Reconstruction of DNA fragments: a graph model

C. E. Ferreira[1] C. C. de Souza[2] Y. Wakabayashi[1]

[1] *Instituto de Matemática e Estatística*
Universidade de São Paulo
e-mail: [cef,yw]@ime.usp.br

[2] *Departamento de Ciência da Computação*
Universidade Estadual de Campinas
cid@dcc.unicamp.br

Abstract

In this paper we consider a problem that arises in the process of reconstruction of DNA fragments. We present a graph theoretical formulation of the problem and mention some extensions. We show this problem to be $\mathcal{NP}$-hard. A 0-1 Integer Linear Programming formulation of the problem is given and preliminary results of a branch-and-bound algorithm based on this formulation are discussed.

Key words: *integer programming, computational biology, covering with paths.*

1 Introduction

Let Σ be a finite alphabet and ℓ a positive integer. If s is a string of characters over Σ, then we denote by *ℓ-last(s)*, resp. *ℓ-first(s)*, the substring of s consisting of the *last*, resp. *first*, ℓ characters of s.

For each positive integer k, we define the *Minimum k-Contig Problem*, denoted by MkCP, as follows. We are given a collection $\mathcal{C}$ of strings of characters over an alphabet Σ and we are willing to concatenate the given strings so as to obtain another collection $\mathcal{C}'$ consisting of a smallest possible number of larger strings. Each string z in $\mathcal{C}'$ must be a concatenation $s_1 s_2 \ldots s_m$ of different strings in $\mathcal{C}$, with the property that for any two consecutive strings in z, say s_j and s_{j+1}, the following holds: *ℓ-last(s_j)= ℓ-first(s_{j+1})* for some $\ell \geq k$ (we call the strings $s_1, s_2, \ldots, s_m$ the *skeleton* of the k-contig). Furthermore, if a string $u \in \mathcal{C}$ is a substring of a string v, which forms a k-contig z in $\mathcal{C}'$, we say that u is also covered by z. The strings in $\mathcal{C}'$ are called *k-contigs*. We require each string in $\mathcal{C}$ to be covered exactly once by exactly one k-contig in $\mathcal{C}'$. Thus the goal is to combine properly the strings in $\mathcal{C}$ to form a minimum number of k-contigs in the new collection $\mathcal{C}'$.

This problem occurs in the reconstruction of DNA fragments. A usual strategy to determine the sequence of the basis in a DNA molecule (which can be seen as a string over the alphabet {A, T, C, G}) can be described as follows. First many copies of this molecule are produced and, afterwards, these copies are (by means of chemical substances) broken into several small pieces that we are able to handle. Then, the problem is *how to "glue" the small pieces in the correct way to reconstruct the original sequence?* (see [M92]).

The MkCP arises in this context. The idea is to obtain —for a given positive integer k and a collection $\mathcal{C}$ of pieces— a collection $\mathcal{C}'$ of k-contigs in such a way that the original molecule is

completely covered by the k-contigs in $\mathcal{C}'$. A collection $\mathcal{C}'$ with the smallest possible cardinality gives a best possible approximation to the original molecule.

Example 1.1: Let $k = 2$ and $\mathcal{C}$ consist of the following strings:

1. CCTAATGCTT
2. TGTTTAGCCTGCGT
3. CTGTGTTTAGCCT
4. GTAGACAACCCTGTG
5. TGTTTAGCCTG
6. CCTGCGTTTTGTGCC
7. TTTTGTCCAT
8. TTTTGTCCATC
9. TGCGTTTTGTGC
10. GACGTAGACA

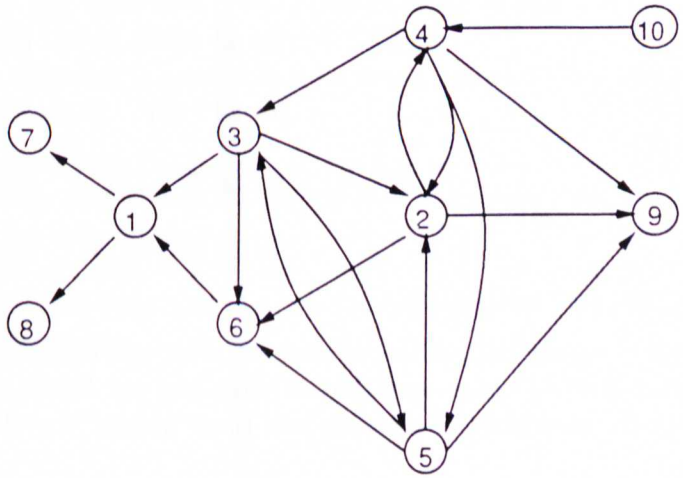
An optimal solution for the corresponding minimum 2-contig problem has cardinality 1 and the skeleton of the 2-contig is given by $\{10, 4, 3, 2, 6, 1, 8\}$. Observe that string 5 is a substring of 2, string 7 a substring of 8, and string 9 is a substring of 6.

2 A model using graph theory

Consider an instance of the MkCP consisting of a collection $\mathcal{C}$ of strings and a positive integer $k \geq 1$. Construct a directed graph $G = (V, A)$ as follows: each node $i \in V$ corresponds to a string $s_i \in \mathcal{C}$, and there is an arc $(i, j) \in A$ if and only if there exists $\ell \geq k$ such that ℓ-*last* (s_i)= ℓ-*first* (s_j).

It is easy to see that any directed path in G corresponds to the skeleton of a k-contig. Nodes corresponding to strings that are substrings of others are called *Steiner nodes*, and need not necessarily be covered by a collection of paths. The other nodes of the graph are called *terminals*. Thus a smallest set of node-disjoint directed paths covering the terminals of G gives a solution for the given instance of the MkCP.

Example 2.1: For the instance given in Example 1.1, the corresponding graph is the following. Nodes 5, 7 and 9 are Steiner Nodes.



There are several other versions of the problem which could be of interest. For instance, we may allow that a string in $\mathcal{C}$ can be used more than once to form k-contigs. The graph model is the same but now we are looking for a collection of node-disjoint walks (or trails) covering the nodes of the graph. Steiner nodes are also admitted.

Other versions allow concatenation of strings which differ by at most ϵ characters in their final and initial positions provided that this occurs in a substring of size at least k. A more elaborated version allows concatenation of strings by considering the *reverse complements*.

In this paper we concentrate our attention in the first version described in the introduction.

3 $\mathcal{NP}$-completeness of MkCP

The version of the problem we are considering here is $\mathcal{NP}$-hard. We prove that the corresponding decision version of the MkCP is $\mathcal{NP}$-complete, even if no Steiner node is allowed. This is shown, by reducing to the MkCP, the problem of finding a minimum covering of the nodes of a directed graph with maximum degree 3 by a collection of node-disjoint paths (which is known to be $\mathcal{NP}$-hard *cf*. [GJ79]).

4 An Integer Programming Formulation for MkCP

We have seen earlier that the MkCP can be modelled as the problem of finding a minimum path covering of a directed graph $G = (V, A)$. The latter problem can be formulated as an integer programming problem as follows. Let us retain our attention to the case where no Steiner node is present. The presence of Steiner nodes can be handled by slightly modifying the objective function.

Let us introduce for each arc $(i, j) \in A$ a variable x_{ij} with the following interpretation:

$$x_{ij} = \begin{cases} 1, & \text{if the arc } (i,j) \text{ is in a path;} \\ 0, & \text{otherwise.} \end{cases}$$

Thus, a 0-1 ILP formulation for the problem is given by

$$\begin{aligned} z = \max \quad & \sum_{e \in A} x_e, \\ & \sum_{j\,:\,(j,i) \in A} x_{ji} \leq 1, \qquad \text{for all } i \in V, \qquad (1) \\ (P) \qquad & \sum_{j\,:\,(i,j) \in A} x_{ij} \leq 1, \qquad \text{for all } i \in V, \qquad (2) \\ & \sum_{e \in E(C)} x_e \leq |C| - 1, \quad \text{for all } \varnothing \neq C \subseteq V, \quad (3) \\ & x_{ij} \in \{0, 1\}, \qquad \text{for all } (i,j) \in A, \qquad (4) \end{aligned}$$

where $E(C) := \{(i, j) \in A \mid i \in C \text{ and } j \in C\}$.

The first set of inequalities means that in the solution at most one arc of A can enter any node of V. Analogously, inequalities (2) guarantee that at most one arc leaves each node of V. The inequalities (3) eliminate the possibility of choosing arcs that "induce" a cycle going through a set of nodes.

It is not difficult to check that a 0-1 vector x satisfies (1) to (4) if and only if the set of arcs $a \in A$ with $x_a = 1$ induces a collection of node-disjoint paths in G. Moreover, we know that the number of arcs in a collection of node-disjoint paths equals the number of vertices in G minus the number of these paths. Thus, minimizing the number of node-disjoint paths is equivalent to maximizing the number of arcs in these paths (value of z).

It is interesting to note that, although the number of inequalities in this formulation is exponential, the separation of these constraints, called *subtour elimination constraints*, can be solved in polynomial time (*cf.* [PG85]). It means that the optimum value of the relaxed LP can be calculated in polynomial time (*cf.* [GLS88]).

5 A branch-and-bound algorithm for the MkCP

One way to tackle the problem is to apply the usual linear programming relaxations in a *branch-and-bound* or *branch-and-cut* algorithm. The latter leads us to the study of valid inequalities for the polyhedron defined as the convex hull of the feasible (integer) solutions of (P). Let us denote this polyhedron by $P_k(G)$, that is,

$$P_k(G) := \text{ conv } \{x \in \mathbb{R}^A | \ x \text{ satisfies (1) to (4)}\}.$$

Note that the vertices of $P_k(G)$ are in one-to-one correspondence with the feasible solutions of MkCP. Furthermore, it can be easily shown that $P_k(G)$ is full-dimensional.

The idea of the approach is to start with a relaxation of the polyhedron $P_k(G)$ and to solve iteratively better approximations of this polyhedron, obtained by using facet-defining or, at least, valid inequalities that are violated by the optimal solution of the current relaxation. When we are not able to find any violated valid inequality the procedure fixes the value of some variable and proceeds in a branch-and-bound fashion.

In the preliminary version of the algorithm we tested the quality of the initial relaxation we propose for the problem. So, we begin with the LP given by constraints (1) and (2), and do not introduce any new inequality in the LP formulation. The problem is, then, solved by applying a plain branch-and-bound procedure.

The results we obtained for medium size instances are promising. We were able to solve instances with up to 100 nodes and 1266 arcs. The original string is 5000 characters long and has been generated randomly on the alphabet {A, T, C, G}. Each substring has length between 500 and 700 characters. Each substring is generated by choosing randomly its length and also the position it starts in the original string.

The following table summarizes the results. In the first column are given the number of nodes and arcs in the graph. In the second column the value of k used to construct the graph is given. In column 3 it is shown the value of the optimum solution. Columns 4 and 5 show the number of LPs solved and the number of nodes in the branch-and-bound tree, respectively. Finally, in the last column we present the CPU time spent to solve the problems in a SPARC 1000. We use CPLEX to solve the LP in each iteration.

# nodes	# arcs	k	opt. sol.	# LP	# BB nodes	CPU Time (sec.)
10	11	3	2	1	1	0.37
15	29	3	2	36	47	0.59
20	47	10	2	1	1	0.16
30	105	5	2	1	1	0.19
50	330	10	2	1	1	0.22
50	333	5	2	3	5	0.33
70	589	10	2	1	1	0.21
70	598	5	2	78	143	1.92
80	798	5	2	3	5	0.30
100	1266	5	2	82	148	3.04

6 Conclusions and future research

The computational results we present in this paper shows that the ILP formulation we suggest for the MkCP can be satisfactorily used to solve medium size (up to 100 nodes and 1200 arcs) instances of the problem. Our aim is to increase the size of the instances we are able to solve to optimality. For that, we plan to use a more powerful machinery, which includes using other classes of valid inequalities to improve the lower bounds in each node of the branch-and-bound tree, and using better primal heuristics. It is still a rather long way until we are able to solve instances with the size of interest for the computational biology community. The DNA molecules that they intend to determine have length of millions of basis, and typically these molecules are broken into several hundreds of pieces.

7 Acknowledgements

We thank Professor João Meidanis (Unicamp) for bringing this problem into our attention.

This research has been partially supported by CNPq (Project ProComb of ProTeM–II–CC; Proc. 680065/94-6). The authors from IME-USP have also been partially supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo – Proc. 93/0603-1).

References

[GJ79] M. R. Garey and D. S. Johnson, *Computer and Intractability - A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York (1979).

[GLS88] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin (1988).

[M92] J. Meidanis, "Algorithms for Problems in Computational Genetics", *PhD. Thesis*, University of Wisconsin-Madison (1992).

[PG85] M. W. Padberg and M. Grötschel, "Polyhedral Aspects of the Travelling Salesman Problem II : Computations", in E. L. Lawler, J. K. Lenstra and A. H. G. Rinooy Kan eds., *The Travelling Salesman Problem*, Wiley, New York (1985).