A Polyhedral Investigation of the LCS Problem and a Repetition-Free Variant

Cristina G. Fernandes, Carlos E. Ferreira,
Christian Tjandraatmadja, and Yoshiko Wakabayashi

Universidade de São Paulo, Brazil
{cris,cef,christj,yw}@ime.usp.br

Abstract. We consider the longest common subsequence problem (LCS) and a variant of it where each symbol may occur at most once in the common subsequence. The LCS is a well-known problem that can be solved in polynomial time by a dynamic programming algorithm. We provide a complete description of a polytope we associate with the LCS. The integrality of this polytope can be derived by showing that it is in fact the clique polytope of a perfect graph. The repetition-free version of the problem is known to be difficult. We also associate a polytope with this version and investigate its facial structure. We present some valid and facet-defining inequalities for this polytope and discuss separation procedures. Finally, we present some computational results of a branch and cut algorithm we have implemented for this problem.

1 Introduction

Given two finite sequences s and t over an alphabet, the longest common subsequence problem (LCS) consists in finding a longest common subsequence of s and t. It is well known that this problem can be solved in polynomial time by a dynamic programming algorithm that runs in $O(|s||t|)$ time (see [4]). The LCS has important applications in Bioinformatics and Computational Biology, where the sequences are genomes and a common subsequence may be interpreted as a similarity measure between the genomes. It is also present in the core of the unix `diff` command.

We associate a polytope with the LCS and study its facial structure. For that, given an instance of the problem, we represent the feasible solutions of this instance as vectors in $\mathbb{R}^n$ (for some dimension n) and consider the polytope defined as the convex hull of these vectors. We give a complete description of this polytope, exhibiting all of its facets (faces of maximal dimension). Since the LCS is polynomially solvable, it is expected that such a complete description can be provided (see [6,7] for details). We discuss the description we provide in this paper and relate it with other known results.

We also study a variation of the LCS in which each symbol of the alphabet is allowed to occur at most once in the sought common subsequence. We refer

E.S. Laber et al. (Eds.): LATIN 2008, LNCS 4957, pp. 329–338, 2008.

to this version as the **Repetition-Free LCS** problem (RFLCS). It has been proved that this problem is NP-hard. Indeed, Adi et al. [1] proved that it is APX-hard even if each symbol appears at most twice in each of the given sequences. Another variation considered in the literature is the so-called *exemplar model*. In this problem, as in the RFLCS, each symbol may appear at most once in the common subsequence. Besides, some symbols are mandatory and must appear in the subsequence. This model has been studied by Bonizzoni et al. [2] who proved that the problem is also APX-hard. Yet another related variation was proposed by Sankoff [9] in the studies of gene families. All these variations may be useful in the comparisons of genomes [2,9] and may prove to be useful in other contexts as well.

This paper is organized as follows. In Section 2 we give a complete description of the polytope we associate with the LCS and show how to solve the separation problem for it in polynomial time. Moreover we show the relationship between the LCS polytope and the clique polytope of a class of perfect graphs. In Section 3 we provide a formulation for the RFLCS and present some results on valid and facet-defining inequalities for the corresponding polytope. Some computational results of a branch and cut algorithm we developed for the RFLCS are shown in Section 4. Finally, in Section 5 we present some conclusions.

2 A Polyhedral Study of the LCS Polytope

Consider an instance of the LCS consisting of two sequences $s = s_1, \ldots, s_n$ and $t = t_1, \ldots, t_m$ over some alphabet. For each symbol a of the alphabet, let $s(a)$ be the set of indices i of s such that $s_i = a$, and let $t(a)$ be defined analogously. Let $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, m\}$ be the set of all pairs (i, j) in $s(a) \times t(a)$ where a ranges over all symbols of the alphabet. That is, $E = \cup_a \big(s(a) \times t(a)\big)$.

Note that any common subsequence w of s and t can be represented by a vector z in $\{0,1\}^E$ as follows. If $w = s_{i_1}, \ldots, s_{i_p} = t_{j_1}, \ldots, t_{j_p}$, where $i_1 < \cdots < i_p$ and $j_1 < \cdots < j_p$, then for each pair ij in E we have that $z_{ij} = 1$ if and only if there exists an index ℓ in $\{1, \ldots, p\}$ such that $i = i_\ell$ and $j = j_\ell$. Thus, the feasible solutions of this instance of the LCS are the vertices of the following polytope (defined as the convex hull of these solutions).

$$P_{\text{LCS}} := \text{conv}\{z \in \{0,1\}^E \mid z \text{ represents a common subsequence of } s \text{ and } t\}.$$

Obviously, finding a vector z^* in P_{LCS} with maximum number of nonzero components is equivalent to finding a longest common subsequence of s and t.

Proposition 1. *The polytope P_{LCS} is full-dimensional.*

Proof. It is sufficient to observe that the zero vector and the unit vectors e^{ij}, for all (i, j) in E (that is, the vector such that $e^{ij}_{ij} = 1$ and $e^{ij}_{k\ell} = 0$ for all $(k, \ell) \neq (i, j)$), are in P_{LCS} and are affinely independent. Thus the polytope P_{LCS} has dimension $|E|$. □

We say that two distinct pairs (i,j) and (k,ℓ) in E **cross** if ($i \leq k$ and $j \geq \ell$) or ($k \leq i$ and $\ell \geq j$). A simple integer programming formulation for the LCS follows.

$$\begin{array}{lrcll} \text{maximize} & \sum_{(i,j)\in E} z_{ij} & & & \\ \text{subject to} & z_{ij} + z_{k\ell} & \leq & 1 & \text{for all } (i,j) \text{ and } (k,\ell) \text{ in } E \text{ that cross,} \\ & z_{ij} & \in & \{0,1\} & \text{for all } (i,j) \text{ in } E. \end{array} \tag{1}$$

Observe that z is a feasible solution of (1) if and only if it corresponds to a common subsequence of s and t.

Proposition 2. *For every (i,j) in E, the inequality $z_{ij} \geq 0$ defines a facet of P_{LCS}.*

Proof. Fix an (i,j) in E. It is easy to see that the zero vector and the unit vectors $e^{k\ell}$ for all $(k,\ell) \neq (i,j)$ are vertices of P_{LCS} that satisfy $z_{ij} = 0$ and they are affinely independent. □

We note that the inequalities in (1) do not always induce facets of P_{LCS}. As we will see in what follows, some of them may possibly be facet-defining. To simplify the notation, if z is a vector and S is a subset of E, then we denote by $z(S)$ the sum $\sum_{(i,j)\in S} z_{ij}$.

We say that a set $S \subseteq E$ is a **star** if any two distinct pairs in S cross. If S is a star then the inequality $z(S) \leq 1$ is called a **star inequality**. Note that the inequalities in (1) are star inequalities defined by stars of cardinality 2.

Lemma 1. *Consider a star $S \subseteq E$. Then the star inequality $z(S) \leq 1$ is valid for P_{LCS}, and it defines a facet if and only if S is maximal.*

Proof. It is immediate that the star inequality $z(S) \leq 1$ is valid for P_{LCS}. Now, let us prove that it defines a facet when S is maximal.

Consider a facet-defining inequality $az \leq \alpha$ and suppose

$$\{z \in \mathbb{R}^E \mid z(S) = 1\} \subseteq F := \{z \in \mathbb{R}^E \mid az = \alpha\}.$$

Note that, for each pair (p,q) in $E \setminus S$, there exists a pair (i,j) in S such that (i,j) and (p,q) do not cross (since S is maximal). Thus, e^{ij} and $e^{ij} + e^{pq}$ are incidence vectors of common subsequences of the sequences s and t and are in F. So, $a_{ij} = ae^{ij} = a(e^{ij} + e^{pq}) = a_{ij} + a_{pq}$, and therefore $a_{pq} = 0$ for every (p,q) in $E \setminus S$. Now observe that, for every (i,j) and (k,ℓ) in S, we have that e^{ij} and $e^{k\ell}$ are in F. Then, $a_{ij} = ae^{ij} = ae^{k\ell} = a_{k\ell} = \alpha$, for some constant α. This proves that $az \leq \alpha$ can be rewritten as $\alpha z(S) \leq \alpha$, and therefore the star inequality $z(S) \leq 1$ is facet-defining.

Conversely, suppose that the star inequality $z(S) \leq 1$ defines a facet of P_{LCS}. If S is not maximal, there exists a pair (p,q) in $E \setminus S$ such that (p,q) crosses all pairs in S. Thus, $S' := S \cup \{(p,q)\}$ is a star and the inequality $z(S') \leq 1$ is valid. In this case, the star inequality $z(S) \leq 1$ can be written as the sum of the inequality $z(S') \leq 1$ and the inequality $z_{pq} \geq 0$, a contradiction. □

A nice result is that the inequalities given in the lemma above define completely the polytope P_{LCS}. More precisely, any facet-defining inequality of P_{LCS} is either a nonnegativity inequality or a maximal star inequality. This result is proved in the next theorem. First, consider the following order relation on the pairs on E. Given (i,j) and (k,ℓ) in E, we write $(i,j) \preceq (k,\ell)$ if $i < k$ or ($i = k$ and $j \leq \ell$). Now, we can define a lexicographical order on the subsets of E as follows. Consider two subsets S_1 and S_2 of E. For $k = 1, 2$, let $((i_1^k, j_1^k), \ldots, (i_{\ell_k}^k, j_{\ell_k}^k))$ be the sequence of the pairs in S_k sorted according to this (total) order and refer to it as the sorted S_k. We say that S_1 is **lexicographically smaller than or equal to** S_2, and write $S_1 \preceq S_2$, if either the sorted S_1 is a prefix of the sorted S_2 or $(i_p^1, j_p^1) \preceq (i_p^2, j_p^2)$ for the first index p such that $(i_p^1, j_p^1) \neq (i_p^2, j_p^2)$ (if it exists).

Theorem 1

$$P_{\text{LCS}} = \{z \in \mathbb{R}^E \mid z_{ij} \geq 0 \quad \textit{for all } (i,j) \in E \textit{ and } z(S) \leq 1 \textit{ for all maximal star } S \subseteq E\}.$$

Proof. Let $az \leq \alpha$ be an inequality that induces a facet, say F, of P_{LCS}. First we prove that, if this inequality has negative coefficients, it must be a nonnegativity constraint for some (i,j) in E. Say $a_{ij} < 0$ and suppose that the inequality $az \leq \alpha$ is not a multiple of $-z_{ij} \leq 0$. Then, there must be a vector z in P_{LCS} such that $az = \alpha$ and $z_{ij} = 1$ (otherwise, F would be contained in the hyperplane $z_{ij} = 0$). But then $z - e^{ij}$ is also a vector in P_{LCS}, and $a(z - e^{ij}) = az - a_{ij} > \alpha$, a contradiction.

So, we may assume that $a_{ij} \geq 0$ for all (i,j) in E. Consider now the support T of the inequality $az \leq \alpha$ and some (i,j) in $E \setminus T$. It is easy to see that there must exist some (k,ℓ) in T that does not cross (i,j). Otherwise F would be contained in the hyperplane $z_{ij} = 0$. It remains to be proved that T is a star.

Let $S_1 = \{(i_1, j_1), (i_2, j_2), \ldots, (i_t, j_t)\}$ be the first maximal star in T in the lexicographical order defined above. If $az \leq \alpha$ is not a maximal star inequality, there must exist some z in $\mathbb{R}^E$ such that $az = \alpha$ and $z(S_1) = 0$. Thus, there must exist two distinct pairs (i,j) and (k,ℓ) in $T \setminus S_1$ with $z_{ij} = z_{k\ell} = 1$ such that every pair in S_1 either crosses (i,j) or (k,ℓ). Assume without loss of generality that $(i,j) \preceq (k,\ell)$. As S_1 is maximal, (i,j) cannot cross all pairs in S_1. Since S_1 is the first maximal star in the lexicographical order, there is an index p in $\{1, \ldots, t\}$ such that (i,j) does not cross (i_p, j_p) and $(i_p, j_p) \preceq (i,j)$. Thus, $j \geq j_p$. But then, (k,ℓ) must cross (i_p, j_p), and therefore $\ell \leq j_p$. This implies that $i \leq k$ and $j \geq j_p \geq \ell$, and hence (i,j) and (k,ℓ) cross, a contradiction. □

2.1 The Separation Problem for the Star Inequalities

A polynomial-time algorithm to solve the separation problem for the star inequalities is the following. Let s and t be the two sequences for which we want to find a longest common subsequence, and let $z \in \mathbb{R}^E$ be such that $0 \leq z_{ij} \leq 1$ for all (i,j) in E. Consider the problem of finding a maximum weight common subsequence of the reversal of s and the sequence t, where the weight of aligning

s_i with t_j is z_{ij}. This problem can be solved in polynomial time using a dynamic programming algorithm (see [4]). Observe that such a common subsequence of the reversal of s and t corresponds to a maximal star (since the weights are non-negative). So, if such a maximum weight star, say S, has value greater than 1, the corresponding star inequality $z(S) \leq 1$ is violated by z. Conversely, if this value is less than or equal to 1, no star inequality is violated by the current solution z.

2.2 The Integrality of the LCS Polytope

Another way of proving that the polytope described in Theorem 1 is integral is by means of the following result shown by Fulkerson [5], Lovász [8] and Chvátal [3]. Given an undirected graph $G = (V, E)$, the **clique polytope** of G is defined as the convex hull of the incidence vectors of the cliques in G. We say a set $S \subseteq V$ is a **stable** set of G if S is a set of pairwise non-adjacent vertices. Consider the polytope:

$$P_C(G) = \{x \in \mathbb{R}^V \mid x_v \geq 0 \quad \text{for all } v \in V \text{ and} \\ x(S) \leq 1 \text{ for all stable set } S \subseteq V\}.$$

Chvátal [3] proved that the description above is indeed the description of the clique polytope of G (and therefore integral) if and only if G is a perfect graph. (A graph is perfect if for every induced subgraph H of G the maximum size of a clique in H is equal to the minimum coloring of H).

Now we observe that the LCS polytope defined in Theorem 1 is the clique polytope of a perfect graph. Let G_{st} be the graph defined as follows: its vertex set consists of all pairs (i, j) in E (as we defined in the beginning of Section 2), and two vertices are adjacent if and only if the corresponding pairs do not cross.

Thus, finding a longest common subsequence of s and t is equivalent to finding a largest clique in the graph G_{st}. Note that the maximal star inequalities $z(S) \leq 1$ that are facets of P_{LCS} are in fact inequalities of the form $x(S) \leq 1$ of the clique polytope $P_C(G_{st})$, where S is a maximal stable set of G_{st}. Thus, $P_{\text{LCS}} = P_C(G_{st})$, that is, P_{LCS} is precisely the clique polytope of the graph G_{st}. Hence, P_{LCS} is integral if and only if G_{st} is perfect.

Now it remains to show that the graph G_{st} is perfect. To this end, we show that its complement $\bar{G}_{st}$ is perfect (it is known that the complement of a perfect graph is also a perfect graph [8]). We claim that $\bar{G}_{st}$ is a comparability graph, and therefore a perfect graph. We recall that a **comparability graph** is a graph that has a transitive orientation, that is, an orientation such that the resulting directed graph satisfies a transitive law: whenever there exist directed edges (a, b) and (b, c), there must exist a directed edge (a, c).

It is not difficult to prove that $\bar{G}_{st}$ is a comparability graph. It suffices to note that the following orientation $\mathcal{G}_{st}$ of $\bar{G}_{st}$ is transitive: if $e = \{u, v\}$ is an edge of $\bar{G}_{st}$, where $u = (i, j)$ and $v = (k, \ell)$ (note that the pairs (i, j) and (k, ℓ) cross), then orient e from u to v if $i \leq k$ and $j \geq \ell$ (see Figure 1). It is immediate that if $u = (i, j)$, $v = (k, \ell)$ and $w = (p, q)$ are vertices of $\mathcal{G}_{st}$, and in this graph there

are arcs going from u to v and from v to w, then this graph has an arc going from u to w. This follows because we have $i \leq k$ and $j \geq \ell$ (as there is an arc from u to v) and we have $k \leq p$ and $\ell \geq q$ (as there is an arc from v to w) and therefore we have $i \leq k$ and $j \geq q$. Thus the orientation we defined is transitive.

Although the LCS polytope can be described as the clique polytope of the graph G_{st}, the proof of the complete description of P_{LCS} we have presented (by means of star inequalities) is interesting, as it is short and self-contained (it does not rely on the result concerning the clique polytope of perfect graphs).

3 Formulation for the RFLCS

Given two sequences s and t, the problem of finding a repetition-free LCS of s and t can be formulated as the following integer program.

$$\begin{array}{lrcll} \text{maximize} & \sum_{(i,j)\in E} z_{ij} & & & \\ \text{subject to} & z(E_a) & \leq & 1 & \text{for all symbol } a \text{ of the alphabet,} \\ & z_{ij} + z_{kl} & \leq & 1 & \text{for all } (i,j) \text{ and } (k,l) \text{ in } E \text{ that cross,} \\ & z_{ij} & \in & \{0,1\} & \text{for all } (i,j) \text{ in } E. \end{array} \tag{2}$$

As in the case of LCS, we can associate with RFLCS the following polytope:

$$P_{\text{RFLCS}} := \text{conv}\{z \in \{0,1\}^E \mid z \text{ represents a repetition-free common subsequence of } s \text{ and } t\}.$$

It is easy to see that the maximal star inequalities are valid for P_{RFLCS}. However, these inequalities are not facet-defining.

For a set $S \subseteq E$, let $S_a = S \cap (s(a) \times t(a))$, where a is a symbol of the alphabet. We say that a set $S \subseteq E$ is an **extended star** if, for every two distinct symbols a and b, each pair in S_a crosses all pairs in S_b. We prove in the next theorem that if S is a maximal extended star then the corresponding star inequality $z(S) \leq 1$ is facet-defining for P_{RFLCS}.

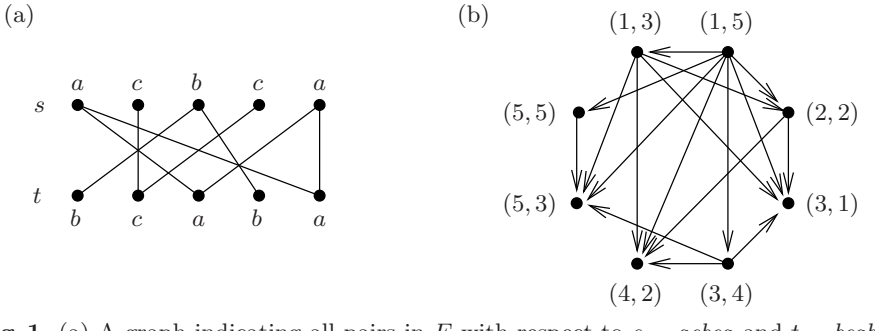

Fig. 1. (a) A graph indicating all pairs in E with respect to $s = acbca$ and $t = bcaba$. (b) The graph $\mathcal{G}_{st}$ for s and t.

Theorem 2. *Let S be a maximal extended star. The inequality $z(S) \leq 1$ defines a facet of P_{RFLCS}.*

Proof. It is immediate that the inequality $z(S) \leq 1$ is valid for P_{RFLCS}. In order to prove that it defines a facet of P_{RFLCS}, consider an inequality $az \leq \alpha$ that defines a facet F of P_{RFLCS} and suppose

$$\{z \in \mathbb{R}^E \mid z(S) = 1\} \subseteq F := \{z \in R^E \mid az = \alpha\}.$$

Consider a pair (i, j) in $E_a \setminus S$ for some symbol a. Since S is a maximal extended star, there is a pair (k, ℓ) in $S \cap E_b$, with $b \neq a$, that does not cross (i, j). Then, $e^{k\ell}$ and $e^{k\ell} + e^{ij}$ are both in F, and this implies that $a_{ij} = 0$ for every (i, j) in $E_a \setminus S$. Since this holds for every symbol a, we conclude that $a_{ij} = 0$ for every (i, j) in $E \setminus S$.

Now observe that for every (i, j) in S we have that $e^{ij} \in F$. Thus, $a_{ij} = \alpha$ for every (i, j) in S. Hence, the inequality $az \leq \alpha$ can be rewritten as $\alpha z(S) \leq \alpha$ and we conclude that the extended star inequality $z(S) \leq 1$ is facet-defining. □

3.1 The Separation Problem for the Extended Star Inequalities

We show now that a polynomial-time algorithm using a dynamic programming algorithm can be used to solve the separation problem for the extended star inequalities. The idea of the algorithm is similar to the separation algorithm shown in the previous section. Let z be a fractional point we intend to separate. Every extended star (see Figure 2(a)) corresponds to an alignment as the one depicted in Figure 2(b), between the sequence s reversed, which we denote by s^r, and the sequence t. In the alignment shown in Figure 2(b) each star corresponds to some particular symbol (that is, only pairs corresponding to the same symbol can cross).

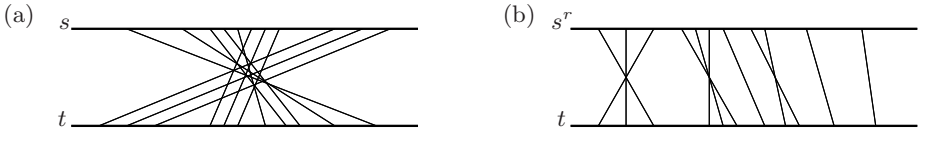

Fig. 2. (a) An extended star between s and t. (b) The corresponding alignment between s^r and t.

The separation algorithm has to find an extended star whose corresponding inequality is violated. For that, it looks for a maximum weight alignment, as the one in Figure 2(b), between s^r and t. The weights for the alignments are given by z. Such an alignment allows crossings only among pairs corresponding to a common symbol. One can find such an alignment by a dynamic programming algorithm.

Indeed, let $mw(n,m)$ be the weight of such a maximum weight alignment between the sequences $s^r[1\,..\,n]$ and $t[1\,..\,m]$. The following recurrence holds for $mw(n,m)$:

$$mw(n,m) = \begin{cases} 0 & \text{if } n = 0 \text{ or } m = 0, \\ \max\{mw(n-1,m), mw(n,m-1)\} & \text{if } s^r[n] \neq t[m], \\ \max_{1 \leq i,j \leq n}\{mw(i-1,j-1) + b_a(i,n,j,m)\} & \text{if } s^r[n] = t[m] = a, \end{cases}$$

where $b_a(i,n,j,m) = \sum\{z_{kl} : i \leq k \leq n,\ j \leq l \leq m,\ (k,l) \in E_a\}$.

It is not hard to see that $mw(n,m)$ can be computed by an $O(n^2m^2)$ algorithm. It is also not hard to derive an algorithm that finds an alignment between s^r and t of weight $mw(n,m)$. If $mw(n,m) < 1$, then such an alignment corresponds to an extended star that is violated. If $mw(n,m) \geq 1$, then no extended star is violated.

4 Computational Results

We implemented a branch and cut algorithm for the RFLCS using the exact separation procedure for the extended star inequalities shown in the previous section. We considered instances consisting of two sequences of equal length $n = 512$, and with alphabet sizes $\frac{1}{8}n, \frac{2}{8}n, \ldots, \frac{7}{8}n$. The instances are generated randomly with uniform probability. To solve the linear programming relaxation we use the GLPK (Gnu Linear Programming Kit), an ANSI C set of routines organized in the form of a callable library.

In the first experiment we tested our branch and cut algorithm on 10 random instances, each one with the above mentioned alphabet size. We limited the running time for each instance to one hour. The results obtained are summarized in Table 1. In each line the results shown correspond to 10 instances with the given alphabet size. The first column shows the *alphabet size*. In the second column we indicate the *average computational time* for the 10 instances; and we also indicate in parenthesis the *minimum and maximum running time*. The next column shows the *average number of cuts* added to the linear program. Then, we indicate how many of the instances could be solved to optimality within one hour. The next two columns show the *average lower and upper bound* achieved within the fixed time limit. Finally, the last two columns exhibit the *average number of active nodes at the moment the program was interrupted* and the *average number of nodes visited in the branch and bound tree* during the execution of the program.

In Table 1, we note that it becomes more difficult to solve instances in which the alphabet size is small compared to the length of the sequences (up to $\frac{3}{8}n$). This difficulty might be explained by the fact that when the alphabet is small then the number of repetitions in the sequences is large, and this implies that such instances may have many feasible solutions. It is interesting to note that we could solve all instances to optimality when the size of the alphabet is at least $\frac{1}{2}n$. The average number of nodes in the branch and cut tree (last column)

Table 1. Computational results with extended star inequalities

alph. size	time	(min/max)	cuts	exact	lb	ub	active	nodes
64	3604.5	(3600/3609)	1162.5	0	44.4	63.9	1.0	0.0
128	3601.6	(3600/3604)	1927.7	0	56.8	75.2	1.0	0.0
192	2761.8	(762/3600)	2532.1	6	58.8	59.7	1.7	10.7
256	604.4	(224/1582)	1474.0	10	54.2	54.2	0.0	5.6
320	245.9	(93/498)	1077.1	10	46.5	46.5	0.0	1.8
384	113.0	(72/200)	797.0	10	43.0	43.0	0.0	1.0
448	76.0	(52/108)	660.6	10	40.7	40.7	0.0	1.4

indicate that most of the work was done in the root node. For the instances with small alphabet we achieve an average gap under 50% to the value of the best solution found in one hour.

In order to investigate the strength of the extended star inequalities we ran GLPK with the integer programming formulation (2). We set the parameters of the solver to use clique and Gomory cuts (these are the best parameters we could found). The results for the easier instances (with alphabet size at least $\frac{1}{2}n$) are summarized in Table 2.

Table 2. Branch and bound with formulation (2)

alph. size	time	(min/max)	cuts	exact	lb	ub	active	nodes
256	3669.3	(3642/3724)	300.0	0	13.3	56.0	29.0	3.0
320	2739.3	(771/3631)	282.6	4	45.7	47.3	45.6	91.7
384	1159.9	(534/2886)	271.9	10	43.0	43.0	0.0	87.2
448	746.1	(254/2537)	236.5	10	40.7	40.7	0.0	117.0

As shown in Table 2, we could solve all 10 instances only for alphabet sizes 384 and 448. It is interesting to observe that in these cases the average time to solve the instances is around 10 times larger than the time we needed using the facet-defining inequalities and our separation procedure. In the other cases, we could solve only 4 instances to optimality, while using the extended star inequalities we could solve all instances within one hour.

5 Conclusion

The LCS is a well-known problem that has many nice applications. Perhaps because of the fact that a polynomial-time algorithm for this problem is known, a polyhedral approach to this problem has not been considered in the literature. We think the polyhedral results we have shown for the LCS in this paper have many interesting aspects: we give a complete and irredundant description of the polytope P_{LCS} we have associated to it and show that the separation problem for

this polytope is polynomial. Furthermore, we have given an alternative proof of the integrality of this polytope by showing its relation with the clique polytope of the graph G_{st}, which we show to be perfect. The repetition-free version of LCS, an NP-hard problem, also has applications in the study of genomes. This variant has been less investigated. The polyhedral approach and the computational results we presented for the RFLCS show that this approach is an improvement to pure branch and bound strategies or simple heuristics. Further facets of this polytope, as well as some heuristics, may be incorporated to the present code, leading to improvements that can be useful to solve some larger instances of the problem.

Acknowledgements

We would like to thank José Coelho de Pina for some valuable comments related to the contents of Section 2.2. We also thank the financial support from CNPq (Proc. 490333/04, 307011/03-8, 308138/04-0, 301919/04-6), ProNEx - Fapesp/CNPq Proc. No. 2003/09925-5 and Fapesp (Proc. 07/54282-6), Brazil.

References

1. Adi, S.S., Braga, M., Fernandes, C.G., Ferreira, C.E., Martinez, F.H.V., Sagot, M.-F., Stefanes, M.A., Tjandraatmadja, C., Wakabayashi, Y.: Repetition-free longest common subsequence. In: Proc. IV Latin-American Algorithms, Graphs and Optimization Symposium (to appear) (2007)
2. Bonizzoni, P., Della Vedova, G., Dondi, R., Fertin, G., Vialette, S.: Exemplar longest common subsequence. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3992, pp. 622–629. Springer, Heidelberg (2006)
3. Chvátal, V.: On certain polytopes associated with graphs. J. Combinatorial Theory Ser. B 18, 138–154 (1975)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press, Cambridge (2001)
5. Fulkerson, D.R.: Anti-blocking polyhedra. J. Combinatorial Theory Ser. B 12, 50–71 (1972)
6. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1, 169–197 (1981)
7. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Heidelberg (1988)
8. Lovász, L.: Normal hypergraphs and the perfect graph conjecture. Discrete Math. 2(3), 253–267 (1972)
9. Sankoff, D.: Genome rearrangement with gene families. Bioinformatics 15(11), 909–917 (1999)