MODELING AND SOLVING STRING SELECTION PROBLEMS

C.N. MENESES, P.M. PARDALOS, M.G.C. RESENDE, AND A. VAZACOPOULOS

ABSTRACT. We consider four important combinatorial problems that arise in computational biology and show how they can be modeled as integer programming problems. These models are then solved using branch-and-bound algorithms. Computational experiments using real and simulated data are performed and the effectiveness of the algorithms is analyzed.

1. INTRODUCTION

In this paper we study four string selection problems that arise in computational biology applications. We show how to model these problems using Integer Programming (IP) and we carry out computational experiments using these models. The experimental results show the effectiveness of IP techniques for solving problems in computational biology. For all tested instances, it was possible to solve them to optimality in a few minutes in a personal computer.

In section 2 we define the problems studied and in section 3 we introduce notation. In section 4 we formulate each problem described in section 2 using integer programming. In section 5 we present computational experiments over real and simulated instances. Finally, in section 6 conclusion remarks are given.

2. DEFINITION OF THE PROBLEMS

In this section we define the problems studied in this paper. For any two strings s and t of same length (i.e., $|s| = |t|$) we denote by $d_H(s,t)$ the Hamming distance between them, which is defined as the number of mismatched positions. For example, if $s =$"ACT" and $t =$"CCA", then $d_H(s,t) = 2$.

Closest Substring Problem (CSSP)
Instance: Given a finite set $\mathcal{S}_c = \{s^1, s^2, \ldots, s^n\}$ of strings of length at least m over an alphabet $\mathcal{A}$.

Objective: Find a string x of length m over $\mathcal{A}$ minimizing d_c such that for *every* string s^i in $\mathcal{S}_c$, $d_H(x,y) \leq d_c$ holds for *some* length-m substring y of s^i.

Farthest Substring Problem (FSSP)
Instance: Given a finite set $\mathcal{S}_f = \{s^1, s^2, \ldots, s^n\}$ of strings of length at least m over an alphabet $\mathcal{A}$.

Date: September 20, 2005.
Key words and phrases. Computational biology, integer programming, optimal solution.
AT&T Labs Research Technical Report TD-6GEQ9K. This research has been partially supported by NSF, NIH and CRDF grants. Cláudio N. Meneses was supported in part by the Brazilian Federal Agency for Higher Education (CAPES) – Grant No. 1797-99-9. To appear in *BIOMAT 2005*.

Objective: Find a string x of length m over $\mathcal{A}$ maximizing d_f such that for *every* string s^i in $\mathcal{S}_f$ and *every* length-m substring y of s^i, $d_H(x,y) \geq d_f$.

Close to Most String Problem (CMSP)
Instance: Given a finite set $\mathcal{S}_c = \{s^1, s^2, \ldots, s^n\}$ of strings of length m over an alphabet $\mathcal{A}$ and a threshold $k_c > 0$.

Objective: Find a string x of length m over $\mathcal{A}$ maximizing the number of strings s^i in $\mathcal{S}_c$ such that $d_H(x,s^i) \leq k_c$.

Distinguishing String Selection Problem (DSSP)
Instance: Given two finite sets of strings $\mathcal{S}_c$ and $\mathcal{S}_f$, all of length at least m over an alphabet $\mathcal{A}$, and two positive integers k_c and k_f.

Objective: Find a string x of length m over $\mathcal{A}$ such that for each string s_c in $\mathcal{S}_c$, there exists *some* length-m substring y_c of s_c satisfying $d_H(x,y_c) \leq k_c$, and for *every* length-m substring y_f of string s_f in $\mathcal{S}_f$, $d_H(x,y_f) \geq k_f$.

All these four problems are NP-hard and have applications in computational biology [3]. There exist approximation algorithms for each of these problems (see e.g., [1, 3, 4]).

3. Notation

In this section we present the notation we will use throughout this paper. We use s^i_j to denote the character in the j-th position of string s^i. For example, if s^i ="ACGT", then s^i_1 ='A', s^i_2 ='C', s^i_3 ='G', and s^i_4 ='T'. Suppose that $\mathcal{S} = \{$"ACGTCA","TTAC","CCGC"$\}$. Note that the first string in $\mathcal{S}$ has length 6 and the other two strings in that set have length 4. We use V_k to denote the set of characters in the k-th position of strings in $\mathcal{S}$. For $\mathcal{S}$ as stated above, we have $V_1 = \{$'A','T','C'$\}$, $V_2 = \{$'C','T'$\}$, $V_3 = \{$'G','A'$\}$, $V_4 = \{$'T','C'$\}$, $V_5 = \{$'C'$\}$, and $V_6 = \{$'A'$\}$. We use $s^{i,k}$ to denote the k-th substring of length m of string s^i. If $\mathcal{S}$ is as stated above and supposing $m = 3$, then $s^{1,k}$, for $k = 1, \ldots, |s^1| - m + 1$, consists of the substrings "ACG","CGT", "GTC", and "TCA". That is, we start with the m first characters of s^i, and then move to right one character of s^i at a time to obtain the remaining length m substrings of s^i. We use V to denote the union of the V_k's. For example, if $\mathcal{S}$ is as stated above, then $V = \cup_{k=1}^{6} V_k = \{$'A','C','G','T'$\}$.

4. Integer Programming Models

In this section we derive IP models for the problems defined in section 2.

4.1. **Closest Substring Problem.** Define the variables

$$x_{j,k} = \begin{cases} 1 & \text{if character } j \text{ is used at } k\text{-th position in a solution} \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{i,k} = \begin{cases} 1 & \text{if length-}m \text{ substring } s^{i,k} \text{ of } s^i \in \mathcal{S}_c \text{ satisfies } d_H(x,s^{i,k}) \leq d_c \\ 0 & \text{otherwise.} \end{cases}$$

Then, the CSSP can be modeled as

$$\min\ d_c$$

subject to:

$$\sum_{j\in V} x_{j,k} = 1 \qquad k = 1,\dots,m \tag{1}$$

$$d_c + \sum_{j=1}^{m} x_{s_j^{i,k},j} \geq m y_{i,k} \qquad k = 1,\dots,|s^i|-m+1; i = 1,\dots,n \tag{2}$$

$$\sum_{k=1}^{|s^i|-m+1} y_{i,k} \geq 1 \qquad i = 1,\dots,n \tag{3}$$

$$x_{j,k} \in \{0,1\} \qquad j \in V, k = 1,\dots,m \tag{4}$$

$$y_{i,k} \in \{0,1\} \qquad k = 1,\dots,|s^i|-m+1; i = 1,\dots,n \tag{5}$$

$$d_c \in Z_+ \tag{6}$$

Equalities (1) guarantee that only one character in V is chosen to each solution position. Inequalities (2) and (3) altogether force x to satisfy $d_H(x, s^{i,k}) \leq d_c$ for at least one length-m substring $s^{i,k}$ of $s^i \in \mathcal{S}_c$. Constraints (4) and (5) state binary variables, and (6) forces d_c to assume a nonnegative integer value.

Example 1. *Suppose* $\mathcal{S}_c = \{$*"ACGTCA","TTAC","CCGC"*$\}$ *and* $m = 4$. *String* $x =$*"CTGC" satisfies* $d_H(x,y) \leq d_c$ *for some substring* y *of* $s^i \in \mathcal{S}_c$ *with* $d_c = 2$.

4.2. **Farthest Substring Problem.** Define the variables

$$x_{j,k} = \begin{cases} 1 & \text{if character } j \text{ is used at } k\text{-th position in a solution} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the FSSP can be modeled as

$$\max\ d_f$$

subject to:

$$\sum_{j\in V} x_{j,k} = 1 \qquad k = 1,\dots,m \tag{7}$$

$$m - \sum_{j=1}^{m} x_{s_j^{i,k},j} \geq d_f \qquad k = 1,\dots,|s^i|-m+1; i = 1,\dots,n \tag{8}$$

$$x_{j,k} \in \{0,1\} \qquad j \in V, k = 1,\dots,m \tag{9}$$

$$d_f \in Z_+ \tag{10}$$

Equalities (7) guarantee that only one character in V is chosen to each solution position. Inequalities (8) force x to satisfy $d_H(x, s^{i,k}) \geq d_f$ for every length-m substring $s^{i,k}$ of $s^i \in \mathcal{S}_f$. Constraints (9) state binary variables, and (10) forces d_f to assume a nonnegative integer value.

Example 2. *Suppose* $\mathcal{S}_f = \{$*"ACGTCA","TTAC","CCGC"*$\}$ *and* $m = 4$. *String* $x =$*"AACG" satisfies* $d_H(x,y) \geq d_f$ *for every substring* y *of* $s^i \in \mathcal{S}_f$ *with* $d_f = 3$.

4.3. **Close to Most String Problem.** Define the variables

$$x_{j,k} = \begin{cases} 1 & \text{if character } j \text{ is used at } k\text{-th position in a solution} \\ 0 & \text{otherwise.} \end{cases}$$
$$y_i = \begin{cases} 1 & \text{if length-}m \text{ string } s^i \in \mathcal{S}_c \text{ satisfies } d_H(x,s^i) \leq k_c \\ 0 & \text{otherwise.} \end{cases}$$

Then, the CMSP can be modeled as

$$\max \sum_{i=1}^{n} y_i$$

subject to:

$$\sum_{j \in V_k} x_{j,k} = 1 \qquad k = 1,\ldots,m \tag{11}$$
$$k_c + \sum_{j=1}^{m} x_{s^i_j,j} \geq m y_i \qquad i = 1,\ldots,n \tag{12}$$
$$x_{j,k} \in \{0,1\} \qquad j \in V_k,\ k = 1,\ldots,m \tag{13}$$
$$y_i \in \{0,1\} \qquad i = 1,\ldots,n \tag{14}$$

Recall that m and k_c are positive constants. Equalities (11) guarantee that only one character in each V_k is chosen. Inequalities (12) say that if a character in a string s^i is not in a solution x, then that character will contribute to increasing the Hamming distance from x to s^i. Constraints (13) and (14) state binary variables.

Example 3. *Suppose* $\mathcal{S}_c = \{$*"ACGT","TTAC","CCGC"*$\}$. *If* $k_c = 2$, *then string* $x =$*"ACAC" is an optimal solution to* $\mathcal{S}$ *and it satisfies* $d_H(x,s^i) \leq 2$ *for the 3 strings in* $\mathcal{S}_c$. *Now, if* $k_c = 1$, *then string* $x =$*"ACGC" is an optimal solution to* $\mathcal{S}$ *and it satisfies* $d_H(x,s^i) \leq 1$ *for 2 strings in* $\mathcal{S}_c$.

4.4. **Distinguishing String Selection Problem.** Note that the DSSP is a decision problem. We cast it as an optimization problem by introducing an objective function. Let

$$x_{j,k} = \begin{cases} 1 & \text{if character } j \text{ is used at } k\text{-th position in a solution} \\ 0 & \text{otherwise.} \end{cases}$$
$$y_{i,k} = \begin{cases} 1 & \text{if length-}m \text{ string } x \text{ satisfies } d_H(x,s^{i,k}) \leq k_c \text{ for some } s^{i,k} \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{S} = \mathcal{S}_c \cup \mathcal{S}_f$, for $\mathcal{S}_c$ and $\mathcal{S}_f$ as defined in the DSSP instance. Let $V = \cup_k V_k$, where V_k is computed over $\mathcal{S}$. Then, the DSSP can be modeled as

Maximize z

subject to:

$$z = 0 \tag{15}$$

$$\sum_{j \in V} x_{j,k} = 1, k = 1, \ldots, m \tag{16}$$

$$k_c + \sum_{j=1}^{m} x_{s_j^{i,k},j} \geq m y_{i,k}, k = 1, \ldots, |s^i| - m + 1; s^i \in \mathcal{S}_c; i = 1, \ldots, |\mathcal{S}_c| \tag{17}$$

$$\sum_{k=1}^{|s^i|-m+1} y_{i,k} \geq 1, s^i \in \mathcal{S}_c; i = 1, \ldots, |\mathcal{S}_c| \tag{18}$$

$$m - \sum_{j=1}^{m} x_{s_j^{i,k},j} \geq k_f, k = 1, \ldots, |s^i| - m + 1; s^i \in \mathcal{S}_f; i = 1, \ldots, |\mathcal{S}_f| \tag{19}$$

$$x_{j,k} \in \{0,1\}, j \in V, k = 1, \ldots, m \tag{20}$$

$$y_{i,k} \in \{0,1\}, k = 1, \ldots, |s^i| - m + 1; s^i \in \mathcal{S}_c; i = 1, \ldots, |\mathcal{S}_c| \tag{21}$$

Recall that k_c and k_f are positive constants. Equalities (16) guarantee that only one character in V is chosen to each solution position. Inequalities (17) and (18) altogether force x to satisfy $d_H(x, s^{i,k}) \leq k_c$ for *at least one* length-m substring $s^{i,k}$ of $s^i \in \mathcal{S}_c$. Inequalities (19) force x to satisfy $d_H(x, s^{i,k}) \geq k_f$ for *every* length-m substring $s^{i,k}$ of $s^i \in \mathcal{S}_f$. Constraints (20) and (21) state binary variables.

5. Experimental Analysis

In this section we present computational experiments carried out with branch-and-bound algorithms (B&B) designed using the IP formulations described in the previous section. Initially, we describe the set of instances used in the tests, and then we show the results obtained by the B&B.

5.1. **Instances and Test Environment.** Two classes of instances were tested. In the first class an alphabet with four characters is used, while the second class uses an alphabet with 20 characters. This choice of alphabets was motivated by real applications on analysis of DNA and amino-acid sequences, respectively.

All tests were executed on a Pentium 4 CPU with speed of 2.80GHz and 512MB of RAM under WindowsXP. The algorithms were implemented in the C++ programming language and Xpress-MP [2] was used to solve the IP formulations.

5.2. **Closest Substring Problem and Farthest Substring Problem.** Table 1 and 2 show results for instances of the CSSP and FSSP. We have limited the branch-and-bound trees to CSSP and FSSP to have at most one hundred and two hundred thousands nodes, respectively. The columns in those tables have the following meaning: n is the number of strings in the set $\mathcal{S}_c$ (resp. $\mathcal{S}_f$), m is as stated in the problem definition, *val* is the solution value for the LP relaxation, *time* is the CPU time in seconds, *best integer* is the value of the best feasible solution found by the B&B considering the specified number of nodes in the branch-and-bound tree, *best lb (up)* gives the best lower (upper) bound value in the branch-and-tree, and *gap* represents the percentual difference between *best integer* and *best lb (up)*.

As can be seen in Table 1, the proposed formulation for the CSSP needs to be improved since sometimes the optimal solution given by the LP relaxation is zero and this value is only improved by adding cuts. In contrast, the formulation for the FSSP is quite strong as can be viewed in Table 2.

TABLE 1. Results to the instances over the alphabet with 20 characters for CSSP.

Instance		LP		IP			
n	m	val	time	best integer	best lb	gap (%)	time
10	100	40.5	0.08	55	53	3.64	344.81
10	120	0.0	0.03	65	63	5.90	426.73
10	140	0.0	0.03	78	60	23.08	479.14
10	160	0.0	0.03	89	60	32.58	704.97
10	180	67.5	0.03	101	85	15.84	876.59

TABLE 2. Results to the instances over the alphabet with 20 characters for FSSP.

Instance		LP		IP			
n	m	val	time	best integer	best ub	gap (%)	time
10	100	82.01	0.05	81	81	0.00	3.59
10	120	96.66	0.09	95	96	1.05	452.06
10	140	109.99	0.22	108	109	0.93	462.36
10	160	125.83	0.23	124	125	0.81	950.26
10	180	141.92	0.64	140	141	0.71	962.28

5.3. **Close to Most String Problem.** The instances for the CMSP are from McClure data set [5] and randomly generated as well. For each of the McClure instance considered, the size of the strings (m) is equal to the length of the smallest string in the set (this was necessary since the McClure instances have strings of different lengths). We removed the last characters for strings with length greater than the minimum.

Tables 3 and 4 show the results for the CMSP. The columns in those tables have the following meaning: *name* is the instance name, n is the number of strings in the set $\mathcal{S}_c$, m is the string lengths, *val* is the solution value for the LP relaxation and IP solution value, *time* is the CPU time in seconds.

Clearly, the results demonstrate the efficacy of the IP formulation proposed.

6. CONCLUDING REMARKS

In this paper we have shown four integer programming formulations to four important combinatorial problems that arise in computational biology. The relevance of these problems in practice justify their study from an exact point of view.

The computational experiments give evidences that the IP formulations are effective for real and randomly generated instances. As future research we intend to test exhaustively the models proposed.

TABLE 3. Results for the McClure instances (alphabet with 20 characters) for CMSP.

Instance				LP		IP	
name	n	m	k	val	time	val	time
mc582.10	10	141	97	10	0.06	10	0.28
mc582.12	12	141	97	12	0.06	12	0.03
mc582.6	6	141	95	6	0.05	6	0.02
mc586.6	6	100	72	6	0.03	6	0.03
mc586.10	10	98	75	10	0.05	10	0.03
mc586.12	12	98	76	12	0.08	12	0.14

TABLE 4. Results for randomly generated instances (alphabet with 4 characters) for CMSP.

Instance			LP		IP	
n	m	k	val	time	val	time
10	1000	579	9.998	0.34	9	1.47
10	1500	870	9.995	0.66	9	0.41
10	2000	1163	9.997	0.89	9	1.53
10	3000	1739	9.998	1.19	9	6.86
10	4000	2320	9.998	1.75	9	9.28
10	5000	2901	9.998	2.36	9	14.99

REFERENCES

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

[2] Dash Optimization Inc. *Xpress-Optimizer Reference Manual*, 2004.

[3] K. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. *Information and Computation*, 185(1):41–55, 2003.

[4] M. Li, B. Ma, and L. Wang. On the closest string and substring problems. *Journal of the ACM*, 49(2):157–171, 2002.

[5] M. McClure, T. Vasi, and W. Fitch. Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.*, 11:571–592, 1994.

[6] C. N. Meneses, Z. Lu, C. A. S. Oliveira, and P. M. Pardalos. Optimal solutions for the closest-string problem via integer programming. *INFORMS Journal on Computing*, 16(4):419–429, 2004.

[7] C. N. Meneses, C. A. S. Oliveira, and P. M. Pardalos. Optimization techniques for string selection and comparison problems in genomics. *IEEE Engineering in Biology and Medicine Magazine*, 24(3):81–87, 2005.

(C.N. Meneses) DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, 303 WEIL HALL, GAINESVILLE, FL 32611 USA.
E-mail address, C.N. Meneses: claudio@ufl.edu

(P.M. Pardalos) DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, 303 WEIL HALL, GAINESVILLE, FL 32611 USA.
E-mail address, P.M. Pardalos: pardalos@ufl.edu

(M.G.C. Resende) INTERNET AND NETWORK SYSTEMS RESEARCH, AT&T LABS RESEARCH, 180 PARK AVENUE, ROOM C241, FLORHAM PARK, NJ 07932 USA.
E-mail address, M.G.C. Resende: mgcr@research.att.com

(A. Vazacopoulos) DASH OPTIMIZATION, ENGLEWOOD CLIFFS, NJ 07632 USA.
E-mail address, A. Vazacopoulos: av@dashoptimization.com