Computers & Operations Research 39 (2012) 709–717

Contents lists available at ScienceDirect

Computers & Operations Research

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

A heuristic algorithm based on Lagrangian relaxation for the closest string problem[☆]

Shunji Tanaka [*]

Department of Electrical Engineering, Kyoto University, Kyotodaigaku-Katsura, Nishikyo-ku, Kyoto 615-8510, Japan

ARTICLE INFO

Available online 17 June 2011

Keywords:
Closest string problem
Mixed-integer programming
Lagrangian relaxation
Tabu search

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.

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 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) [2–4] 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 [5–7]. 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 [10–12]. It is true that this method can solve small-sized problems, but it fails when the problem size becomes large.

Studies on (meta)heuristic algorithms have been increasing in these few years [10,12–17]. 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 min 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

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

[*] Tel.: +81 75 383 2204; fax: +81 75 383 2201.
E-mail address: tanaka@kuee.kyoto-u.ac.jp

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.06.005

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 $a_1,\ldots,a_M$ by $\Sigma\ (=\{a_1,\ldots,a_M\})$ and define an index set $\mathcal{M}$ by $\mathcal{M}=\{1,\ldots,M\}$. 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 $s_i\ (i\in\mathcal{N}=\{1,\ldots,N\})$ 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 $s_i\ (i\in\mathcal{N})$. Here, the Hamming distance $d_{\mathrm{H}}(s_i,s)$ between s_i and s is defined by

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

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

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

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

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

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

Its cardinality is denoted by $m^j=|\mathcal{A}^j|\le M$ and index sets $\mathcal{M}^j$ are defined by $\mathcal{M}^j=\{1,\ldots,m^j\}$ for $j\in\mathcal{L}$. Let us also define ν_i^j $(i\in\mathcal{N}, j\in\mathcal{L})$ so that the ν_i^jth element of $\mathcal{A}^j$ is equal to $s_i[j]$. Next, we introduce decision variables $x_{kj}\ (k\in\mathcal{M}^j, j\in\mathcal{L})$ that are defined by

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

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

$$d^*=\min_{d,\boldsymbol{x}} \quad d, \tag{6}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{M}^j} x_{kj}=1, \quad j\in\mathcal{L}, \tag{7}$$

$$d+\sum_{j\in\mathcal{L}} x_{\nu_i^j j}\ge L, \quad i\in\mathcal{N}, \tag{8}$$

$$d\ge 0, \tag{9}$$

$$x_{kj}\in\{0,1\}, \quad k\in\mathcal{M}^j,\ j\in\mathcal{L}. \tag{10}$$

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

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

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 $\mu_i\ge 0\ (i\in\mathcal{N})$ and the following Lagrangian relaxation (LR) is obtained:

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

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

$$\hat{d}^*(\boldsymbol{\mu})\le d^*. \tag{13}$$

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

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

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

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

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

$$\hat{d}_j^*(\boldsymbol{\mu})=-\max_{\boldsymbol{x}_j}\quad\sum_{i\in\mathcal{N}}\mu_i x_{\nu_i^j j}=-\max_{\boldsymbol{x}_j}\quad\sum_{k\in\mathcal{M}^j}\left(\sum_{\substack{i\in\mathcal{N}\\ \nu_i^j=k}}\mu_i\right)x_{kj}, \tag{16}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{M}^j}x_{kj}=1, \tag{17}$$

$$x_{kj}\in\{0,1\}, \quad k\in\mathcal{M}^j. \tag{18}$$

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

$$\hat{d}^*(\boldsymbol{\mu})=\sum_{j\in\mathcal{L}}\hat{d}_j^*(\boldsymbol{\mu})+L\sum_{i\in\mathcal{N}}\mu_i. \tag{19}$$

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

$$\hat{d}_j^*(\boldsymbol{\mu})=-\max_{k\in\mathcal{M}^j}\sum_{\substack{i\in\mathcal{N}\\ \nu_i^j=k}}\mu_i. \tag{20}$$