Computers and Operations Research 80 (2017) 94–100

Contents lists available at ScienceDirect

Computers and Operations Research

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

An improved integer linear programming formulation for the closest 0-1 string problem

Claudio Arbib[a], Mara Servilio[b,*], Paolo Ventura[b]

[a] Dipartimento di Scienze/Ingegneria dell'Informazione e Matematica, Università degli Studi dell'Aquila via Vetoio, Coppito, I-67010 L'Aquila, Italy
[b] Istituto di Analisi dei Sistemi e Informatica "A. Ruberti" CNR, Via dei Taurini, 19 - 00185, Roma, Italy

ARTICLE INFO

Article history:
Received 30 December 2015
Revised 18 November 2016
Accepted 19 November 2016
Available online 19 November 2016

Keywords:
Closest string problem
Branch-and-cut
Continuous relaxation

ABSTRACT

The Closest String Problem (CSP) calls for finding an n-string that minimizes its maximum Hamming distance from m given n-strings. Recently, integer linear programs (ILP) have been successfully applied within heuristics to improve efficiency and effectiveness. We consider an ILP for the binary case (0-1 CSP) that updates the previous formulations and solve it by branch-and-cut. The method separates in polynomial time the first closure of $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts and can either be used stand-alone to find optimal solutions, or as a plug-in to improve heuristics based on the exact solution of reduced problems. Due to the parity structure of the right-hand side, the impressive performances obtained with this method in the binary case cannot be directly replicated in the general case.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

Let A be an alphabet with p symbols. The Closest String—or Center String—Problem (CSP) calls for finding a string $\mathbf{x} \in A^n$ that better approximates a given set S of strings $\mathbf{s}^1, \dots, \mathbf{s}^m \in A^n$. Approximation is measured with the Hamming distance $d(\mathbf{x}, \mathbf{y})$, that counts the number of different components in $\mathbf{x}, \mathbf{y}$. An optimal solution of the CSP is an $\mathbf{x}^*$ that, among all strings $\mathbf{x} \in A^n$, minimizes the maximum distance $d(\mathbf{x}, \mathbf{s}^i)$ from any $\mathbf{s}^i \in S$.

The CSP arises in such fields as computational biology and coding theory, and is NP-hard. The alphabet A can contain two or more symbols, depending on application: for example, $p = 2$ in encoding problems, $p = 4$ in DNA recognition etc. In the former case we refer to binary (or 0-1) CSP.

Due to its importance, the problem has recently attracted extensive research, see e.g. [5,8–10]. Various integer linear programming (ILP) formulations have also been proposed to solve it, see [1,6,7], and ILP is a key factor of success for the present state-of-the-art heuristics [2,3]. Therefore, improving the performance of ILP formulations for the CSP is a way to improve the performance of those algorithms.

In this paper we focus on the binary CSP. We revise the formulation in [1] and strengthen the polyhedron Q of its continuous relaxation by the first closure of $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts (in short, $\{0, \frac{1}{2}\}$-CG cuts). We prove that when the polyhedron Q' of the first closure is defined by the inequalities of our formulation, the points in $Q - Q'$ can be separated in polynomial-time. We also point out that, with the formulation here considered, Q' has different properties in the general and in the binary case: in the former, we observe that $Q = Q'$, thus separating over Q' is pointless; on the contrary, the cuts in Q' are generally very effective in the binary case. Based on this analysis, we develop a branch-and-cut algorithm for the binary CSP, and test it on instances from [3]. The cuts in the first closure are often sufficient to get an impressive speed-up of CPU time.

2. An integer linear programming formulation for the general CSP

Let d denote the largest Hamming distance of the desired string $\mathbf{x}$ from a string in the target set S. In the so-called "natural" formulation [1], $\mathbf{x}$ is encoded by a matrix $\mathbf{Y} \in \{0, 1\}^{p \times n}$ with exactly one 1 per column: the entries $y_{\alpha k}$ of $\mathbf{Y}$ are binary decision variables that assign a symbol $\alpha \in A$ to each component x_k of $\mathbf{x}$. The problem is formulated as follows:

$$\min \quad d \tag{1}$$

$$d + \sum_{k=1}^{n} y_{s^i_k k} \geq n \quad i = 1, \dots, m \tag{2}$$

$$\sum_{\alpha \in A} y_{\alpha k} = 1 \quad k = 1, \dots, n \tag{3}$$

* Corresponding author. Mara Servilio
E-mail addresses: claudio.arbib@univaq.it (C. Arbib), mara.servilio@iasi.cnr.it (M. Servilio), paolo.ventura@iasi.cnr.it (P. Ventura).

http://dx.doi.org/10.1016/j.cor.2016.11.019
0305-0548/© 2016 Elsevier Ltd. All rights reserved.

$$y_{\alpha k} \geq 0 \tag{4}$$

$$-y_{\alpha k} \geq -1 \tag{5}$$

$$y_{\alpha k} \quad \text{integer} \quad \alpha \in A, k = 1, \dots, n$$

In the ith constraint (2), s^i_k is the symbol of A occurring at the kth component of $\mathbf{s}^i$: hence the summation on the left-hand side counts the bits of $\mathbf{x}$ that are equal to the corresponding bits of $\mathbf{s}^i$. The distance between $\mathbf{x}$ and $\mathbf{s}^i$ is therefore the complement of this summation to n. For instance, for $A = \{a, b, c, d\}$, $n = 5$ and $\mathbf{s}^i = abbcd$, inequality (2) reads

$$d + y_{a1} + y_{b2} + y_{b3} + y_{c4} + y_{d5} \geq 5$$

or, eliminating y_{d5} by (3) as in [1],

$$d + y_{a1} + y_{b2} + y_{b3} + y_{c4} - y_{a5} - y_{b5} - y_{c5} \geq 4$$

The nonzero support of these inequalities does not seem to have a combinatorial structure that can be exploited to efficiently separate $\{0, \frac{1}{2}\}$-CG cuts. Then we suggest here a "dense" formulation where such cuts can easily be separated. To this aim, we encode a generic string $\mathbf{s}^i$ in the same way as the $\mathbf{x}$, setting $s^i_{\alpha k} = 1$ if $s^i_k = \alpha$ and 0 otherwise, for any $\alpha \in A$. The following expression

$$f^i_\alpha(x_k) = (y_{\alpha k} - s^i_{\alpha k})^2 = y_{\alpha k} - 2s^i_{\alpha k} y_{\alpha k} + s^i_{\alpha k}$$

gets value 0 if $y_{\alpha k} = s^i_{\alpha k}$ (that is, $x_k = s^i_k$) and 1 otherwise. In the latter case, $y_{\alpha k}$ differs from $s^i_{\alpha k}$ in exactly two cases; therefore

$$f^i(x_k) = \sum_{\alpha \in A} f_\alpha(x_k) = \sum_{\alpha \in A} (y_{\alpha k} - 2s^i_{\alpha k} y_{\alpha k} + s^i_{\alpha k})$$

gets value 0 for $x_k = s^i_k$ and 2 otherwise. Consequently

$$2d(\mathbf{x}, \mathbf{s}^i) = \sum_{k=1}^n f^i(x_k) = \sum_{k=1}^n \sum_{\alpha \in A} (y_{\alpha k} - 2s^i_{\alpha k} y_{\alpha k} + s^i_{\alpha k})$$

Using the expression above and observing that

$$\sum_{k=1}^n \sum_{\alpha \in A} s^i_{ak} = n$$

we can replace (2) by

$$2d + \sum_{k=1}^n \sum_{\alpha \in A} (2s^i_{\alpha k} - 1) y_{\alpha k} \geq n \quad i = 1, \dots, m \tag{6}$$

Note that the coefficient $(2s^i_{\alpha k} - 1)$ of any variable $y_{\alpha k}$ in inequality (6) is ± 1: therefore we refer to (6) as to *dense inequalities*. Because of hyperplanes (3), the polyhedron (3)–(6) has dimension $(p-1)n + 1$.

3. Reformulation for the binary case

Assuming $A = \{0, 1\}$, the components of $\mathbf{Y}$ and $\mathbf{S}^i$ of Section 2 become

$$y_{1k} = x_k, \quad s^i_{1k} = s^i_k, \quad y_{0k} = 1 - x_k, \quad s^i_{0k} = 1 - s^i_k$$

where complementation derives from the assignment equations (3). The Hamming distance between $\mathbf{x}$ and $\mathbf{s}^i$ is then directly expressed by

$$d(\mathbf{x}, \mathbf{s}^i) = \sum_{k=1}^n [s^i_k(1 - x_k) + (1 - s^i_k) x_k]$$

with $\mathbf{x} \in \{0, 1\}^n$. Therefore, a string $\mathbf{x}$ whose distance from any $\mathbf{s}^i$ is at most d must fulfill

$$x(N^i_0) - x(N^i_1) = \sum_{k \in N^i_0} x_k - \sum_{k \in N^i_1} x_k \leq d - \sum_{k=1}^n s^i_k$$

where N^i_0 and N^i_1 denote the set of indexes k such that $s^i_k = 0$ and $s^i_k = 1$, respectively. Rewriting the above condition with $d = 2\delta$ and $n^i = \sum_{k=1}^n s^i_k$, we get our formulation:

$$\min \quad \delta \tag{7}$$

$$2\delta - \sum_{k \in N^i_0} x_k + \sum_{k \in N^i_1} x_k \geq n^i \qquad i = 1, \dots, m \tag{8}$$

$$x_k \geq 0 \tag{9}$$

$$-x_k \geq -1 \tag{10}$$

$$x_k \quad \text{integer} \quad k = 1, \dots, n$$

Just like (6), inequalities (8) have the x coefficients in $\{-1, +1\}$ and are again called *dense*. We distinguish between *odd* and *even* dense inequalities according to the parity of the right-hand side n^i. Note that in the non-binary CSP, all dense inequalities have the same parity (in fact, in this case n^i is always equal to n). In the binary CSP, instead, the parity of the right-hand sides is instance-dependent. In the test bed used for our computational experiments we observed odd and even n^i's quite randomly distributed. This fact plays a crucial role in the strength of the method here proposed, as we will see next.

4. $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts for the binary case

Unlike the general Chvátal-Gomory cuts, $\{0, \frac{1}{2}\}$-CG cuts are not derived from the polyhedron Q obtained by linearly relaxing the integer formulation but from the particular system of linear inequalities used to describe Q. In general, let $\mathcal{S}$ denote the system of linear inequalities of an ILP formulation:

$$\mathcal{S} = \{a^i y \geq b^i \text{ with } a^i \in \mathbb{Z}^m \text{ and } b^i \in \mathbb{Z} \text{ for all } i \in I\},$$

and define the feasible set and its linear relaxation, respectively, as

$$P = \{y \in \mathbb{Z}^n : y \text{ satisfies } \mathcal{S}\} \quad Q = \{y \in \mathbb{R}^n : y \text{ satisfies } \mathcal{S}\}$$

A $\{0, \frac{1}{2}\}$-CG cut for P is obtained by combining inequalities in $\mathcal{S}$ with multipliers that are either 0 or $\frac{1}{2}$, so that the coefficients at the left-hand side are integer and the right-hand side is not. In this way, one can round the right-hand side up to the closest integer, and get an inequality which is valid for P and not for Q. Equivalently, a $\{0, \frac{1}{2}\}$-CG cut $ax \geq b$ can be derived from a linear combination of $a^i\mathbf{x} \geq b^i$ with $\lambda^i \in \{0, 1\}$ such that

$$a_j = \sum_{i \in I} \lambda^i a^i_j \text{ is even for } j = 1, \dots, n \quad b = \sum_{i \in I} \lambda^i b^i \text{ is odd} \tag{11}$$

Let $\mathcal{S}'$ contain all the $\{0, \frac{1}{2}\}$-CG cuts that can be derived from the inequalities of $\mathcal{S}$. Such a system is called the *first* $\{0, \frac{1}{2}\}$*-CG closure* of $\mathcal{S}$.

Take $\bar{\mathbf{y}} \in Q$, and consider the problem of separating $\bar{\mathbf{y}}$ with a cut in $\mathcal{S}'$, that is, finding an inequality of $\mathcal{S}'$ that is violated by $\bar{\mathbf{y}}$, or conclude that $\mathcal{S}'$ does not contain such an inequality. The problem can be rephrased as follows:

Problem 1. Find $\lambda^i \in \{0, 1\}$ fulfilling (11) and such that

$$viol(\lambda, \bar{\mathbf{y}}) = -\sum_{j=1}^n \left(\frac{1}{2} \sum_{i \in I} \lambda^i a^i_j \right) \bar{y}_j + \left\lceil \frac{\sum_{i \in I} \lambda^i b^i}{2} \right\rceil > 0.$$

Rewrite the violation as

$$viol(\lambda, \bar{\mathbf{y}}) = -\frac{1}{2} \sum_{i \in I} \lambda^i \sum_{j=1}^n a^i_j \bar{y}_j + \frac{1}{2} \left(\sum_{i \in I} \lambda^i b^i + 1 \right)$$

Download English Version:

https://daneshyari.com/en/article/4959093

Download Persian Version:

https://daneshyari.com/article/4959093

Daneshyari.com