A heuristic algorithm based on Lagrangian relaxation for the closest string problem*

Shunji Tanaka[a,*]

[a]*Department of Electrical Engineering, Kyoto University*
Kyotodaigaku-Katsura, Nishikyo-ku, Kyoto 615-8510, Japan

Abstract

The closest string problem that arises in both computational biology and coding theory is to find a string minimizing the maximum Hamming distance from a given set of strings. This study proposes an efficient heuristic algorithm for this NP-hard problem. The key idea is to apply the Lagrangian relaxation technique to the problem formulated as a mixed integer programming problem. This enables us to decompose the problem into trivial subproblems corresponding to each position of the strings. Furthermore, a feasible solution can be easily obtained from a solution of the relaxation. Based on this, a heuristic algorithm is constructed by combining a Lagrangian multiplier adjustment procedure and a tabu search. Computational experiments will show that the proposed algorithm can find good approximate solutions very fast.

Keywords:
closest string problem, mixed-integer programming, Lagrangian relaxation, tabu search

1. Introduction

In computational biology problems related with strings often arise: Given strings are compared with each other and their common part is searched for. Among such problems, the closest string problem, which is also referred to as consensus string problem or center string problem in the literature, is to find a string

*Tel.: +81-75-383-2204, Fax: +81-75-383-2201, E-mail: tanaka@kuee.kyoto-u.ac.jp

*A preliminary version of the paper was presented at 2010 IEEE International Conference on Systems, Man and Cybernetics, October 2010.

that minimizes the maximum Hamming distance from a given set of strings. For example, a closest string for the three strings GCGT, AGTT and CTGC is ATGT and the maximum Hamming distance is 2. This problem also appears in coding theory as an equivalent problem to compute the covering radius of codes [1].

The closest string problem is known to be NP-hard because its decision problem version over a binary alphabet is NP-complete [1]. Theoretically, PTAS (Polynomial Time Approximation Schemes) [2, 3, 4] can find a good approximate solution in polynomial time. However, they are not directly applicable to practical problems because of their high time complexity [4]. With regard to exact algorithms, some fixed-parameter algorithms have been proposed so far [5, 6, 7]. These algorithms are not for minimizing the maximum Hamming distance but for a decision problem version of the problem. Therefore, it is necessary to apply the algorithms repeatedly to obtain a closest string. Moreover, they are effective only when the maximum Hamming distance is small. There are also some researches on exact algorithms for polynomially solvable classes [8, 9], but it is not direct to extend them to the general problem. An alternative and simple way to solve the problem exactly is to formulate it as a (mixed-)integer programming problem [2, 10] and apply a general MIP solver [10, 11, 12]. It is true that this method can solve small-sized problems, but it fails to when the problem size becomes large.

Studies on (meta)heuristic algorithms have been increasing in these few years [13, 10, 14, 15, 12, 16, 17]. Most of them were applied to small-sized instances with the string length not more than 1000. The only exception is [12], where a simple local search [10] was improved to parallel multistart one and it was applied to medium-size instances with the string length up to 5000. In the case of 20 characters, solutions by their algorithm were on average at most 6.5% worse than optimal or best solutions by a commercial MIP solver, although the algorithm took 1 or 2 minutes on a parallel machine with 28 nodes.

The purpose of this study is to propose a more efficient heuristic algorithm for the closest string problem. The key idea is to apply the Lagrangian relaxation technique to the mixed-integer programming formulation, which enables us to obtain a tight lower bound and an approximate solution at the same time. Based on this, a heuristic algorithm will be constructed by combining a Lagrangian multiplier adjustment procedure and a tabu search. Computational experiments will show that the proposed algorithm can find optimal or near-optimal solutions very quickly.

This paper is organized as follows. In Section 2, the closest string problem is formally described and it is formulated as a mixed-integer programming problem. In Section 3, the formulated problem is relaxed via the Lagrangian relaxation

technique. Next, in Section 4, three lemmas on the properties of the Lagrangian relaxation are presented and proved. Then, in Section 5, a heuristic algorithm is constructed based on the relaxation. In Section 6, computational experiments are conducted and the proposed algorithm is compared with the existing algorithms. Finally, Section 7 concludes this study.

2. Problem Description and Formulation

In this section the closest string problem will be formulated as a mixed-integer programming problem. In [10], three types of formulations are presented. Among them, this study employs the formulation originally proposed in [2] whose linear programming relaxation gives a tight lower bound.

Let us denote an *alphabet* composed of M characters a_1, ..., a_M by Σ ($= \{a_1, \ldots, a_M\}$) and define an index set $\mathscr{M}$ by $\mathscr{M} = \{1, \ldots, M\}$. A *string* s over Σ is a sequence of characters in Σ. The length of s is denoted by $|s|$ and the jth character of s is denoted by $s[j]$. That is, s belongs to $\Sigma^{|s|}$ and is described by $s = s[1] \cdots s[|s|]$.

Assume that N strings s_i ($i \in \mathscr{N} = \{1, \ldots, N\}$) of length L ($|s_1| = \cdots = |s_N| = L$) over Σ are given. The closest string problem considered in this study is to find a string s of length L over Σ that minimizes the maximum Hamming distance from s_i ($i \in \mathscr{N}$). Here, the Hamming distance $d_{\mathrm{H}}(s_i, s)$ between s_i and s is defined by

$$d_{\mathrm{H}}(s_i, s) = |\{ j \in \mathscr{L} \mid s_i[j] \neq s[j] \}|, \tag{1}$$

where $\mathscr{L} = \{1, \ldots, L\}$. By using $d_{\mathrm{H}}(s_i, s)$, the problem can be formulated as as follows.

$$d^* = \min_{s} \max_{i \in \mathscr{N}} d_{\mathrm{H}}(s_i, s), \tag{2}$$

$$\text{s.t.} \quad s[j] \in \Sigma, \quad j \in \mathscr{L}. \tag{3}$$

Let us define sets of characters $\mathscr{A}^j \subseteq \Sigma$ ($j \in \mathscr{L}$) composed of those appearing at the jth position of s_i ($i \in \mathscr{N}$) by

$$\mathscr{A}^j = \bigcup_{i \in \mathscr{N}} \{ s_i[j] \}. \tag{4}$$

Its cardinality is denoted by $m^j = |\mathscr{A}^j| \leq M$ and index sets $\mathscr{M}^j$ are defined by $\mathscr{M}^j = \{1, \ldots, m^j\}$ for $j \in \mathscr{L}$. Let us also define v_i^j ($i \in \mathscr{N}$, $j \in \mathscr{L}$) so that the

v_i^j-th element of $\mathscr{A}^j$ is equal to $s_i[j]$. Next, we introduce decision variables x_{kj} ($k \in \mathscr{M}^j$, $j \in \mathscr{L}$) that are defined by

$$x_{kj} = \begin{cases} 1 & s[j] \text{ is the } k\text{th element of } \mathscr{A}^j, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Then, the closest string problem can be formulated as the following mixed-integer programming problem (IP):

$$\begin{aligned} d^* \;=\; & \min_{d,\boldsymbol{x}} \; d, && (6)\\ & \text{s.t.} \sum_{k \in \mathscr{M}^j} x_{kj} = 1, \quad j \in \mathscr{L}, && (7)\\ & \quad d + \sum_{j \in \mathscr{L}} x_{v_i^j, j} \geq L, \quad i \in \mathscr{N}, && (8)\\ & \quad d \geq 0, && (9)\\ & \quad x_{kj} \in \{0,\, 1\}, \quad k \in \mathscr{M}^j,\; j \in \mathscr{L}. && (10) \end{aligned}$$

In (IP), the constraints (7) require that exactly one character is chosen from $\mathscr{A}^j$ for s[j]. The constraints (8) define the maximum Hamming distance d from the strings s_i ($i \in \mathscr{N}$) because $d_{\mathrm{H}}(s_i, s)$ can be expressed by

$$d_{\mathrm{H}}(s_i, s) = L - \sum_{j \in \mathscr{L}} x_{v_i^j, j}. \tag{11}$$

3. Lagrangian Relaxation

Since the closest string problem is formulated as the mixed-integer programming problem (IP), we can solve it by applying a general MIP solver. However, it takes long computation time when the number of strings N or the string length L is large. Therefore, this study employs the Lagrangian relaxation technique to obtain a lower bound of (IP) and, at the same time, a good approximate solution of (IP). Here, the violation of the constraints (8) in (IP) is penalized by Lagrangian multipliers $\mu_i \geq 0$ ($i \in \mathscr{N}$) and the following Lagrangian relaxation (LR) is obtained:

$$\begin{aligned} \widehat{d}^*(\boldsymbol{\mu}) \;=\; & \min_{d,\boldsymbol{x}} \left\{ d + \sum_{i \in \mathscr{N}} \mu_i (L - d - \sum_{j \in \mathscr{L}} x_{v_i^j, j}) \right\} \\ =\; & \min_{d,\boldsymbol{x}} \left\{ \left(1 - \sum_{i \in \mathscr{N}} \mu_i \right) d - \sum_{i \in \mathscr{N}} \mu_i \sum_{j \in \mathscr{L}} x_{v_i^j, j} \right\} + L \sum_{i \in \mathscr{N}} \mu_i, \qquad (12)\\ & \text{s.t. (7), (9), (10).} \end{aligned}$$

Clearly, $\widehat{d^*}(\boldsymbol{\mu})$ satisfies

$$\widehat{d^*}(\boldsymbol{\mu}) \leq d^*. \tag{13}$$

From the first term of the righthand side of (12), $d^*(\boldsymbol{\mu})$ becomes $d^*(\boldsymbol{\mu}) = -\infty$ by choosing $d = +\infty$ if $\sum_{i\in\mathscr{N}} \mu_i > 1$ holds. Therefore, we can assume that $\sum_{i\in\mathscr{N}} \mu_i \leq 1$ for obtaining a lower bound of d^*. In this case,

$$\left(1 - \sum_{i\in\mathscr{N}} \mu_i\right) d \geq 0 \tag{14}$$

holds and the equality is always achievable by choosing d as $d = 0$. Hence, (LR) can be converted to the following equivalent problem (LR'):

$$\begin{aligned} \widehat{d^*}(\boldsymbol{\mu}) \;=\; & -\max_{\boldsymbol{x}} \sum_{j\in\mathscr{L}} \sum_{i\in\mathscr{N}} \mu_i x_{v_i^j, j} + L \sum_{i\in\mathscr{N}} \mu_i, \\ & \text{s.t. (7), (10).} \end{aligned} \tag{15}$$

Since the first term of the righthand side of (15) is decomposable with respect to $j \in \mathscr{L}$, we can decompose (LR') into L subproblems (LR$_j$), which are given by

$$\begin{aligned} \widehat{d_j^*}(\boldsymbol{\mu}) \;=\; & -\max_{\boldsymbol{x}_j} \sum_{i\in\mathscr{N}} \mu_i x_{v_i^j, j} \\ =\; & -\max_{\boldsymbol{x}_j} \sum_{k\in\mathscr{M}^j} \left(\sum_{\substack{i\in\mathscr{N} \\ v_i^j = k}} \mu_i \right) x_{kj}, && (16) \\ & \text{s.t.} \sum_{k\in\mathscr{M}^j} x_{kj} = 1, && (17) \\ & \quad x_{kj} \in \{0,\, 1\}, \quad k \in \mathscr{M}^j. && (18) \end{aligned}$$

By using $\widehat{d_j^*}(\boldsymbol{\mu})$, $\widehat{d^*}(\boldsymbol{\mu})$ is expressed by

$$\widehat{d^*}(\boldsymbol{\mu}) = \sum_{j\in\mathscr{L}} \widehat{d_j^*}(\boldsymbol{\mu}) + L \sum_{i\in\mathscr{N}} \mu_i. \tag{19}$$

The subproblem (LR$_j$) is easy to solve in $O(N)$ time and $\widehat{d_j^*}(\boldsymbol{\mu})$ is given by

$$\widehat{d_j^*}(\boldsymbol{\mu}) = -\max_{k\in\mathscr{M}^j} \sum_{\substack{i\in\mathscr{N} \\ v_i^j = k}} \mu_i. \tag{20}$$

Therefore, (LR) can be solved in $O(LN)$ time for a given set of Lagrangian multipliers $\boldsymbol{\mu}$ and the optimal objective value $\widehat{d}^*(\boldsymbol{\mu})$ of (LR) can be expressed by

$$\widehat{d}^*(\boldsymbol{\mu}) = -\sum_{j\in\mathscr{L}} \max_{k\in\mathscr{M}^j} \sum_{\substack{i\in\mathscr{N}\\ v_i^j=k}} \mu_i + L\sum_{i\in\mathscr{N}} \mu_i. \tag{21}$$

It should be noted that the *median string problem* to minimize the sum of Hamming distances can be formulated as

$$\begin{aligned} d^*_{\text{median}} &= \min_{\boldsymbol{x}} \sum_{i\in\mathscr{N}} \left(L - \sum_{j\in\mathscr{L}} x_{v_i^j,j} \right) \\ &= -\max_{\boldsymbol{x}} \sum_{i\in\mathscr{N}} \sum_{j\in\mathscr{L}} x_{v_i^j,j} + LN, \\ &\quad \text{s.t. (7), (10).} \end{aligned} \tag{22}$$

By comparing (22) with (15), we can say that (LR') is equivalent to the *weighted median string problem* to minimize the *weighted* sum of Hamming distances.

It is true that (13) always holds, but we should choose appropriate Lagrangian multipliers in (LR') to obtain a tight lower bound of d^*. For this purpose, we consider the following Lagrangian dual (DLR).

$$\widehat{d}^*_{\max} = \max_{\boldsymbol{\mu}} \widehat{d}^*(\boldsymbol{\mu}), \tag{23}$$

$$\text{s.t.} \sum_{i\in\mathscr{N}} \mu_i \le 1, \tag{24}$$

$$\mu_i \ge 0, \quad i \in \mathscr{N}. \tag{25}$$

How to solve (DLR) will be explained in Section 5.

4. Properties of Lagrangian Relaxation

In this section three lemmas on the properties of the Lagrangian relaxation will be presented.

As already explained in the preceding section, a lower bound of (IP) can be obtained by solving the Lagrangian dual (DLR). Another way to obtain a tight lower bound is to solve the linear programming relaxation (LP) of (IP) generated by relaxing the integrity constraints (10), which is given by

$$\widetilde{d}^* = \min_{d,\boldsymbol{x}} d, \tag{26}$$

$$\begin{aligned} &\text{s.t. (7), (8), (9),} \\ &\quad x_{kj} \ge 0, \quad j \in \mathscr{L},\ k \in \mathscr{M}^j. \end{aligned} \tag{27}$$

The following lemma confirms that $\widehat{d}^*_{\max}$ in (23) and $\widetilde{d}^*$ in (26) are identical.

Lemma 1. *The optimal objective values of the Lagrangian dual (DLR) and the LP relaxation (LP) are identical:*

$$\widehat{d}^*_{\max} = \max_{\boldsymbol{\mu}} \widehat{d}^*(\boldsymbol{\mu}) = \widetilde{d}^*. \tag{28}$$

PROOF. See Appendix A.

The second lemma enables us to simplify (DLR).

Lemma 2. *Optimal Lagrangian multipliers μ_i^* ($i \in \mathscr{N}$) for (DLR) satisfy $\sum_{i\in\mathscr{N}} \mu_i^* = 1$.*

PROOF. See Appendix B.

From (21) and Lemma 2, $\widehat{d}^*_{\max}(=\widetilde{d}^*)$ can be obtained by the following (DLR'):

$$\widehat{d}^*_{\max} = -\min_{\boldsymbol{\mu}} \sum_{j\in\mathscr{L}} \max_{k\in\mathscr{M}^j} \sum_{\substack{i\in\mathscr{N} \\ v_i^j=k}} \mu_i + L, \tag{29}$$

$$\text{s.t. } (25),$$

$$\sum_{i\in\mathscr{N}} \mu_i = 1. \tag{30}$$

The last lemma is on equally distributed strings.

Lemma 3. *Assume that $P(s_i[j] = \alpha_k) = p_k$ ($i \in \mathscr{N}$, $j \in \mathscr{L}$, $k \in \mathscr{M}$) holds. That is, the probability of the occurrence of each character at any position in any string is independent. Then, for a sufficiently large L there exist optimal Lagrangian multipliers satisfying*

$$\mu_i^* = 1/N, \quad i \in \mathscr{N}. \tag{31}$$

PROOF. See Appendix C.

This lemma together with the Lagrangian dual (DLR') implies an intuitive fact that a closest string is also a median string for equally distributed strings. However, the converse is not always true because the median string problem in general does not have a unique solution. For example, AC, AT, GC and GT are all median strings for the two strings AC and GT, while only AT and GC are the closest strings.

5. Proposed Algorithm

The proposed heuristic algorithm is composed of two parts. The first is a subgradient algorithm to solve the Lagrangian dual (DLR'): Lagrangian multipliers $\boldsymbol{\mu}$ are iteratively updated to maximize $\widehat{d^*}(\boldsymbol{\mu})$. More specifically, for some $\boldsymbol{\mu}$ the Lagrangian relaxation (LR') is solved and then $\boldsymbol{\mu}$ is updated by the subgradient information. It is repeated until a termination condition is satisfied. The second is a tabu search to improve a solution of (LR'). It is often the case with heuristic algorithms based on the Lagrangian relaxation technique that a solution of a Lagrangian relaxation is not always feasible for the original problem and hence some conversion algorithm is required. Fortunately, such a conversion is unnecessary in the proposed algorithm: An optimal solution of (LR') is also feasible for the original problem (IP). More precisely, an optimal solution $\boldsymbol{x}$ of (LR') is integral and hence is feasible for (IP). In addition, the minimum d satisfying the constraints (8) and (9) gives the maximum Hamming distance for this $\boldsymbol{x}$. However, it does not mean that this solution is optimal also for (IP). Therefore, a tabu search [18] is applied for further improvement. In the following subsections, the multiplier adjustment by the subgradient algorithm will be first explained, and the tabu search will be given next.

5.1. Multiplier Adjustment by Subgradient Algorithm

It is a standard method to apply the subgradient algorithm to adjust Lagrangian multipliers for a Lagrangian dual (eg. [19]) because its objective function is convex but nondifferentiable. Here, this algorithm is modified to satisfy the constraint (30).

Let $\mu_i^{(t)}$ ($i \in \mathscr{N}$) be Lagrangian multipliers at the tth iteration and $x_{kj}^{(t)}$ ($k \in \mathscr{M}^j$, $j \in \mathscr{L}$) be the optimal solution of (LR') for $\boldsymbol{\mu}^{(t)}$. The multipliers $\mu_i^{(t+1)}$ at the $(t+1)$th iteration are calculated as follows.

$$\nu_i^{(t+1)} = \max\left\{ \mu_i^{(t)} + \alpha^{(t)} \frac{d_{\text{best}} - \widehat{d^*}(\boldsymbol{\mu}^{(t)})}{\sum_{i \in \mathscr{N}} (g_i^{(t)})^2} g_i^{(t)},\ 0 \right\}, \tag{32}$$

$$\mu_i^{(t+1)} = \frac{\nu_i^{(t+1)}}{\sum_{i \in \mathscr{N}} \nu_i^{(t+1)}}, \tag{33}$$

$$g_i^{(t)} = L - \widehat{d^*}(\boldsymbol{\mu}^{(t)}) - \sum_{j \in \mathscr{L}} x^{(t)}_{\nu_i^j, j}, \tag{34}$$

where $\alpha^{(t)}$ is the step size and d_{best} is the objective value of the current best solution of (IP) obtained heuristically by the tabu search in the next subsection. The only difference from the ordinary subgradient algorithm is that multipliers are scaled by (33) to satisfy the constraint (30).

5.2. Tabu Search

To obtain a good approximate solution of (IP), the solution $\boldsymbol{x}^{(t)}$ of the relaxation (LR') for $\boldsymbol{\mu}^{(t)}$ is improved by a simple tabu search. A single move in the tabu search is determined as follows. Here, the string yielded by the current solution $\boldsymbol{x}^{(t)}$ is denoted by s.

1. Let the objective value of s be d $(= \max_{i \in \mathcal{N}} d_{\mathrm{H}}(s_i, s))$.
2. Let the index set of strings $\mathcal{N}_{\max} \subseteq \mathcal{N}$ be
$$\mathcal{N}_{\max} = \left\{ i \,\middle|\, d_{\mathrm{H}}(s_i, s) = d \right\}. \tag{35}$$
3. Every position p such that $s_i[p] = s[p]$ holds for some $i \in \mathcal{N}_{\max}$ is labeled as tabu-active.
4. For every tabu-inactive position p:
   - Obtain the set of candidate characters $\mathscr{C}^p \subset \mathscr{A}_p$ by
   $$\mathscr{C}^p = \bigcup_{i \in \mathcal{N}_{\max}} \{ s_i[p] \}. \tag{36}$$
   - For every element c_q^p of $\mathscr{C}^p$ $(1 \leq q \leq |\mathscr{C}^p|)$:
     - Generate a candidate string $\widehat{s}^{(p,q)}$ by
     $$\widehat{s}^{(p,q)}[j] = \begin{cases} s[j], & j \neq p, \\ c_q^p, & j = p. \end{cases} \tag{37}$$
     - Compute $V(p,q)$ by
     $$V(p,q) = \sum_{i \in \mathcal{N}} (d_{\mathrm{H}}(s_i, \widehat{s}^{(p,q)}) - d_{\mathrm{H}}(s_i, s)) d_{\mathrm{H}}(s_i, s). \tag{38}$$
5. Find (p^*, q^*) that minimizes $V(p^*, q^*)$.
6. Update s by $s := \widehat{s}^{(p^*,q^*)}$ and the position p^* is labeled as tabu-active.

In every move of the tabu search, it is ensured that the maximum Hamming distance d does not increase. For this purpose, such a position that changing the character of s at that position increases d is labeled as tabu-active in 3) of the procedure. To determine the next move, an alternative objective function $V(p,q)$ in (38) is used in place of the maximum Hamming distance for $\widehat{s}^{(p,q)}$. It is because the maximum Hamming distance may not decrease in a single move and hence is inappropriate for determining the best move. Since $(d_{\mathrm{H}}(s_i,\widehat{s}^{(p,q)}) - d_{\mathrm{H}}(s_i,s))$ takes only -1, 0 and 1, $V(p,q)$ can be rewritten as follows.

$$\begin{aligned} V(p,q) &= \sum_{i\in\mathscr{N}} V_i(p,q), \\ V_i(p,q) &= \begin{cases} -d_{\mathrm{H}}(s_i,s) & \text{if } d_{\mathrm{H}}(s_i,s) \text{ decreases,} \\ d_{\mathrm{H}}(s_i,s) & \text{if } d_{\mathrm{H}}(s_i,s) \text{ increases,} \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{39}$$

This implies that $V(p,q)$ becomes small if

(a) The number of strings s_i satisfying $d_{\mathrm{H}}(s_i,\widehat{s}^{(p,q)}) < d_{\mathrm{H}}(s_i,s)$ is large.
(b) $d_{\mathrm{H}}(s_i,s)$ is large when $d_{\mathrm{H}}(s_i,\widehat{s}^{(p,q)}) < d_{\mathrm{H}}(s_i,s)$.
(c) The number of strings s_i satisfying $d_{\mathrm{H}}(s_i,\widehat{s}^{(p,q)}) > d_{\mathrm{H}}(s_i,s)$ is small.
(d) $d_{\mathrm{H}}(s_i,s)$ is small when $d_{\mathrm{H}}(s_i,\widehat{s}^{(p,q)}) > d_{\mathrm{H}}(s_i,s)$.

For example, let us consider that four given strings s_1, …, s_4 and the current solution s for them are as in Figure 1. Since $d_{\mathrm{H}}(s_2,s) = d_{\mathrm{H}}(s_4,s) = \max_{1\le i\le 4} d_{\mathrm{H}}(s_i,s)$, $\mathscr{N}_{\max}$ becomes $\mathscr{N}_{\max} = \{2,4\}$. The only candidate position p in 4) of the procedure is 4 and $\mathscr{C}^4 = \{\mathrm{G},\mathrm{A}\}$. The objective values V for the candidate characters in $\mathscr{C}^4$ are $V(4,1) = 1-3+0+0 = -2$ and $V(4,2) = 1+0-2-3 = -4$. Hence "A" is chosen for the next move and the new solution is ATCAGT.

6. Computational Experiments

6.1. Results for randomly generated instances

The algorithm proposed in the preceding section is applied to randomly generated instances. Following [12], they are generated as follows.

(1) Instances where each character occurs with the equal probability $1/M$. Three types of alphabets are considered in these instances: $M = 2$ (binary codes, $\Sigma = \{0,1\}$), $M = 4$ (DNA sequences, $\Sigma = \{\mathrm{A},\mathrm{C},\mathrm{G},\mathrm{T}\}$) and $M = 20$ (amino acid sequences).

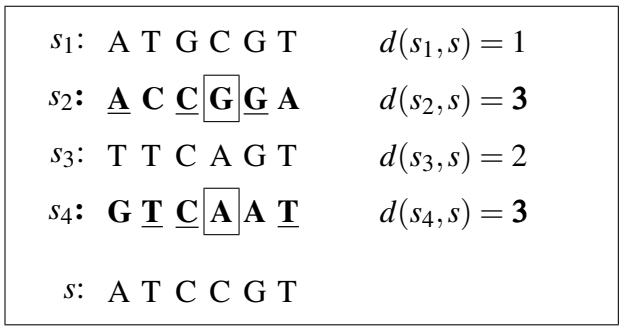

Figure 1: Choice of a move in the tabu search ($L = 6$, $M = 4$, $N = 4$)

(2) As another type of instances with $M = 4$, the probabilities of the occurrences of the characters A, C, G and T are chosen as 0.14, 0.36, 0.36 and 0.14, respectively. It simulates the genome of the Actinobacteria Streptomyces coelicolor whose GC-content is 72%.

The number of strings N and the string length L are chosen as $N \in \{10, 20, \ldots, 50\}$ and $L \in \{1000, 2000, \ldots, 5000\}$. For each combination of M, N and L, 10 instances are generated. The computation is performed by running a program coded in C (gcc) on a desktop computer with an Intel Core™ i7 960 CPU (3.2GHz). Parameters for the subgradient algorithm in 5.1 and the tabu search in 5.2 are determined by some preliminary experiments. In the subgradient algorithm, the step size $\alpha^{(t)}$ is initialized as $\alpha^{(0)} = 2.0$ and decreased by $\alpha^{(t+1)} = 0.8\alpha^{(t)}$ when the current best lower bound $\max_{s \le t} \widehat{d}^*(\boldsymbol{\mu}^{(s)})$ is not improved for 5 successive iterations. The initial Lagrangian multipliers $\boldsymbol{\mu}^{(0)}$ are chosen as $\mu_i^{(0)} = 1/N$ from Lemma 3. The iteration is terminated when the gap $(d_{\text{best}} - \max_{s \le t} \widehat{d}^*(\boldsymbol{\mu}^{(s)}))$ becomes less than one, i.e. the current best solution is proved to be optimal, or $\alpha^{(t)} \le 10^{-3}$ is satisfied. In the tabu search, the tabu length is chosen as $\max(\lceil N/10 \rceil, 2)$. The tabu search is terminated if the objective value d is not improved in $4N$ successive moves.

The results are summarized in Tables 1–4. In these tables, "LB av." denotes the average of the best lower bounds obtained by the subgradient algorithm for

(DLR'), "d_{best} av." the average of the best objective values (maximum Hamming distances) of approximate solutions obtained by the proposed algorithm, "opt." the number of instances for which the gap between the best lower bound and the best objective value is less than one, "average gap abs." the average gap, "average gap %" the average gap in percent, "maximum gap abs." the maximum gap, "maximum gap %" the maximum gap in percent, and "time (s)" the average CPU time in seconds. Here, the averages and maxima are calculated over 10 instances. In addition, the average gap is the average of $(d_{\mathrm{best}} - \lceil \mathrm{LB} \rceil)$, and the average gap in percent is the average of $100(d_{\mathrm{best}} - \lceil \mathrm{LB} \rceil)/\lceil \mathrm{LB} \rceil$, where LB and d_{best} denote the best lower bound and the objective value of the best approximate solution, respectively. The maximum gap and the maximum gap in percent are calculated in a similar manner. The reason why LB is rounded upwards to the nearest integer is that $\lceil LB \rceil$ actually is a lower bound because of the integrity of the optimal objective value.

From the tables, we can see that very good solutions are obtained by the proposed algorithm. Indeed, the gap is at most 2. This fact also implies that a tight lower bound is obtained by the Lagrangian relaxation technique. The instances with $N = 10$ are the easiest and the solutions are almost always optimal. The problem becomes harder as N increases and when $N = 50$, optimality of approximate solutions is not ensured except when $M = 20$. The hardest instance type seems the one with $M = 2$ (Table 1) or with 72% GC-content (Table 3). It follows that instances with a smaller number of characters are harder for the proposed algorithm.

*6.2. **Comparison with the MIP approach***

Next, the proposed algorithm will be compared with the MIP approach. The instances in the preceding subsection are solved also by a general MIP solver to (IP). As the MIP solver, IBM ILOG CPLEX 12.1 is applied and the maximum CPU time is limited to 3600 seconds. Please note that CPLEX uses 8 threads, while the proposed algorithm is a single-thread program.

The results are summarized in Tables 5–8. In these tables, "opt." denotes the number of optimally solved instances by the MIP approach, "d_{MIP} the objective value of the optimal (or best) solution obtained within the time limit. "$d_{\mathrm{best}} - d_{\mathrm{MIP}}$" denotes the difference between the objective values by the proposed algorithm and the MIP approach, and "-1", "0" and "1" denote the numbers of instances for which the difference are -1, 0 and 1, respectively. "average gap" and "maximum gap" are computed in a similar way to those in Tables 1–4 for d_{best} and d_{MIP}. For example, "average gap %" denotes the average of

$100(d_{\text{best}} - d_{\text{MIP}})/d_{\text{MIP}}$. Finally, "time (s)" denotes the average CPU time of the MIP approach in seconds.

From the tables, it can be observed that the absolute difference between the objective values of the proposed algorithm and the MIP approach is at most 1 although the former is much faster than the latter. This fact confirms the effectiveness of the proposed algorithm. Moreover, the proposed algorithm can always find solutions better than or equal to those by the MIP approach in much shorter time for the instances with $M = 20$ (Table 8). On the other hand, a better solution is found only for a single instance of the instances with $M = 4$ and 72% GC-content (Table 7). Therefore, we can say that the instances with $M = 20$ are the easiest and those with $M = 4$ and 72% GC-content (especially when N is large) are the hardest for the proposed algorithm.

6.3. Comparison with the existing heuristic algorithms

In this subsection the proposed algorithm will be compared with the existing (meta)heuristic algorithms [15, 12, 16, 17]. Here, [13] and [14] are not taken into consideration because experimental results presented in [13, 14] are insufficient for a significant comparison: The GA in [13] was applied only to a single instance, and the parallel GA and SA in [14] to such instances where both the string length L and the number of strings N are not more than 40.

First, the compounded genetic and simulated annealing algorithm (CGSA) in [15] is considered. In [15], 18 randomly generated instances with $M = 2$, $N \in \{10, 15, 20\}$ and $L \in \{300, 400, \ldots, 800\}$ were solved by the CGSA and the results are summarized in Table 9. In Table 9, "gap" is the gap between the lower bound by the linear programming relaxation (LP) and the best solution by CGSA among 20 trials, and "time" is the average CPU time on a computer with a 2.5GHz Intel Pentium4 CPU. By comparing Table 9 with the first two rows of Table 1, we can safely conclude that the proposed algorithm outperforms the CGSA in both solution quality and CPU time even if the difference of the computer speed is taken into account.

In [12] a multistart local search was run on a parallel machine with 28 nodes. Then, its solutions were compared with optimal ones obtained by applying a commercial MIP solver to (IP) (when it failed to solve (IP) within 3600 seconds, the best solution was used instead). The average gaps between the objective values of their solutions and those of optimal (or best) solutions are summarized in Table 10. In addition to them, average CPU times are also presented. By comparing "average gap" in Table 10 with "average gap %" in Tables 5–8, we can verify that the proposed algorithm can find better solutions than the algorithm in [12] because

the latter is much smaller than the former. Moreover, Table 10 and Tables 1–4 tell that the proposed algorithm is faster than the algorithm in [12] although the former is coded as a single-thread program, while the latter was run on a parallel machine.

Next, the data-coded GA in [16] is considered. Here, exactly the same instances with $L = 500$ in [16] are solved by the proposed algorithm and the results are presented in Table 11 together with those by the data-coded GA in [16]. In Table 11, "objective" of the GA denotes the best solutions among 40 trials and "time" the average computation time of the trials on a 2.53GHz Intel Pentium4 CPU. This table shows that the proposed algorithm always yields a better solution than the data-coded GA and most instances are solved optimally, although the former is applied only once for each instance. Furthermore, the former is much faster than the latter even if the difference of the computer speed is taken into account.

Finally, the Ant Colony Optimization algorithm in [17] is considered. The instances in [17] are generated randomly for $M = 4$, $N \in \{10, 20, \ldots, 50\}$ and $L \in \{10, 20, \ldots, 50\} \cup \{100, 200, \ldots, 1000\}$. Table 12 summarizes the results for the five instances with $L = 1000$, where the averages are of 20 trials and the computation was performed on a 1.86GHz Intel Pentium M 750 CPU.

It is possible that uniform randomness of the instances in [17] is not sufficient because the objective values are too worse than those in Table 2 (note that the objective values in Tables 1–4 are consistent with the optimal (or best) ones in [12] except for instances with GC-content 72%). Nevertheless, the proposed algorithm seems better than the Ant Colony Optimization algorithm in [17]. For example, the proposed algorithm can always find optimal solutions for instances with $N = 10$ and $L = 1000$, while the solution by the Ant Colony Optimization algorithm deviates every trial. In addition, the former is faster than the latter even if the difference of the computer speed is taken into account.

7. Conclusion

In this study an efficient heuristic algorithm was proposed for the closest string problem based on the Lagrangian relaxation technique. First, some properties of the Lagrangian relaxation including the relation to the median string problem were proved and presented. Next, a heuristic algorithm was constructed. In this framework, Lagrangian multipliers are adjusted by the subgradient algorithm and a tabu search is applied to improve solutions of the Lagrangian relaxation obtained in the course of the algorithm. Computational experiments showed that the proposed algorithm can find optimal or near-optimal solutions very quickly and outperforms

the existing heuristic algorithms. Future research directions will be to construct an exact algorithm based on the properties shown in this study, to extend the proposed algorithm to the closest substring problem, and so on.

Acknowledgement

This research is partly supported by the Kayamori Foundation of Informational Science Advancement.

References

[1] Frances M, Litman A. On covering problems of codes. Theor Comput Syst 1997;30:113–119.

[2] Ben-Dor A, Lancia G, Lancia J, Perone J, Ravi R. Banishing bias from consensus sequences. Lect Notes Comput Sci 1997;1264:247–261.

[3] Deng X, Li G, Wang L. Center and distinguisher for strings with unbounded alphabet. J Comb Optim 2002:6;383–400.

[4] Li M, Ma B, Wang L. On the closest string and substring problems. J ACM 2002:49;157–171.

[5] Gramm J, Niedermeier R, Rossmanith P. Fixed-parameter algorithms for closest string and related problems. Algorithmica 2003:37;25–42.

[6] Ma B, Sun X. More efficient algorithms for closest string and substring problems. Lect Notes Comput Sci 2008;4955:396-409.

[7] Wang L, Zhu B. Efficient algorithms for the closest string and distinguishing string selection problems. Lect Notes Comput Sci 2009;5598:261–270.

[8] Boucher C, Brown DG, Durocher S. On the structure of small motif recognition instances. Lect Notes Comput Sci 2008;5280:269-281.

[9] Amir A, Landau GM, Na JC, Park H, Park K, Sim JS. Consensus optimizing both distance sum and radius. Lect Notes Comput Sci 2009;5721:234-242.

[10] Meneses CN, Lu Z, Oliveira CAS, Pardalos PM. Optimal solutions for the closest-string problem via integer programming. INFORMS J Comput 2004:16;419–429.

[11] Meneses CN, Oliveira CAS, Pardalos PM. Optimization techniques for string selection and comparison problems in genomics: Applying integer programming and heuristic algorithms for improved solutions. IEEE Eng Med Biol 2005:24;81–87.

[12] Gomes FC, Meneses CN, Pardalos PM, Viana GVR. A parallel multistart algorithm for the closest string problem. Comput Oper Res 2008:35;3636–3643.

[13] Mauch H, Melzer MJ, Hu JS. Genetic algorithm approach for the closest string problem. Proc Comput Syst Bioinformatics (CSB) 2003:560–561.

[14] Liu X, He H, Sýkora O. Parallel genetic algorithm and parallel simulated annealing algorithm for the closest string problem. Lect Notes Comput Sci 2005:3584;591–597.

[15] Liu X, Holger M, Hao Z, Wu G. A compounded genetic and simulated annealing algorithm for the closest string problem. Proc Int Conf Bioinformatics Biomed Eng (iCBBE) 2008:702–705.

[16] Julstrom BA. A data-based coding of candidate strings in the closest string problem. Proc Ann Conf Companion Genetic Evolut Comput Conf (GECCO) 2009:2053–2058.

[17] Faro S, Pappalardo E. Ant-CSP: An Ant Colony Optimization algorithm for the closest string problem. Lect Notes Comput Sci 2010:5901;370–381.

[18] Glover F, Taillard E, de Werra D. A user's guide to tabu search. Ann Oper Res 1993:41;3–28.

[19] Fisher ML. An applications oriented guide to Lagrangian relaxation. Interfaces 1985:15;10–21.

Appendix A. Proof of Lemma 1

Let us consider the linear programming relaxation (LLR) of (LR). Since (LLR) has integral optimal solutions, the optimal objective values of (LR) and (LLR) are identical. It is well-known from the linear programming theory that the optimal objective value of a linear programming problem is achievable by its Lagrangian relaxation if multipliers are chosen as equal to the optimal dual variables. Since

(LLR) is identical to a Lagrangian relaxation of (LP), we can conclude that (28) holds. ■

Appendix B. Proof of Lemma 2

From (21), $\widehat{d}^*_{\max}$ satisfies

$$\begin{aligned}\widehat{d}^*_{\max} &= -\sum_{j\in\mathscr{L}}\max_{k\in\mathscr{M}^j}\sum_{\substack{i\in\mathscr{N}\\ v_i^j=k}}\mu_i^* + L\sum_{i\in\mathscr{N}}\mu_i^* \\ &= \sum_{j\in\mathscr{L}}\left(\sum_{i\in\mathscr{N}}\mu_i^* - \max_{k\in\mathscr{M}^j}\sum_{\substack{i\in\mathscr{N}\\ v_i^j=k}}\mu_i^*\right) \geq 0. \end{aligned} \tag{B.1}$$

Assume that $\sum_{i\in\mathscr{N}}\mu_i^* \neq 1$. Since $\sum_{i\in\mathscr{N}}\mu_i^* < 1$ from the discussion in Section 3, $\sum_{i\in\mathscr{N}}\mu_i^*$ can be expressed by $\sum_{i\in\mathscr{N}}\mu_i^* = 1/\beta$ where $\beta > 1$. Now, let us define $\mu_i^\dagger$ ($i\in\mathscr{N}$) by $\mu_i^\dagger = \beta\mu_i$. Then, $\widehat{d}(\boldsymbol{\mu}^\dagger)$ is given by

$$\begin{aligned}\widehat{d}(\boldsymbol{\mu}^\dagger) &= -\sum_{j\in\mathscr{L}}\max_{k\in\mathscr{M}^j}\sum_{\substack{i\in\mathscr{N}\\ v_i^j=k}}\mu_i^\dagger + L\sum_{i\in\mathscr{N}}\mu_i^\dagger \\ &= -\beta\sum_{j\in\mathscr{L}}\max_{k\in\mathscr{M}^j}\sum_{\substack{i\in\mathscr{N}\\ v_i^j=k}}\mu_i^* + \beta L\sum_{i\in\mathscr{N}}\mu_i^* \\ &= \beta\widehat{d}^*_{\max} > \widehat{d}^*_{\max}. \end{aligned} \tag{B.2}$$

It contradicts the optimality of $\boldsymbol{\mu}^*$. ■

Appendix C. Proof of Lemma 3

Here, only the case when $N = 2$ and $\Sigma = \{\mathrm{A}, \mathrm{B}\}$ ($M = 2$) is considered for simplicity because the same arguments hold in more general cases. In the considered case $(s_1[j], s_2[j])$ can be (A, A), (B, B), (A, B) and (B, A). From the assumption of the lemma, the numbers of the occurrences of (A, B) and (B, A) are the same for a sufficiently large L. Therefore, if we define sets of positions $\mathscr{P}_l \subset \mathscr{L}$ ($l = 1,2,3$) by

$$\begin{aligned}\mathscr{P}_1 &= \{j\in\mathscr{L} \,|\, s_1[j] = s_2[j]\}, \\ \mathscr{P}_2 &= \{j\in\mathscr{L} \,|\, s_1[j] = \mathrm{A} \wedge s_2[j] = \mathrm{B}\}, \\ \mathscr{P}_3 &= \{j\in\mathscr{L} \,|\, s_1[j] = \mathrm{B} \wedge s_2[j] = \mathrm{A}\}, \end{aligned} \tag{C.1}$$

then $|\mathscr{P}_2| = |\mathscr{P}_3|$ holds.

Let $Q : \mathscr{P}_2 \to \mathscr{P}_3$ be an arbitrary bijective mapping. By Q, define a bijective mapping $\mathscr{S} : s \to s'$ on Σ^L as follows.

$$s'[j] = \begin{cases} s[j] & j \in \mathscr{P}_1, \\ s[Q(j)] & j \in \mathscr{P}_2, \\ s[Q^{-1}(j)] & j \in \mathscr{P}_3. \end{cases} \tag{C.2}$$

Then, $\mathscr{S}(s_1) = s_2$ and $\mathscr{S}(s_2) = s_1$ hold. This implies that s_1 commutes s_2 and vice versa only by re-sequencing the strings. Therefore, if $\boldsymbol{\mu}^* = (\mu_1^*, \mu_2^*)$ is optimal for (DLR), then also is $\boldsymbol{\mu}^\dagger = (\mu_2^*, \mu_1^*)$.

Let us assume that $\mu_1^* \neq \mu_2^*$ and define $\sigma_1 = 2$ and $\sigma_2 = 1$. Then, from the optimality of $\boldsymbol{\mu}^\dagger$,

$$\widehat{d}^*_{\max} = L - \sum_{j \in \mathscr{L}} \max_{k \in \mathscr{M}^j} \sum_{\substack{i \in \mathscr{N} \\ v_i^j = k}} \mu_i^* = L - \sum_{j \in \mathscr{L}} \max_{k \in \mathscr{M}^j} \sum_{\substack{i \in \mathscr{N} \\ v_i^j = k}} \mu_{\sigma_i}^* \tag{C.3}$$

holds. Since

$$\sum_{j \in \mathscr{L}} \max_{k \in \mathscr{M}^j} \sum_{\substack{i \in \mathscr{N} \\ v_i^j = k}} \mu_i \tag{C.4}$$

is a convex function of $\boldsymbol{\mu}$,

$$\sum_{j \in \mathscr{L}} \max_{k \in \mathscr{M}^j} \sum_{\substack{i \in \mathscr{N} \\ v_i^j = k}} (\mu_i^* + \mu_{\sigma_i}^*) \leq \sum_{j \in \mathscr{L}} \max_{k \in \mathscr{M}^j} \sum_{\substack{i \in \mathscr{N} \\ v_i^j = k}} \mu_i^* + \sum_{j \in \mathscr{L}} \max_{k \in \mathscr{M}^j} \sum_{\substack{i \in \mathscr{N} \\ v_i^j = k}} \mu_{\sigma_i}^* \tag{C.5}$$

is satisfied. It follows that

$$L - \sum_{j \in \mathscr{L}} \max_{k \in \mathscr{M}^j} \sum_{\substack{i \in \mathscr{N} \\ v_i^j = k}} \frac{\mu_i^* + \mu_{\sigma_i}^*}{2} \geq \frac{\widehat{d}^*_{\max} + \widehat{d}^*_{\max}}{2} = \widehat{d}^*_{\max} \tag{C.6}$$

is satisfied. Therefore, $(\boldsymbol{\mu}^* + \boldsymbol{\mu}^\dagger)/2 = ((\mu_1^* + \mu_2^*)/2, (\mu_1^* + \mu_2^*)/2)$ is also optimal for (DLR). ■

Table 1: Computational results for instances with $M = 2$

L	N	LB av.	d_{best} av.	opt.	average gap		maximum gap		time (s)
					abs.	%	abs.	%	
1000	10	378.03	378.7	8	0.20	0.05	1	0.26	0.11
1000	20	411.69	412.6	6	0.40	0.10	1	0.24	0.38
1000	30	427.63	428.9	5	0.60	0.14	2	0.47	1.32
1000	40	439.05	440.8	0	1.30	0.30	2	0.46	2.18
1000	50	445.65	447.7	0	1.70	0.38	2	0.45	2.94
2000	10	754.34	755.1	7	0.30	0.04	1	0.13	0.59
2000	20	821.70	822.7	5	0.50	0.06	1	0.12	1.02
2000	30	855.46	856.7	4	0.60	0.07	1	0.12	2.15
2000	40	875.43	877.3	0	1.40	0.16	2	0.23	4.62
2000	50	889.61	891.9	0	1.90	0.21	2	0.23	6.19
3000	10	1130.53	1131.3	6	0.40	0.04	1	0.09	1.56
3000	20	1234.99	1235.9	6	0.40	0.03	1	0.08	1.92
3000	30	1281.96	1283.3	1	0.90	0.07	1	0.08	4.08
3000	40	1313.00	1314.7	0	1.00	0.08	1	0.08	6.54
3000	50	1331.68	1333.9	0	1.80	0.14	2	0.15	9.76
4000	10	1508.98	1509.8	5	0.50	0.03	1	0.07	3.50
4000	20	1646.14	1646.9	8	0.20	0.01	1	0.06	1.63
4000	30	1710.44	1711.7	1	0.90	0.05	1	0.06	7.42
4000	40	1748.77	1750.5	0	1.10	0.06	2	0.11	8.50
4000	50	1776.39	1778.4	0	1.50	0.08	2	0.11	13.08
5000	10	1881.76	1882.5	7	0.30	0.02	1	0.05	3.16
5000	20	2059.80	2060.6	7	0.30	0.01	1	0.05	4.14
5000	30	2139.53	2140.5	6	0.40	0.02	1	0.05	6.26
5000	40	2184.16	2185.7	0	1.10	0.05	2	0.09	11.89
5000	50	2219.12	2220.9	0	1.40	0.06	2	0.09	15.84

Table 2: Computational results for instances with $M = 4$

L	N	LB av.	d_{best} av.	opt.	average gap		maximum gap		time (s)
					abs.	%	abs.	%	
1000	10	579.79	580.3	10	0.00	0.00	0	0.00	0.00
1000	20	631.27	631.8	9	0.10	0.02	1	0.16	0.42
1000	30	653.94	655.1	4	0.60	0.09	1	0.15	1.87
1000	40	668.09	669.8	0	1.20	0.18	2	0.30	3.00
1000	50	676.45	678.5	0	1.50	0.22	2	0.30	3.80
2000	10	1161.78	1162.1	10	0.00	0.00	0	0.00	0.01
2000	20	1262.41	1262.9	10	0.00	0.00	0	0.00	0.08
2000	30	1306.51	1307.5	7	0.30	0.02	1	0.08	2.80
2000	40	1333.80	1335.5	0	1.20	0.09	2	0.15	6.41
2000	50	1352.29	1354.3	0	1.40	0.10	2	0.15	8.24
3000	10	1739.01	1739.5	10	0.00	0.00	0	0.00	0.03
3000	20	1892.54	1893.0	10	0.00	0.00	0	0.00	0.03
3000	30	1960.41	1961.5	3	0.70	0.04	1	0.05	6.08
3000	40	2000.66	2002.1	1	0.90	0.04	1	0.05	9.41
3000	50	2027.91	2029.7	0	1.40	0.07	2	0.10	12.73
4000	10	2323.26	2323.5	10	0.00	0.00	0	0.00	0.03
4000	20	2524.22	2524.6	9	0.10	0.00	1	0.04	2.05
4000	30	2614.83	2615.6	7	0.30	0.01	1	0.04	4.28
4000	40	2667.43	2668.8	0	1.00	0.04	1	0.04	12.27
4000	50	2703.05	2704.9	0	1.20	0.04	2	0.07	16.99
5000	10	2903.70	2904.2	10	0.00	0.00	0	0.00	0.04
5000	20	3153.97	3154.6	8	0.20	0.01	1	0.03	4.30
5000	30	3267.63	3268.5	7	0.30	0.01	1	0.03	7.14
5000	40	3333.69	3335.0	3	0.70	0.02	1	0.03	13.72
5000	50	3378.38	3380.1	0	1.30	0.04	2	0.06	21.31

Table 3: Computational results for instances with 72% GC-content

L	N	LB av.	d_{best} av.	opt.	average gap		maximum gap		time (s)
					abs.	%	abs.	%	
1000	10	524.86	525.4	10	0.00	0.00	0	0.00	0.00
1000	20	563.15	564.2	4	0.60	0.11	1	0.18	0.96
1000	30	578.57	580.0	0	1.00	0.17	1	0.17	1.93
1000	40	587.94	589.7	0	1.20	0.20	2	0.34	2.56
1000	50	593.71	596.0	0	1.90	0.32	2	0.34	3.31
2000	10	1053.51	1053.8	10	0.00	0.00	0	0.00	0.00
2000	20	1125.96	1126.8	7	0.30	0.03	1	0.09	0.92
2000	30	1156.61	1157.9	4	0.60	0.05	1	0.09	3.96
2000	40	1174.14	1176.1	0	1.50	0.13	2	0.17	5.68
2000	50	1186.96	1189.3	0	1.80	0.15	2	0.17	7.22
3000	10	1577.18	1577.6	10	0.00	0.00	0	0.00	0.01
3000	20	1687.69	1688.7	5	0.50	0.03	1	0.06	2.20
3000	30	1735.03	1736.4	3	0.70	0.04	1	0.06	6.11
3000	40	1761.60	1763.5	0	1.50	0.09	2	0.11	9.18
3000	50	1777.94	1780.1	0	1.70	0.10	2	0.11	11.25
4000	10	2098.59	2098.9	10	0.00	0.00	0	0.00	0.01
4000	20	2249.99	2250.7	8	0.20	0.01	1	0.04	1.28
4000	30	2313.14	2314.5	2	0.80	0.03	1	0.04	7.97
4000	40	2344.90	2346.7	0	1.20	0.05	2	0.09	12.18
4000	50	2370.78	2373.2	0	1.90	0.08	2	0.08	15.78
5000	10	2626.12	2626.5	10	0.00	0.00	0	0.00	0.03
5000	20	2813.84	2814.4	7	0.30	0.01	1	0.04	3.79
5000	30	2889.76	2891.2	2	0.80	0.03	1	0.03	10.11
5000	40	2934.45	2936.1	1	1.10	0.04	2	0.07	16.73
5000	50	2961.16	2963.5	0	1.80	0.06	2	0.07	20.27

Table 4: Computational results for instances with $M = 20$

L	N	LB av.	d_{best} av.	opt.	average gap		maximum gap		time (s)
					abs.	%	abs.	%	
1000	10	781.91	782.3	10	0.00	0.00	0	0.00	0.00
1000	20	838.08	838.4	10	0.00	0.00	0	0.00	0.01
1000	30	861.30	861.8	10	0.00	0.00	0	0.00	0.01
1000	40	874.30	874.9	9	0.10	0.01	1	0.11	0.70
1000	50	883.14	883.7	8	0.20	0.02	1	0.11	1.80
2000	10	1566.15	1566.6	10	0.00	0.00	0	0.00	0.00
2000	20	1677.18	1677.7	10	0.00	0.00	0	0.00	0.02
2000	30	1722.76	1723.1	10	0.00	0.00	0	0.00	0.41
2000	40	1748.52	1749.1	10	0.00	0.00	0	0.00	0.04
2000	50	1766.37	1767.1	9	0.10	0.01	1	0.06	2.31
3000	10	2347.40	2347.8	10	0.00	0.00	0	0.00	0.01
3000	20	2516.07	2516.6	10	0.00	0.00	0	0.00	0.03
3000	30	2584.07	2584.6	10	0.00	0.00	0	0.00	0.12
3000	40	2623.20	2623.6	10	0.00	0.00	0	0.00	0.40
3000	50	2649.37	2650.1	8	0.20	0.01	1	0.04	8.72
4000	10	3129.40	3129.8	10	0.00	0.00	0	0.00	0.01
4000	20	3353.93	3354.4	10	0.00	0.00	0	0.00	0.04
4000	30	3446.56	3447.1	10	0.00	0.00	0	0.00	0.07
4000	40	3498.05	3498.6	10	0.00	0.00	0	0.00	0.12
4000	50	3532.48	3533.1	9	0.10	0.00	1	0.03	7.63
5000	10	3911.66	3912.2	10	0.00	0.00	0	0.00	0.02
5000	20	4190.77	4191.3	10	0.00	0.00	0	0.00	0.06
5000	30	4307.47	4307.8	10	0.00	0.00	0	0.00	1.38
5000	40	4372.59	4373.1	10	0.00	0.00	0	0.00	0.61
5000	50	4416.49	4417.0	10	0.00	0.00	0	0.00	0.73

Table 5: Comparison with the MIP approach ($M = 2$)

L	N	opt.	$d_{\text{best}} - d_{\text{MIP}}$			average gap		maximum gap		time (s)
			-1	0	1	abs.	%	abs.	%	
1000	10	8	0	10	0	0.00	0.00	0	0.00	720.00
1000	20	7	0	9	1	0.10	0.02	1	0.24	1103.93
1000	30	2	5	4	1	-0.40	-0.09	1	0.23	2930.18
1000	40	0	0	8	2	0.20	0.05	1	0.23	3600.00
1000	50	0	0	9	1	0.10	0.02	1	0.22	3600.00
2000	10	9	0	10	0	0.00	0.00	0	0.00	360.09
2000	20	5	0	10	0	0.00	0.00	0	0.00	1800.31
2000	30	2	3	6	1	-0.20	-0.02	1	0.12	2884.23
2000	40	0	0	7	3	0.30	0.03	1	0.11	3600.00
2000	50	0	0	9	1	0.10	0.01	1	0.11	3600.00
3000	10	9	0	10	0	0.00	0.00	0	0.00	360.18
3000	20	6	0	10	0	0.00	0.00	0	0.00	1444.68
3000	30	2	0	9	1	0.10	0.01	1	0.08	2884.90
3000	40	0	0	10	0	0.00	0.00	0	0.00	3600.00
3000	50	0	1	9	0	-0.10	-0.01	0	0.00	3600.00
4000	10	9	0	10	0	0.00	0.00	0	0.00	360.13
4000	20	8	0	10	0	0.00	0.00	0	0.00	725.24
4000	30	0	1	9	0	-0.10	-0.01	0	0.00	3600.00
4000	40	0	0	9	1	0.10	0.01	1	0.06	3600.00
4000	50	0	0	10	0	0.00	0.00	0	0.00	3600.00
5000	10	9	0	10	0	0.00	0.00	0	0.00	360.15
5000	20	7	0	10	0	0.00	0.00	0	0.00	1297.09
5000	30	1	5	5	0	-0.50	-0.02	0	0.00	3245.93
5000	40	0	0	9	1	0.10	0.00	1	0.05	3600.00
5000	50	0	3	7	0	-0.30	-0.01	0	0.00	3600.00

Table 6: Comparison with the MIP approach ($M = 4$)

L	N	opt.	$d_{\text{best}} - d_{\text{MIP}}$			average gap		maximum gap		time (s)
			-1	0	1	abs.	%	abs.	%	
1000	10	10	0	10	0	0.00	0.00	0	0.00	0.11
1000	20	10	0	9	1	0.10	0.02	1	0.16	2.04
1000	30	6	0	8	2	0.20	0.03	1	0.15	1450.50
1000	40	1	0	7	3	0.30	0.04	1	0.15	3240.42
1000	50	0	0	5	5	0.50	0.07	1	0.15	3600.00
2000	10	10	0	10	0	0.00	0.00	0	0.00	0.33
2000	20	10	0	10	0	0.00	0.00	0	0.00	2.80
2000	30	9	0	8	2	0.20	0.02	1	0.08	383.46
2000	40	2	0	6	4	0.40	0.03	1	0.08	2881.66
2000	50	0	0	6	4	0.40	0.03	1	0.07	3600.00
3000	10	10	0	10	0	0.00	0.00	0	0.00	0.31
3000	20	10	0	10	0	0.00	0.00	0	0.00	8.90
3000	30	4	0	9	1	0.10	0.01	1	0.05	2271.46
3000	40	0	1	9	0	-0.10	-0.00	0	0.00	3600.00
3000	50	0	0	6	4	0.40	0.02	1	0.05	3600.00
4000	10	10	0	10	0	0.00	0.00	0	0.00	0.70
4000	20	10	0	9	1	0.10	0.00	1	0.04	265.77
4000	30	6	1	9	0	-0.10	-0.00	0	0.00	1717.12
4000	40	0	0	10	0	0.00	0.00	0	0.00	3600.00
4000	50	0	0	8	2	0.20	0.01	1	0.04	3600.00
5000	10	10	0	10	0	0.00	0.00	0	0.00	0.68
5000	20	10	0	8	2	0.20	0.01	1	0.03	204.94
5000	30	7	0	10	0	0.00	0.00	0	0.00	1349.66
5000	40	0	3	7	0	-0.30	-0.01	0	0.00	3600.00
5000	50	0	0	7	3	0.30	0.01	1	0.03	3600.00

Table 7: Comparison with the MIP approach ($M = 4$ and GC-content is 72%)

L	N	opt.	$d_{\text{best}} - d_{\text{MIP}}$			average gap		maximum gap		time (s)
			-1	0	1	abs.	%	abs.	%	
1000	10	10	0	10	0	0.00	0.00	0	0.00	0.07
1000	20	9	0	6	4	0.40	0.07	1	0.18	361.79
1000	30	5	0	6	4	0.40	0.07	1	0.17	1807.24
1000	40	0	0	8	2	0.20	0.03	1	0.17	3600.00
1000	50	0	0	1	9	0.90	0.15	1	0.17	3600.00
2000	10	10	0	10	0	0.00	0.00	0	0.00	0.53
2000	20	10	0	8	2	0.20	0.02	1	0.09	3.83
2000	30	8	0	6	4	0.40	0.03	1	0.09	722.59
2000	40	1	0	5	5	0.50	0.04	1	0.09	3274.13
2000	50	0	0	3	7	0.70	0.06	1	0.08	3600.00
3000	10	10	0	10	0	0.00	0.00	0	0.00	0.36
3000	20	10	0	6	4	0.40	0.02	1	0.06	13.32
3000	30	4	0	9	1	0.10	0.01	1	0.06	2162.45
3000	40	0	0	5	5	0.50	0.03	1	0.06	3600.00
3000	50	0	0	3	7	0.70	0.04	1	0.06	3600.00
4000	10	10	0	10	0	0.00	0.00	0	0.00	0.85
4000	20	10	0	8	2	0.20	0.01	1	0.04	169.21
4000	30	3	0	9	1	0.10	0.00	1	0.04	2629.78
4000	40	0	0	8	2	0.20	0.01	1	0.04	3600.00
4000	50	0	0	2	8	0.80	0.03	1	0.04	3600.00
5000	10	10	0	10	0	0.00	0.00	0	0.00	0.79
5000	20	10	0	7	3	0.30	0.01	1	0.04	203.96
5000	30	6	0	6	4	0.40	0.01	1	0.03	1825.67
5000	40	0	1	7	2	0.10	0.00	1	0.03	3600.00
5000	50	0	0	2	8	0.80	0.03	1	0.03	3600.00

Table 8: Comparison with the MIP approach ($M = 20$)

L	N	opt.	$d_{\text{best}} - d_{\text{MIP}}$			average gap		maximum gap		time (s)
			-1	0	1	abs.	%	abs.	%	
1000	10	10	0	10	0	0.00	0.00	0	0.00	0.10
1000	20	10	0	10	0	0.00	0.00	0	0.00	0.59
1000	30	10	0	10	0	0.00	0.00	0	0.00	2.69
1000	40	9	0	10	0	0.00	0.00	0	0.00	431.81
1000	50	6	2	8	0	-0.20	-0.02	0	0.00	1441.46
2000	10	10	0	10	0	0.00	0.00	0	0.00	0.27
2000	20	10	0	10	0	0.00	0.00	0	0.00	0.81
2000	30	10	0	10	0	0.00	0.00	0	0.00	46.69
2000	40	10	0	10	0	0.00	0.00	0	0.00	34.01
2000	50	9	0	10	0	0.00	0.00	0	0.00	371.31
3000	10	10	0	10	0	0.00	0.00	0	0.00	0.38
3000	20	10	0	10	0	0.00	0.00	0	0.00	1.22
3000	30	10	0	10	0	0.00	0.00	0	0.00	7.12
3000	40	9	1	9	0	-0.10	-0.00	0	0.00	436.00
3000	50	6	2	8	0	-0.20	-0.01	0	0.00	1451.05
4000	10	10	0	10	0	0.00	0.00	0	0.00	0.54
4000	20	10	0	10	0	0.00	0.00	0	0.00	1.79
4000	30	10	0	10	0	0.00	0.00	0	0.00	125.76
4000	40	9	1	9	0	-0.10	-0.00	0	0.00	468.19
4000	50	6	3	7	0	-0.30	-0.01	0	0.00	1447.59
5000	10	10	0	10	0	0.00	0.00	0	0.00	0.74
5000	20	10	0	10	0	0.00	0.00	0	0.00	3.23
5000	30	10	0	10	0	0.00	0.00	0	0.00	44.27
5000	40	8	2	8	0	-0.20	-0.00	0	0.00	735.64
5000	50	8	2	8	0	-0.20	-0.00	0	0.00	879.72

Table 9: Average gap and average CPU time for instances with $M = 2$, $L \in \{300, 400, \ldots, 800\}$ and $N = \{10, 15, 20\}$ in [15]

average gap (%)	average time[a] (s)
7.4	19.3

a: 2.5GHz Pentium4.

Table 10: Average gaps from optimal solutions and average CPU times for instances with $L \in \{1000, 2000, \ldots, 5000\}$ and $N \in \{10, 20, 30\}$ in [12]

type	average gap (%)	average time[b] (s)
$M = 2$	0.5–9.3	0.1–8.0
$M = 4$	1.0–3.1	1.2–72.8
$M = 4$ and GC-content is 72%	0.9–3.2	2.7–61.8
$M = 20$	2.2–6.5	3.1–112.1

b: parallel computing with 28 nodes (Xeon DP).

Table 11: Comparison of the proposed algorithm with the GA in [16] for instances with $L = 500$

M	no.	proposed		GA in [16]	
		objective	time (s)	objective	time[c] (s)
2	1	187	0.16	188	22.5
	2	186	0.14	188	22.5
	3	**190**	0.00	191	22.4
	4	**193**	0.00	194	24.5
	5	**188**	0.00	189	22.4
4	1	**290**	0.00	292	23.0
	2	**290**	0.00	293	23.0
	3	**290**	0.00	292	23.1
	4	**292**	0.00	294	23.0
	5	**292**	0.00	294	22.9
10	1	**362**	0.00	364	21.8
	2	**362**	0.00	363	21.9
	3	**365**	0.00	366	21.8
	4	**366**	0.00	368	21.6
	5	**364**	0.00	365	21.7
20	1	**393**	0.00	395	20.5
	2	**392**	0.00	394	20.6
	3	**393**	0.00	396	20.6
	4	**393**	0.00	395	20.6
	5	**393**	0.00	395	20.6
30	1	**405**	0.00	408	20.0
	2	**404**	0.00	407	20.2
	3	**401**	0.00	403	20.1
	4	**406**	0.00	409	19.9
	5	**404**	0.00	407	20.0

bold: optimal, c: 2.53GHz Pentium4.

Table 12: Results of the Ant Colony Optimization algorithm in [17] for five instances with $M = 4$ and $L = 1000$

N	average objective	standard deviation	average time[d] (s)
10	652	3.72	7.85
20	695	2.49	11.80
30	713	2.29	10.70
40	722	1.91	16.00
50	729	1.68	18.30

d: 1.86GHz Pentium M 750.