PLANOS DE CORTE E HEURÍSTICAS PARA O *CLOSEST STRING PROBLEM*

Omar Latorre Vilca
Universidade Federal do ABC (UFABC)
Rua Santa Adélia, 166. Bairro Bangu. Santo André - SP - Brasil
omar.vilca@ufabc.edu.br

Cláudio Nogueira de Meneses
Universidade Federal do ABC (UFABC)
Rua Santa Adélia, 166. Bairro Bangu. Santo André - SP - Brasil
claudio.meneses@ufabc.edu.br

RESUMO

Neste artigo consideramos o *Closest String Problem* (CSP). Baseando-se em uma formulação em programação linear inteira para o CSP, mostramos uma nova classe de cortes e suas separações. Resultados computacionais são reportados.

Palavras-chave: programação linear inteira, planos de corte, algoritmo de separação.

Área principal: PM - Programação Matemática, OC - Otimização Combinatória

ABSTRACT

In this paper we consider the Closest String Problem (CSP). Based on an integer linear programming formulation for the CSP we show a new class of cuts and their separation. Computational results are reported.

Keywords: integer linear programming, cutting planes, separation algorithm.

Main area: PM - Mathematical Programming, OC - Combinatorial Optimization

1 Introdução

Neste artigo consideramos o problema de otimização combinatória chamado *Closest String Problem* (CSP). Dadas duas *strings*, s e t, de igual tamanho (número de caracteres da *string*), a distância de Hamming, denotada por $d_H(s,t)$, mede o número de posições em que s e t diferem. Por exemplo, se $s = CCACT$ e $t = TACCA$, então $d_H(s,t) = 4$. O CSP consiste em: dado um conjunto finito $\mathbf{S} = \{s^1, s^2, \cdots, s^n\}$ com n *strings*, todas de mesmo tamanho m, sobre um alfabeto $\mathbf{A}$, deseja-se encontrar uma *string* x, de tamanho m, sobre $\mathbf{A}$ e minimizando o valor de d tal que para toda *string* $s^i \in \mathbf{S}$ tem-se $d_H(x, s^i) \leq d$.

O CSP tem sido bastante estudado nos últimos anos devido às suas aplicações, conforme descrito em Stormo, G. e Hartzell III, G. (1991), Hertz, G. e Stormo, G. (1995), Gasieniec, L. et al. (1999), Rajasekaran, S. et al. (2001a), Rajasekaran, S. et al. (2001b) e Lanctot, K. et al. (2003). Em termos de complexidade computacional, o CSP pertence a classe NP-difícil (ver referência Lanctot, K. et al. (2003)). Quanto aos métodos para resolver instâncias do problema, foram propostas heurísticas (Gomes, F.C. et al. (2008)), algoritmos de aproximação (Ben-Dor, A. (1997), Li, M. et al. (2002), Lanctot, K. et al. (2003)) e algoritmos exatos. Os métodos exatos foram baseados em programação linear inteira (Meneses, C.N. et al. (2004), Meneses, C. N. et al. (2005)) e na teoria de *fixed-parameter* (Fellows, M.R. et al. (2002), Gramm, J. et al. (2003)).

Neste artigo apresentamos uma classe de planos de corte para o CSP. O restante do artigo está organizado da seguinte maneira. A Seção 2 faz uma pequena revisão sobre programação linear inteira e métodos comumente usados para resolvê-la. A Seção 3 mostra uma formulação em programação linear inteira para o CSP, que aparece na referência Meneses, C.N. et al. (2004). A Seção 4 apresenta uma nova classe de inequações válidas para o CSP e como separá-las. Na Seção 5 são discutidos os experimentos computacionais. Finalmente, na Seção 6 as conclusões são apresentadas.

2 Programação Linear Inteira (PLI)

Nesta seção discutimos alguns métodos normalmente usados na resolução de problemas, que admitem uma formulação em programação linear que exigem soluções inteiras.

Os conceitos discutidos nesta seção foram compilados de Taha, H. A. (1987), Nemhauser, G. L. e Wolsey, L. A. (1988) e de Souza, C. C. (1993).

Considere o problema de programação linear abaixo:

$$min\{cx : Ax \leq b, x \in \Re^n_+\},$$

onde $c \in \Re^n$, A é uma matriz $m \times n$ e $b \in \Re^m$.

Se as variáveis forem inteiras ($x \in \mathbf{Z}^n_+$, ao invés de $x \in \Re^n_+$), o problema é chamado de problema de *Programação Linear Inteira* (PLI). Além disso, se as variáveis são restritas a valores 0 ou 1, temos um problema PLI 0-1. As soluções do problema PLI 0-1 são pontos (0-1) satisfazendo o sistema linear $Ax \leq b$.

Geralmente problemas PLI 0-1 pertencem a classe de complexidade *NP-difícil*. Uma maneira de atacar estes problemas é resolver suas relaxações lineares. Numa relaxação linear as restrições de integralidade são substituídas por restrições lineares. Existem duas abordagens clássicas para resolver problemas PLI 0-1, uma é usando relaxações lineares: Algoritmo de Planos de Corte Fracionário e o algoritmo de *branch-and-bound*, a outra é a enumeração implícita.

2.1 Algoritmo de Planos de Corte Fracionário

Seja **P** o conjunto das soluções viáveis do problema PLI 0-1. Algoritmo de Planos de Corte Fracionário (APCF) é baseado no uso de desigualdades válidas (cortes) para **P**, ou seja, desigualdades que são satisfeitas por todos os pontos de **P**.

A cada iteração i do APCF, uma relaxação LP^i do problema *PLI* é resolvida. Seja x^i uma solução ótima obtida ao se resolver a relaxação linear LP^i, se x^i está em **P** o algoritmo termina retornando x^i como uma solução ótima do problema *PLI*, caso contrário, a relaxação deve ser melhorada. Para isto, encontra-se uma desigualdade válida, $\pi x \leq \pi_0$, para **P** que é violada por x^i e realiza-se uma nova iteração para a relaxação LP^{i+1} qué obtida de LP^i incluindo-se a desigualdade $\pi x \leq \pi_0$.

Sejam z^i e z^{i+1} os valores das soluções ótimas de LP^i e LP^{i+1} respectivamente, isto é, $z^i = cx^i$ e $z^i = cx^{i+1}$. Assumindo que o *PLI* é um problema de minimização, tem-se que $z^{i+1} \geq z^i$, ou seja, o limite inferior fornecido pelo valor ótimo da relaxação linear cresce monotonicamente a cada iteração, aproximando-se do valor ótimo do *PLI*.

3 Formulação em PLI 0-1 para o *Closest String Problem*

Nesta seção relembramos a formulação e o teorema que aparecem em Meneses, C.N. et al. (2004). O teorema serve para diminuir o espaço das soluções viáveis, onde se encontram as soluções ótimas para as instâncias do CSP.

Teorema 1 (Meneses, C.N. et al. (2004)). Dada uma instância do *Closest String Problem*, existe uma solução ótima onde o caracter ótimo na posição k está também na posição k em uma das *strings* no conjunto $\mathbf{S} = \{s^1, s^2, \ldots, s^n\}$ de *strings*. □

Defina $V_j = \cup_{i=1}^{n} s_j^i \quad$ para $j = 1, \ldots, m$.

Exemplo 1. Assuma $\mathbf{S} = \{$AATCC,CCAAT,CCTAC,TCACC$\}$. Então os conjuntos V_j são: $V_1 = \{A, C, T\}, V_2 = \{A, C\}, V_3 = \{A, T\}, V_4 = \{A, C\}, V_5 = \{C, T\}$. O Teorema 1 garante que para encontrar uma solução ótima $x = (x_1, x_2, x_3, x_4, x_5)$ é suficiente atribuir a x_j um elemento do conjunto V_j, para $j = 1, \ldots, 5$. □

Um modelo matemático em programação linear inteira binária é como segue. Primeiro definimos as variáveis do problema:

$$x_{j,k} = \begin{cases} 1 & \text{se o caracter } j \text{ é usado na posição } k \text{ em uma solução} \\ 0 & \text{caso contrário} \end{cases}$$

A formulação é a seguinte:

$$\begin{array}{lll} \min & d & \\ s.a.: & \sum_{j \in V_k} x_{j,k} = 1 & k = 1, \ldots, m \\ & d \geq m - \sum_{j=1}^{m} x_{s_j^i, j} & i = 1, \ldots, n \\ & x_{j,k} \in \{0, 1\} & j \in V_k; k = 1, \ldots, m \\ & d \geq 0 \text{ e inteiro} & \end{array}$$

A formulação tem $m + n$ restrições e $1 + \sum_{k=1}^{m} |V_k|$ variáveis. Em Meneses, C.N. et al. (2004) é demonstrado, de forma experimental, que esta formulação é muito forte, pois costuma fornecer excelentes limites inferiores nos valores de soluções ótimas inteiras.

4 Uma Classe de Cortes e suas Separações

Assuma que temos uma solução viável, de valor D, para uma instância do CSP. Possivelmente, uma boa solução viável assim que D é provavelmente ótimo. Esta boa solução viável poderia, por exemplo, ser obtida usando um procedimento de arredondamento, a partir da solução obtida pela relaxação linear do modelo apresentado na seção anterior.

Seja $\mathbf{S} = \{s^1, s^2, ..., s^n\}$, com $|s^i| = m$ para $i = 1, \ldots, n$. Tome qualquer *string* $s^i \in \mathbf{S}$ e considere qualquer subconjunto B de s^i consistindo de D caracteres. Comparando as correspondentes posições de B em s^i, temos que uma solução ótima para $\mathbf{S}$ não pode ser diferente em todos os caracteres em B, caso contrário esta solução teria custo $\geq D$. Como já temos uma solução de valor D, então procuramos por uma solução de valor $\leq D-1$. Portanto, concluímos que

$$\sum_{j \in Ind(B)} x_{s^i[j],j} \geq 1 \tag{1}$$

é uma inequação válida (corte), onde $Ind(B)$ é o conjunto das posições de B em s^i.

Exemplo 2. Seja $\mathbf{S} = \{s^1, s^2, s^3\}$, onde $s^1 = ACT$,$s^2 = CCG$ e $s^3 = TCA$. Considerando o resultado no Teorema 1, temos que uma solução ótima $x = (x_1, x_2, x_3)$ para $\mathbf{S}$ satisfaz $x_1 \in V_1 = \{A, C, T\}$, $x_2 \in V_2 = \{C\}$ e $x_3 \in V_3 = \{A, G, T\}$. Uma solução viável é $x = ACG$ com valor 2. Para $s^1 = ACT$ as subsequências de comprimento dois são AC, AT, CT; implicando que $Ind(AC) = \{1,2\}$, $Ind(AT) = \{1,3\}$ e $Ind(CT) = \{2,3\}$. Assim, temos os cortes para $s^1 = ACT$:

$$x_{A,1} + x_{C,2} \geq 1$$
$$x_{A,1} + x_{T,3} \geq 1$$
$$x_{C,2} + x_{T,3} \geq 1$$

□

Existem $n \times \binom{m}{D}$ tais inequações possíveis (isto é um número exponencial, visto que D pode ser proporcional a m). Agora mostramos como encontrar um corte violado (se ele existe) em tempo polinomial. Suponha que x^* é uma solução fracionária ótima obtida pela relaxação linear.

Considere cada *string* $s^i \in \mathbf{S}$ por vez. Para $j = 1, ..., m$ defina

$$a_j = x^*_{s^i[j],j}$$

Por exemplo, para s^1 temos

$$a_1 = x^*_{s^1[1],1}, a_2 = x^*_{s^1[2],2}, a_3 = x^*_{s^1[3],3}, \ldots, a_{m-1} = x^*_{s^1[m-1],m-1}, a_m = x^*_{s^1[m],m}.$$

Considerando o Exemplo 2, temos para $s^1 = ACT$: $a_1 = x^*_{A,1}$, $a_2 = x^*_{C,2}$, $a_3 = x^*_{T,3}$.

Agora, ordene os a_j em ordem não decrescente, $a_{p(1)} \leq ... \leq a_{p(m)}$, e seja B o conjunto dos primeiros D valores nesta ordem dos a_j (isto é, $B = \{p(1), ..., p(D)\}$). Então B alcança a mínima soma possível de x^* com relação a s^i. Se esta soma for < 1, então encontramos uma inequação (1) violada; caso contrário não há inequações violadas para s^i, e então passamos para a análise de s^{i+1}. Ou seja, se $\sum_{k=1}^{D} a_{p(k)} < 1$ então a inequação $\sum_{k=1}^{D} x_{s^i[p(k)],p(k)} \geq 1$ precisa ser incluída no modelo linear.

Assim provamos o seguinte teorema:

Teorema 2. As inequações (1) podem ser separadas em tempo polinomial (nominalmente em $O(nmlogm)$).

Exemplo 3. Seja $\mathbf{S} = \{ATTGGA, CTGATG, CTGACT, AGTCGA, GCCTGT\}$. Então $V_1 = \{A,C,G\}$, $V_2 = \{C,G,T\}$, $V_3 = \{C,G,T\}$, $V_4 = \{A,C,G,T\}$, $v_5 = \{C,G,T\}$,$v_6 = \{C,G,T\}$. O modelo em programação linear é como segue:

$$\begin{aligned}
\min \quad & d \\
s.a.: \quad & x_{A,1} + x_{C,1} + x_{G,1} = 1 \\
& x_{C,2} + x_{G,2} + x_{T,2} = 1 \\
& x_{C,3} + x_{G,3} + x_{T,3} = 1 \\
& x_{A,4} + x_{C,4} + x_{G,4} + x_{T,4} = 1 \\
& x_{C,5} + x_{G,5} + x_{T,5} = 1 \\
& x_{A,6} + x_{G,6} + x_{T,6} = 1 \\
& d + x_{A,1} + x_{T,2} + x_{T,3} + x_{G,4} + x_{G,5} + x_{A,6} \geq 6 \\
& d + x_{C,1} + x_{T,2} + x_{G,3} + x_{A,4} + x_{T,5} + x_{G,6} \geq 6 \\
& d + x_{C,1} + x_{T,2} + x_{G,3} + x_{A,4} + x_{C,5} + x_{T,6} \geq 6 \\
& d + x_{A,1} + x_{G,2} + x_{T,3} + x_{C,4} + x_{G,5} + x_{A,6} \geq 6 \\
& d + x_{G,1} + x_{C,2} + x_{C,3} + x_{T,4} + x_{G,5} + x_{T,6} \geq 6 \\
& \text{todas as variáveis são} \geq 0
\end{aligned}$$

Uma solução ótima para o programa linear acima é: $d = 3.666667, x_{A,1} = 0.75, x_{C,1} = 0.25$, $x_{T,2} = 0.583333, x_{G,2} = 0.416667$, $x_{G,3} = 1.0, x_{C,4} = 0.166667, x_{T,4} = 0.833333$, $x_{G,5} = 1.0, x_{G,6} = 0.5, x_{T,6} = 0.5$ e todas as outras variáveis têm valores iguais a zero. Desta solução concluímos que $d \geq 3.666667$ e inteiro. Assim, $d \geq 4$. Adicionando o plano de corte $d \geq 4$ ao programa linear e resolvendo-o novamente, obtemos a solução: $d = 4.0$, $x_{A,1} = 0.5$, $x_{C,1} = 0.5$, $x_{T,2} = 0.5$, $x_{G,2} = 0.5$, $x_{G,3} = 1.0$, $x_{T,4} = 1.0$, $x_{G,5} = 1.0$, $x_{G,6} = 1.0$ e todas as outras variáveis têm valores iguais a zero.

Consideramos agora a classe de corte da inequação (1): para $D = 4$ as seguintes inequações são violadas:

$$x_{A,1} + x_{T,3} + x_{G,4} + x_{A,6} \geq 1 \tag{2}$$
$$x_{C,1} + x_{A,4} + x_{C,5} + x_{T,6} \geq 1 \tag{3}$$
$$x_{A,1} + x_{T,3} + x_{C,4} + x_{A,6} \geq 1 \tag{4}$$
$$x_{G,1} + x_{C,2} + x_{C,3} + x_{T,6} \geq 1 \tag{5}$$

As inequações (2), (3), (4) e (5) foram determinadas a partir de s^1, s^3, s^4 e s^5, respectivamente. Incluindo estas inequações no programa linear, junto com $d \geq 4$, e resolvendo o programa linear novamente obtemos uma solução ótima inteira dada por $x = CTCCGG$. □

5 Experimentos Computacionais

Nesta seção descrevemos os resultados computacionais obtidos com as nossas implementações.

5.1 Ambiente dos Experimentos

Todas as implementações foram feitas em C++ e os testes foram realizados em um computador com a seguinte configuração: Dell processador Intel Core I5 3.33 Ghz com 4 GB de memoria RAM e

Sistema Operacional Linux Ubuntu 11.04, 32 bits. As soluções ótimas das instâncias testadas foram obtidas por meio do *software* IBM ILOG CPLEX versão 12.4.

5.2 Instâncias de Teste

Vinte e cinco instâncias foram geradas utilizando o gerador de instâncias discutido na referência Meneses, C.N. et al. (2004).

5.3 Resultados Computacionais

Os resultados obtidos com as implementações são apresentados nas Tabelas 1 e 2.

O cabeçalho na Tabela 1 possui os seguintes significados: a primeira coluna indica a instância testada, com a indicação dos parâmetros (n) número de *strings*, (m) tamanho da *string*; a coluna (PLI) indica os valores das soluções ótimas obtidos usando Programação Linear Inteira; depois temos as relaxações lineares (PL) do modelo linear e (PC) os valores das soluções após a inserção dos planos de corte discutidos na seção anterior; e por fim os valores obtidos usando uma heurística (Arred.) de Arredondamento baseada nas soluções das relaxações lineares, (Heur.) Heurística apresentada em Meneses, C.N. et al. (2004), e (Heur. e Arred.) Heurística e Arredondamento em uma só heurística.

Na Tabela 2, a coluna Num. PC representa o número de planos de corte inseridos no modelo linear.

Dos resultados mostrados nas Tabelas 1 e 2 concluímos que:

(a) O modelo linear fornece excelentes limites inferiores nos valores de soluções ótimas;

(b) Com a inserção dos planos de corte, os valores das relaxações se tornaram iguais aos valores das soluções ótimias em 24 das 25 instâncias testadas. A exceção ocorreu para a instância com $n = 25$ e $m = 100$;

(c) A heurística de arredondamento forneceu excelentes limites superiores e estes foram quase sempre melhorados com o uso da heurística apresentada em Meneses, C.N. et al. (2004).

(d) Devido ao modelo linear ser muito forte, todas as instâncias testadas puderam ser resolvidas em pouco tempo de computação (isto é, em menos de um minuto);

(e) As heurísticas executaram extremamente rápido (isto é, em menos de 5 segundos).

6 Conclusões

Este artigo considera um importante problema de otimização combinatória chamado *Closest String Problem*. É discutido um modelo de programação linear conhecido na literatura. A partir deste modelo apresentamos uma classe de inequações válidas (cortes) e um algoritmo de separação destes cortes. Provamos que esta classe de cortes pode ter um número exponencial de inequações e mostramos que o algoritmo de separação destes cortes tem complexidade polinomial. Foram testadas várias instâncias do problema. Os resultados demonstraram que o modelo linear fornece excelentes limites inferiores e os cortes são bastante úteis para acelerar a resolução do problema de maneira exata.

Instância		Solução Exata	Programa Linear		Heurísticas		
n	m	PLI	PL	PC	Arred.	Heur.	Heur. e Arred.
10	100	59	58,32	59	60	60	60
10	200	119	118,20	119	119	121	119
10	300	173	172,37	173	173	176	173
10	400	235	234,10	235	235	238	235
10	500	289	288,20	289	289	293	289
15	100	61	60,25	61	62	62	62
15	200	124	123,05	124	124	126	124
15	300	185	184,65	185	186	189	186
15	400	246	245,12	246	246	250	246
15	500	305	304,86	305	306	311	306
20	100	64	63,50	64	66	66	65
20	200	127	126,55	127	128	130	128
20	300	191	190,69	191	193	195	192
20	400	254	253,50	254	255	259	255
20	500	317	316,17	317	319	323	319
25	100	66	64,87	65	67	67	67
25	200	130	129,02	130	131	133	131
25	300	193	192,51	193	194	198	194
25	400	258	257,96	258	260	264	260
25	500	322	321,24	322	323	329	323
30	100	67	66,08	67	68	68	68
30	200	131	130,86	131	133	135	133
30	300	197	196,70	197	198	201	198
30	400	265	264,03	265	266	270	266
30	500	329	328,43	329	331	337	331

Tabela 1: Comparação entre os valores das soluções.

Instância		Solução Exata	Programa Linear			Heurísticas		
n	m	PLI(s)	PL(s)	PC(s)	Num. PC	Arred.(s)	Heur.(s)	Heur. e Arred.(s)
10	100	1,2	< 1	1	1	< 1	< 1	< 1
10	200	2,4	< 1	2	1	< 1	< 1	< 1
10	300	4,8	< 1	3	1	< 1	< 1	< 1
10	400	6,5	< 1	5	1	< 1	1	< 1
10	500	6,1	< 1	6	1	< 1	1	< 1
15	100	1,3	< 1	1	1	< 1	< 1	< 1
15	200	7,7	< 1	3	1	< 1	< 1	< 1
15	300	6,6	< 1	5	1	< 1	1	< 1
15	400	18,6	< 1	7	1	< 1	1	1
15	500	32,3	< 1	9	3	< 1	2	1
20	100	3,2	< 1	2	1	< 1	< 1	< 1
20	200	9,1	< 1	5	1	< 1	1	< 1
20	300	18,6	< 1	7	1	< 1	1	< 1
20	400	16,8	< 1	10	1	< 1	2	1
20	500	33	< 1	12	1	< 1	3	1
25	100	2,5	< 1	3	1	< 1	< 1	< 1
25	200	6,6	< 1	6	1	< 1	1	< 1
25	300	12,2	< 1	9	1	< 1	2	1
25	400	13	< 1	12	1	< 1	3	1
25	500	17	< 1	14	1	< 1	3	2
30	100	17	< 1	4	1	< 1	< 1	< 1
30	200	34,8	< 1	7	1	< 1	1	1
30	300	33,2	< 1	11	1	< 1	2	1
30	400	46,1	< 1	13	1	< 1	3	2
30	500	57,4	< 1	16	1	< 1	4	2

Tabela 2: Comparação entre os tempos de execução das heurísticas. Os tempos são dados em segundos.

Referências

Taha, H. A. (1987), Operations Research - An Introduction, Fourth Edition, Macmillan Publishing Company.

Nemhauser, G. L. e Wolsey, L. A. (1988), Integer and Combinatorial Optimization, New York, *John Wiley and Sons*.

Stormo, G. e Hartzell III, G.W. (1991), Identifying protein-binding sites from unaligned DNA fragments, *Proc. Natl. Acad. Sci. USA*, vol 88, 5699-5703.

G. Hertz and G. Stormo (1995), Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps, *Proc. 3rd Int'l Conf. Bioinformatics and Genome Research*, Lim and Cantor, World Scientific, 201–216.

Ben-Dor, A. and Lancia, G. and Perone, J. and Ravi, R. (1997), Banishing bias from consensus sequences, A. Apostolico and J. Hein editors, *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching*, vol 1264, Lecture notes in computer science, Springer-Verlag, Aarhus, Denmark, 247–261.

Gasieniec, L., Jansson, J. e Lingas, A. (1999), Efficient Approximation Algorithms for the Hamming Center Problem, *Proc. Tenth ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, Maryland, Society for Industrial and Applied Mathematics, Philadelphia, PA, S905–S906.

Rajasekaran, S., Hu, Y., Luo, J., Nick, H., Pardalos, P., Sahni, S. e Shaw, G. (2001a), Efficient Algorithms for Similarity Search, *Journal of Combinatorial Optimization*, vol 5, 125–132.

Rajasekaran, S., Nick, H., Pardalos, P., Sahni, S. e Shaw, G. (2001b), Efficient Algorithms for Local Alignment Search, *Journal of Combinatotial Optimization*, vol 5, 117-124.

Fellows, M.R., Gramm, J. e Niedermeier, R. (2002), On the parameterized intractability of Closest Substring and related problems, H. Alt and A. Ferreira editors, *Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science (STACS 2002)*, Lecture Notes in Computer Science, Springer-Verlag, no. 2285, 262–273.

Li, M., Ma, B. e Wang, L. (2002), On the closest string and substring problems, *Journal of the ACM*, vol 49, no. 2, 157–171.

Lanctot, K., Li, M., Ma, B., Wang, S. e Zhang, L. (2003), Distinguishing string selection problems, *Information and Computation*, vol 185, no. 1, 41–55.

Meneses, C.N., Lu, Z., Oliveira, C.A.S. e Pardalos, P.M. (2004), Optimal Solutions for the Closest String Problem via Integer Programming, *INFORMS Journal on Computing*, vol 16, no. 4, 419–429.

Meneses, C. N., Pardalos, P. M., Resende, M. G. C. e Vazacopoulos, A. (2005), Modeling and Solving String Selection Problems, *BIOMAT 2005 International Symposium on Mathematical and Computational Biology – Selected Contributed Papers*.

Gomes, F.C., Meneses, C.N., Pardalos, P.M. e Viana, G.V.R. (2008), A Parallel Multistart Algorithm for the Closest String Problem, *Computers & Operations Research*, 35, 3636-3643.

de Souza, C. C. (1993), The Graph Equipartition Problem: Optimal Solutions, Extensions and Applications, PhD Thesis, Université Catholique de Louvain.