimal solutions can be obtained for problem instances of small and medium size from the literature by solving the proposed integer linear programming model with CPLEX. Furthermore, new best-known solutions are obtained for all considered problem instances from the literature. Concerning the heuristic, we were able to show that it outperforms heuristic competitors from the related literature.

1. Introduction

Optimization problems related to strings—such as protein or DNA sequences—are very common in bioinformatics. Examples include string selection problems (Meneses, Oliveira, & Pardalos, 2005; Mousavi, Babaie, & Montazerian, 2012; Pappalardo, Pardalos, & Stracquadanio, 2013), the longest common subsequence problem and its variants (Hsu & Du, 1984; Smith & Waterman, 1981), alignment problems (Gusfield, 1997; Rajasekaran, Nick, Pardalos, Sahni, & Shaw, 2001), and similarity search (Rajasekaran, Hu, Luo, Nick, Pardalos, Sahni, & Shaw, 2001). These problems are often computationally very hard, if not even *NP*-hard (Garey & Johnson, 1979). In this work we deal with the *minimum common string partition* (MCSP) problem, which can be described as follows. We are given two related input strings that have to be partitioned each into the same collection of substrings. The size of the collection is subject to minimization. A formal description of the problem will be provided in Section 1.1. The MCSP problem has applications, for example, in the bioinformatics field. Chen, Zheng, Fu, Nan, Zhong, Lonardi, and Jiang (2005) point out that the MCSP problem is closely related to the problem of sorting by reversals with duplicates, a key problem in genome rearrangement.

In this paper we introduce the first integer linear program (ILP) for solving the MCSP problem. An experimental evaluation on problem instances from the related literature shows that this ILP can be efficiently solved, for example, by using any version of IBM ILOG CPLEX. However, a study on new instances of larger size demonstrates the limitations of the model. Therefore, we additionally introduce a deterministic 2-phase heuristic which is strongly based on the original ILP. The experimental evaluation shows that the heuristic is applicable to larger problem instances than the original ILP. Moreover, it is shown that the heuristic outperforms competitor algorithms from the related literature on known problem instances.

1.1. Problem description

The MCSP problem can technically be described as follows. Given are two input strings s_1 and s_2, both of length n over a finite alphabet Σ. These two strings are required to be *related*, which means that each letter appears the same number of times in each of them. Note that this definition implies that s_1 and s_2 have the same length. A valid solution to the MCSP problem is obtained by partitioning s_1 into a set P_1 of non-overlapping substrings, and s_2 into a set P_2 of non-overlapping substrings, such that $P_1 = P_2$. Moreover, we are interested in finding a valid solution such that $|P_1| = |P_2|$ is minimal.

Consider the following example. Given are DNA sequences $s_1 =$ **AGACTG** and $s_2 =$ **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 1, that is, $P_1 = P_2 = \{\mathbf{A}, \mathbf{A}, \mathbf{C}, \mathbf{T}, \mathbf{G}, \mathbf{G}\}$. The

[*] Corresponding author at: Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, San Sebastian, Spain. Tel.: +34 943017119.

E-mail addresses: christian.blum@ehu.es (C. Blum), ja.lozano@ehu.es (J. A. Lozano), ppinacho@santotomas.cl (P. Davidson).

http://dx.doi.org/10.1016/j.ejor.2014.10.049
0377-2217/© 2014 Elsevier B.V. All rights reserved.

objective function value of this solution is 6. However, the optimal solution, with objective function value 3, is $P_1 = P_2 = \{\mathbf{ACT}, \mathbf{AG}, \mathbf{G}\}$.

1.2. Related work

The MCSP problem has been introduced by Chen et al. (2005) due to its relation to genome rearrangement. More specifically, it has applications in biological questions such as: May a given DNA string possibly be obtained by rearrangements of another DNA string? The general problem has been shown to be *NP*-hard even in very restrictive cases (Goldstein, Kolman, & Zheng, 2005). Other papers concerning problem hardness consider, for example, the *k*-MCSP problem, which is the version of the 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 Goldstein et al. (2005). When the input strings are over an alphabet of size *c*, the corresponding problem is denoted as MCSP^c. Jiang et al. proved that the decision version of the MCSP^c problem is *NP*-complete when $c \geq 2$ (Jiang, Zhu, Zhu, & Zhu, 2012).

The MCSP has been considered quite extensively by researchers dealing with the approximability of the problem. Cormode and Muthukrishnan (2007), for example, proposed an $O(lognlog^*n)$-approximation for the *edit distance with moves* problem, which is a more general case of the MCSP problem. Shapira and Storer (2002) extended on this result. Other approximation approaches for the MCSP problem have been proposed in Kolman and Waleń (2007). In this context, Chrobak, Kolman, and Sgall (2004) 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 $\Omega(\log(n))$. In the case of the general MCSP problem, the approximation ratio is between $\Omega(n^{0.43})$ and $O(n^{0.67})$, assuming that the input strings use an alphabet of size $O(\log(n))$. Kaplan and Shafrir (2006) raised 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 (Kolman, 2005). Recently, Goldstein and Lewenstein proposed a greedy algorithm for the MCSP problem that runs in $O(n)$ time (see Goldstein & Lewenstein, 2011). He (2007) introduced a greedy algorithm with the aim of obtaining better average results.

Damaschke (2008) was the first one to study the fixed-parameter tractability (FPT) of the problem. Later, Jiang et al. (2012) showed that both the *k*-MCSP and MCSP^c problems admit FPT algorithms when *k* and *c* are constant parameters. Finally, Fu, Jiang, Yang, and Zhu (2011) and Ding and Fu (2013) proposed an $O(2^{nn^{O(1)}})$ time algorithm for the general case and an $O(n(logn)^2)$ time algorithm applicable under some constraints.

To our knowledge, the only metaheuristic approaches that have been proposed in the related literature for the MCSP problem are (1) the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System by Ferdous and Sohel Rahman (2014, 2013) and (2) the probabilistic tree search algorithm by Blum, Lozano, and Pinacho Davidson (2014). Both works applied their algorithm to a range of artificial and real DNA instances from Ferdous and Sohel Rahman (2013).

1.3. Organization of the paper

The remaining part of the paper is organized as follows. In Section 2, the ILP model for solving the MCSP is outlined. Moreover, an experimental evaluation is provided. The deterministic heuristic, together with an experimental evaluation, is described in Section 3. Finally, in Section 4 we provide conclusions and an outlook to future work.

2. An integer linear program to solve the MCSP

In the following we present the first ILP model for solving the MCSP. For this, the definitions provided in the following are required. Note that an illustrative example is provided in Section 2.3.

2.1. Preliminaries

Henceforth, a *common block* b_i of input strings s_1 and s_2 is denoted as a triple $(t_i, k1_i, k2_i)$ where t_i is a string which can be found starting at position $1 \leq k1_i \leq n$ in string s_1 and starting at position $1 \leq k2_i \leq n$ in string s_2. Moreover, let $B = \{b_1, \ldots, b_m\}$ be the (ordered) set of all possible common blocks of s_1 and s_2.[1] Given the definition of B, any valid solution $\mathcal{S}$ to the MCSP problem is a subset of B—that is, $\mathcal{S} \subset B$—such that:

1. $\sum_{b_i \in \mathcal{S}} |t_i| = n$, that is, the sum of the length of the strings corresponding to 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 overlap neither in s_1 nor in s_2.

Moreover, any (valid) partial solution $\mathcal{S}_{\text{partial}}$ is a subset of B fulfilling the following conditions: (1) $\sum_{b_i \in \mathcal{S}_{\text{partial}}} |t_i| < n$ and (2) for any two common blocks $b_i, b_j \in \mathcal{S}_{\text{partial}}$ it holds that their corresponding strings overlap neither in s_1 nor in s_2. Note that any valid partial solution can be extended to be a valid solution. Furthermore, given a partial solution $\mathcal{S}_{\text{partial}}$, set $B(\mathcal{S}_{\text{partial}}) \subset B$ denotes the set of common blocks that may be used in order to extend $\mathcal{S}_{\text{partial}}$ such that the result is again a valid (partial) solution.

2.2. The integer linear program

First, two binary $m \times n$ matrices $M1$ and $M2$ are defined as follows. In both matrices, row $1 \leq i \leq m$ corresponds to common block $b_i \in B$. Moreover, a column $1 \leq j \leq n$ corresponds to position j in input string s_1, respectively s_2. In general, the entries of matrix $M1$ are set to zero. However, in each row i, the positions that string t_i (of common block b_i) occupies in input string s_1 are set to one. Correspondingly, the entries of matrix $M2$ are set to zero, apart from the fact that in each row i the positions occupied by string t_i in input string s_2 are set to one. Henceforth, the position (i, j) of a matrix M is denoted by $M_{i,j}$. Finally, we introduce for each common block $b_i \in B$ a binary variable x_i. With these definitions we can express the MCSP in form of the following integer linear program, henceforth referred to by ILP_{orig}.

$$\min \sum_{i=1}^{m} x_i \tag{1}$$

subject to :

$$\sum_{i=1}^{m} M1_{i,j} \cdot x_i = 1 \quad \text{for } j = 1, \ldots, n \tag{2}$$

$$\sum_{i=1}^{m} M2_{i,j} \cdot x_i = 1 \quad \text{for } j = 1, \ldots, n \tag{3}$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, \ldots, m$$

Hereby, the objective function minimizes the number of selected common blocks. Constraints (2) make sure that the strings corresponding to the selected common blocks do not overlap in input string s_1, while constraints (3) make sure that the strings corresponding to the selected common blocks do not overlap in input string s_2. Moreover, note that constraints (2) and (3) implicitly ensure that the sum of the length of the strings corresponding to the selected common blocks is equal to n.

2.3. Example

As an example, consider the small problem instance from Section 1.1. The complete set of common blocks (B) as induced by

[1] The way in which B is ordered is of no importance.