Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 66 (2018) 15–22

www.elsevier.com/locate/endm

ILP-Based Reduced Variable Neighborhood Search for Large-Scale Minimum Common String Partition [⋆]

Christian Blum [1]

Artificial Intelligence Research Institute (IIIA-CSIC)
Campus of the UAB, Bellaterra, Spain

Abstract

The minimum common string partition problem is a challenging NP-hard optimization problem from the bioinformatics field. In this work we, first, present a modification which allows to apply the current state-of-the-art technique from the literature to much larger problem instances. Second, also based on the introduced modification, we develop a reduced variable neighborhood search algorithm for tackling large-scale problem instances. The skaking step of this algorithm destroys the incumbent solution partially, in a randomized way, and generates a complete solution on the basis of the partial solution by means of integer linear programming techniques. The proposed algorithm is compared to the state-of-the-art technique from the literature. The results show that the proposed algorithm consistently outperforms the state-of-the-art algorithm in the context of problem instances based on large alphabet sizes.

Keywords: Minimum common string partition, reduced variable neighborhood search, integer linear programming

1 Introduction

The minimum common string partition (MCSP) problem is a string-based combinatorial optimization problem arising in bioinformatics. In this problem

[⋆] This work was funded by proyect TIN2015-66863-C2-1-R (Spanish Ministry for Economy and Competitiveness, FEDER funds from the European Union).

[1] Email: `christian.blum@iiia.csic.es`

https://doi.org/10.1016/j.endm.2018.03.003
1571-0653/© 2018 Elsevier B.V. All rights reserved.

we are given two input strings s_1 and s_2, both of length n over a finite alphabet Σ. The two input strings fulfill the property of being *related*, which means that each letter of Σ has the same number of occurrences in each string. Therefore, 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. As an example, consider the following two DNA sequences: $s_1 = \mathbf{AGACTG}$ and $s_2 = \mathbf{ACTAGG}$. Counting the different letters in both strings, it is easy to confirm that the two strings are related. Given such a problem instance, a trivial valid solution can always 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 objective function value of this solution is 6. The optimal solution in this example, with objective function value 3, is $P_1 = P_2 = \{\mathbf{ACT}, \mathbf{AG}, \mathbf{G}\}$.

The MCSP problem, as pointed out by Chen et al. [5], is closely related to the problem of sorting by reversals with duplicates, which is one of the key problems in genome rearrangement. It has been shown to be NP-hard even in very restrictive cases [8]. The greedy algorithm by He [9] was the first practical optimization technique that was introduced. The first metaheuristics proposed in the related literature for the MCSP problem were (1) the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System by Ferdous and Sohel Rahman [6] and (2) the probabilistic tree search algorithm by Blum et al. [1]. Both works made use of a range of artificial and real DNA instances from [6] for the experimental evaluation. Later on, the focus shifted to integer linear programming (ILP) techniques. More specifically, the first ILP model was proposed in [2], together with a deterministic 2-phase heuristic based on the ILP model. Later on, improved ILP models were presented in [7,4]. The current state-of-the-art technique is a construct, merge, solve & adapt (CMSA) approach from [3]. This hybrid technique sequentially applies an ILP solver to a reduced sub-instance of the original problem instance.

1.1 Contribution of this Work

First, we present a modification of the greedy algorithm by He [9] and the ILP model from [2]. These two techniques are the principal components of the state-of-the-art CMSA algorithm from [3]. The afore-mentioned modification allows to apply CMSA to much larger problem instances: the largest problem instances studied in [3] consisted of input strings of length 2000, whereas here we apply our techniques to problem instances with strings of up to 20000 characters. Moreover, equally based on the introduced modification, we develop a reduced variable neighborhood search (RVNS) technique, which—as will be shown—has advantages over CMSA in the context of problem instances based on large alphabets.

2 Introduced Modification

Both the greedy algorithm from [9] (even though originally not described in this way) and the ILP model from [2] are based on the concept of *common blocks*. More specifically, a common block b_i of input strings s_1 and s_2 is a triple $(t_i, k1_i, k2_i)$ where t_i is a substring starting at position $1 \leq k1_i \leq n$ in s_1 and at position $1 \leq k2_i \leq n$ in s_2. Moreover, B is the set of all possible common blocks with respect to s_1 and s_2. Given the common block concept, any valid solution S to the MCSP problem is a subset of B—that is, $S \subset B$—such that (1) $\sum_{b_i \in \mathscr{S}} |t_i| = n$, that is, the sum of the length of the substrings corresponding to the common blocks in S is equal to the length of the input strings, and (2) for any two common blocks $b_i, b_j \in \mathscr{S}$ it holds that the substrings they represent neither overlap in s_1 nor in s_2.

Based on this definition, the greedy algorithm from [9] can be explained as follows. Given a valid partial solution S^p, let $N(S^p) \subset B$ denote the set of common blocks that can be added to S^p such that the result is again a valid (possibly still partial) solution. The greedy algorithm starts with $S^p := \emptyset$. At each iteration, one of the blocks $b_i \in N(S^p)$ is selected such that

$$b_i := \operatorname{argmax}\{|t_j| \mid b_j \in N(S^p)\} \ . \tag{1}$$

This block is then added to S^p, and this process is continued until S^p is a complete—in the sense of non-extensible—solution. In other words, the algorithm stops once $N(S^p)$ is empty.

One aspect that slows this algorithm down is that B contains exponentially many common blocks. In fact, most of these common blocks represent substrings of length one. A study from [2] has shown that, for randomly generated input strings and an alphabet of size four, about 75% of all common blocks of B refer to substrings of length one. Therefore, the modification considered in this paper considers the subset $B^{>1}$ of B that only contains the common blocks that represent substrings of length greater or equal to two. For the execution of the greedy algorithm, set B is replaced with set $B^{>1}$. However, when the algorithm stops due to $N(S^p)$ being empty, the resulting set S^p does generally not correspond to a complete solution. Nevertheless, we can be sure that the remaining (uncovered) parts of input strings s_1 and

s_2 must be covered by common blocks of length one. Therefore, a complete solution can be derived from S^p by finding any bijective mapping from the uncovered positions of s_1 to the uncovered positions of s_2, in such a way that positions that are mapped to each other have the same letters. This can be done in linear time by simply mapping the first uncovered occurrence of each letter $l \in \Sigma$ in s_1 to the first uncovered occurrence of that letter in s_2, the second occurrence of this letter in s_1 to the second one in s_2, and so on.

The same modification can be applied to the ILP model from [2]. The original ILP model is as follows. Two binary $m \times n$ matrices $M1$ and $M2$ are defined, in which row $1 \leq i \leq m$ corresponds to common block $b_i \in B$, that is, both matrices have $|B|$ rows. Moreover, a column $1 \leq j \leq n$ corresponds to position j in input string s_1, respectively s_2. In each row i of $M1$, the positions $k1_i \leq j \leq k1_i + |t_i| - 1$ are set to one, whereas the remaining positions are set to zero (correspondingly in the case of $M2$). In the following, the position (i, j) of a matrix M is denoted by $M_{i,j}$. The ILP model, which makes use of a binary variable x_i for each common block $b_i \in B$, can then be phrased as follows:

$$\mathbf{min} \sum_{b_i \in B} x_i \tag{2}$$

subject to:

$$\sum_{b_i \in B} M1_{i,j} \cdot x_i = 1 \quad \text{for } j = 1, \ldots, n \tag{3}$$

$$\sum_{b_i \in B} M2_{i,j} \cdot x_i = 1 \quad \text{for } j = 1, \ldots, n \tag{4}$$

$$x_i \in \{0, 1\} \quad \text{for } b_i \in B$$

Note that the objective function minimizes the number of selected common blocks. Constraints (3) and (4) make sure that the corresponding substrings do not overlap, neither in s_1 nor in s_2. Moreover, due to constraints (3) and (4), the sum of the length of the chosen substrings is n.

In the modified ILP model, we (1) replace B with $B^{>1}$, (2) exchange the objective function $\sum_{b_i \in B} x_i$ with $\left(\sum_{b_i \in B^{>1}} + (n - \sum_{b_i \in B^{>1}} x_i |t_i|)\right)$, and (3) exchange the equality signs in constraints (3) and (4) with a $\leq$-symbol. Note that the additional part of the objective function corresponds to the number of uncovered positions, given the set of selected common blocks.

With these two modifications we changed the implementation of the current state-of-the-art method CMSA from [3] and—in this way—made it applicable to larger problem instances, as shown in Section 4.

Algorithm 1 Reduced Variable Neighborhood Search (RVNS)

```
input: neighborhood functions $N_k$, $k = 1, \ldots, k_{\max}$
$S$ := GenerateInitialSolution(), $k := 1$
while termination conditions not satisfied do
    $S'$ := ChooseRandomNeighbor($N_k(S)$)
    if $|S'| < |S|$ then
        $S := S'$, $k := 1$
    else
        $k \leftarrow k + 1$
        if $k > k_{\max}$ then $k := 1$ end if
    end if
end while
```

3 ILP-Based Reduced Variable Neighborhood Search

Reduced variable neighborhood search (RVNS) is an algorithm originally proposed in [10], which is obtained from the standard variable neighborhood search (VNS) by removing the local search phase. In other words, the algorithm explores the pre-defined neighborhoods randomly by means of the shaking procedure. As mentioned in [10], RVNS is thought for the application to large-scale problem instances for which the local search step might be computationally too heavy. The pseudo-code of RVNS is provided in Algorithm 1.

Note that a solution S in this framework is (generally) an incomplete solution, being the result of working with the modified greedy and ILP techniques outlined in Section 2. However, the objective function values given in Section 4 are the values of the corresponding complete solutions obtained by the simple mapping procedure as outlined in Section 2. In the following, the components of Algorithm 1 are described in detail. First, the modified greedy algorithm described in Section 2 is applied in function GenerateInitialSolution(). Second, the following is done in function ChooseRandomNeighbor($N_k(S)$). A neighborhood function is defined via a solution destruction rate. More specifically, given a destruction rate $0 < d < 1$, a random neighbor of a solution S is obtained by randomly removing $\lfloor |S| \cdot d \rfloor$ common blocks from S and subsequently re-constructing the resulting partial solution S^p by applying the modified ILP model from Section 2, with the following additional set of constraints:

$$x_i = 1 \;\; \forall \, b_i \in S^p \tag{5}$$

A specific neighborhood $N_k()$ is defined via a minimum (resp. maximum) destruction rate $d_{\min}$ (resp. $d_{\max}$). The destruction rate d_k that is applied in the context of neighborhood function $N_k()$ is $d_k := d_{\min} + (k-1) \cdot \frac{(d_{\max} - d_{\min})}{k_{\max} - 1}$. For all experiments outlined later on we used $k_{\max} = 10$, that is, a set of ten different neighborhood functions. Note that both $d_{\min}$ and $d_{\max}$ are important parameters of RVNS.

4 Experimental Evaluation

The following three algorithmic techniques are experimentally evaluated: (1) the modified greedy algorithm (denoted by Greedy), (2) the modified state-of-the-art algorithm Cmsa [3], and (3) the ILP-based RVNS (henceforth denoted by Rvns). All algorithms were implemented in ANSI C++ using GCC 4.6.3. In addition, the ILP models involved in the three techniques were solved with the ILP solver IBM ILOG CPLEX v12.7 in one-threaded mode. The experimental evaluation has been performed on a cluster of PCs with Intel(R) Xeon(R) CPU 5670 CPUs of 12 nuclei of 2933 MHz and at least 40 Gigabytes of RAM. As the algorithms proposed in this paper are thought for the application to large-scale instances, we generated 10 instances uniformly at random for each combination of $n \in \{2000, 4000, \ldots, 20000\}$ and alphabet size $|\Sigma| \in \{10, 100\}$. In total, this set thus consists of 200 benchmark instances. The automatic configuration tool irace (see http://iridia.ulb.ac.be/irace/) was used for tuning the parameters of Cmsa and Rvns. For space reasons the tuning can not be described in more detailed. However, the obtained parameter values for Rvns are as follows (the ones for Cmsa will be provided in an extended paper version): (1) $d_{\min} = 0.3$ and $d_{\max} = 0.5$ for instances with $n \leq 10000$ and $|\Sigma| = 10$, (2) $d_{\min} = 0.1$ and $d_{\max} = 0.3$ for instances with $n > 10000$ and $|\Sigma| = 10$, (3) $d_{\min} = 0.1$ and $d_{\max} = 0.8$ for instances with $n \leq 10000$ and $|\Sigma| = 100$, and (4) $d_{\min} = 0.3$ and $d_{\max} = 0.8$ for instances with $n > 10000$ and $|\Sigma| = 10$.

All three algorithms were applied exactly once (time limit: 600 CPU sec.) to each of the 200 problem instances. The results are presented in Table 1 in terms of averages over the 10 instances for each combination of $|\Sigma|$ and n. Note that columns **avg** provide the average solution quality, whereas columns **time** provide the average computation time at which the best solutions of each run were found. The following can be observed. First, Rvns consistently outperforms Cmsa for instances with $|\Sigma| = 100$. The outcome for instances with $|\Sigma| = 10$ is not so clear. While Rvns seems to have advantages over Cmsa for the smaller problem instances (≤ 8000), the opposite is the case for the large problem instances.

Table 1
Experimental results.

$\|\Sigma\|$	n	Greedy		Cmsa		Rvns	
		avg	**time**	**avg**	**time**	**avg**	**time**
10	2000	860.4	0.5	788.8	428.6	**781.2**	406.2
	4000	1562.3	2.0	1451.4	469.0	**1437.3**	543.8
	6000	2245.4	4.6	2110.6	389.7	**2094.4**	522.9
	8000	2885.3	8.2	2763.9	449.1	**2746.3**	592.9
	10000	3525.7	13.4	**3373.4**	419.1	3386.3	588.1
	12000	4149.4	19.0	**3973.1**	518.2	4007.1	595.1
	14000	4757.4	30.3	**4569.8**	516.6	4619.1	588.6
	16000	5361.7	39.5	**5164.4**	517.0	5222.4	556.7
	18000	5945.1	45.0	5806.6	453.2	**5805.1**	596.7
	20000	6539.1	65.2	**6367.2**	464.0	6394.1	548.3
100	2000	1719.7	0.4	1712.1	2.7	**1711.0**	0.4
	4000	3206.1	1.1	3155.0	52.5	**3149.7**	2.6
	6000	4564.4	2.3	4436.0	97.6	**4423.6**	12.6
	8000	5886.7	4.0	5648.7	182.8	**5629.3**	53.7
	10000	7154.1	6.4	6778.1	184.4	**6750.4**	263.4
	12000	8367.1	9.2	7839.6	283.2	**7811.2**	366.5
	14000	9595.4	12.8	8895.2	420.9	**8866.8**	405.3
	16000	10782.7	16.0	9944.4	577.2	**9891.7**	438.7
	18000	11953.1	20.1	11004.2	527.0	**10918.9**	507.2
	20000	13156.8	26.5	12127.6	498.6	**11921.3**	567.8

5 Conclusions and Outlook

This work has added two contributions to the literature on the minimum common string partition problem. First, it introduced a modification of the algorithmic components that are used with the current state-of-the-art technique from the literature (called CMSA). Accordingly, we were able to apply CMSA to much larger problem instances than done so far. Second, using the same modifications, the paper introduced an ILP-based reduced variable neighborhood search (RVNS) technique for solving the same problem. We were able to show that this newly developed technique outperforms CMSA consistently for problem instances based on a rather large alphabet. Immediate future work will center on exploring more deeply the relation between CMSA and RVNS, for example, in the context of problem instances considering a wider range of different alphabet sizes.

References

[1] Blum, C., J. A. Lozano and P. Pinacho Davidson, *Iterative probabilistic tree search for the minimum common string partition problem*, in: M. J. Blesa, C. Blum and S. Voss, editors, *Proc. of HM 20104– 9th International Workshop on Hybrid Metaheuristics*, Lecture Notes in Computer Science **8457** (2014), pp. 154–154.

[2] Blum, C., J. A. Lozano and P. Pinacho Davidson, *Mathematical programming strategies for solving the minimum common string partition problem*, European Journal of Operational Research **242** (2015), pp. 769–777.

[3] Blum, C., P. Pinacho, M. López-Ibáñez and J. A. Lozano, *Construct, merge, solve & adapt: A new general algorithm for combinatorial optimization*, Computers & Operations Research **68** (2016), pp. 75–88.

[4] Blum, C. and G. R. Raidl, *Computational performance evaluation of two integer linear programming models for the minimum common string partition problem*, Optimization Letters **10** (2016), pp. 189–205.

[5] Chen, X., J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi and T. Jiang, *Computing the assignment of orthologous genes via genome rearrangement*, in: *Proc. of the Asia Pacific Bioinformatics Conference 2005*, 2005, pp. 363–378.

[6] Ferdous, S. M. and M. Sohel Rahman, *Solving the minimum common string partition problem with the help of ants*, in: Y. Tan, Y. Shi and H. Mo, editors, *Proc. of ICSI 2013 – 4th International Conference on Advances in Swarm Intelligence*, Lecture Notes in Computer Science **7928**, Springer Berlin Heidelberg, 2013 pp. 306–313.

[7] Ferdous, S. M. and M. Sohel Rahman, *An integer programming formulation of the minimum common string partition problem*, PloS one **10** (2015), p. e0130266.

[8] Goldstein, A., P. Kolman and J. Zheng, *Minimum common string partition problem: Hardness and approximations*, in: R. Fleischer and G. Trippen, editors, *Proc. of ISAAC 2004 – 15th International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science **3341**, Springer Berlin Heidelberg, 2005 pp. 484–495.

[9] He, D., *A novel greedy algorithm for the minimum common string partition problem*, in: I. Mandoiu and A. Zelikovsky, editors, *Proc. of ISBRA 2007 – Third International Symposium on Bioinformatics Research and Applications*, Lecture Notes in Computer Science **4463**, Springer Berlin Heidelberg, 2007 pp. 441–452.

[10] Mladenović, N. and P. Hansen, *Variable neighborhood search*, Computers & Operations Research **24** (1997), pp. 1097–1100.