Maximum Letter-Duplicated Subsequence (short paper)*

Riccardo Dondi[1,*,†], Mehdi Hosseinzadeh[1,†] and Alexandru Popa[2,†]

[1]*Università degli Studi di Bergamo, Bergamo, Italy*
[2]*University of Bucharest, Romania*

Abstract
In this contribution we consider Max-LL-DUP, an optimization problem that asks for a letter-duplicated subsequence of an input string that contains the maximum number of letters of the alphabet over which the input string is defined. When each letter has at most four occurrences in the input string, we prove that the problem is APX-hard. Then, we give a linear-time algorithm when each letter has at most three occurrences in the input string.

Keywords
String Algorithms, LCS, APX-hardness

1. Introduction

Detecting gene duplications in genomes is a fundamental problem in biology. Recently, some approaches have modeled this problem as the identification of a subsequence in a given input string that represents a genome [1, 2, 3, 4]. The approach proposed by [3] aims to detect *tandem duplications*, in particular when mutations may occur after duplications, and looks for a *letter-duplicated subsequence* of a given string. A letter-duplicated subsequence consists of the concatenation of substrings on a single letter that have length at least two. In [3] it is introduced a decision problem that asks whether there exists a letter-duplicated subsequence of a given string that contains each letter of the alphabet over which the input string is defined. The problem has been shown to be NP-complete, even when each letter has at most six occurrences, while it is polynomial-time solvable when each letter has at most three occurrences [3].

Since in many cases there is no letter-duplicated subsequence of a string that contains all the letters of the alphabet Σ, we relax this constraint and we introduce and study an optimization problem, called Max-LL-DUP, that asks for the maximum number of letters of Σ that are contained in a letter-duplicated subsequence of S. In Section 2, we start by introducing some definitions and by providing the formal definition of Max-LL-DUP. In Section 3, we prove that Max-LL-DUP is APX-hard even if each letter has at most four occurrences in the input string,

ICTCS'24: Italian Conference on Theoretical Computer Science, September 11–13, 2024, Torino, Italy
*Corresponding author.
†These authors contributed equally.
riccardo.dondi@unibg.it (R. Dondi); m.hosseinzadeh@unibg.it (M. Hosseinzadeh); alexandru.popa@fmi.unibuc.ro (A. Popa)
0000-0002-6124-2965 (R. Dondi); 0000-0003-3275-6286 (M. Hosseinzadeh); 0000-0003-3364-1210 (A. Popa)

and in Section 4, we present a polynomial-time algorithm when each letter has at most three occurrences. We conclude this contribution with some future directions.

2. Definitions

Given a string S, we denote by $|S|$ its length. We denote by $S[i]$ the letter of S in position i, with $1 \leq i \leq |S|$. Given two positions i and j, with $1 \leq i \leq j \leq |S|$, we denote by $S[i, j]$ the substring of S between i and j. If $i = j$, then $S[i, i]$ is the letter $S[i]$. Given a letter $x \in \Sigma$, we denote by x^i, with $i \geq 1$, a string consisting of i occurrences of x. A subsequence S' of S is obtained by removing some letters, possibly none, of S. Now, we introduce the definition of letter-duplicated subsequence.

Definition 1. *Given a string S on alphabet Σ, a subsequence S' of S is letter-duplicated if $S' = x_1^{a_1} \dots x_z^{a_z}$ where $x_i \in \Sigma$ and $a_i \geq 2$, with $1 \leq i \leq z$, and $x_i \neq x_{i+1}$, with $1 \leq i \leq z-1$.*

A letter-duplicated subsequence of S of length two is called a *pair*. Consider two pairs, $S[i]S[j]$, with $1 \leq i < j \leq |S|$, and $S[l]S[h]$ with $1 \leq l < h \leq |S|$, where $S[i] = S[j] \neq S[l] = S[h]$. The two pairs are *crossing* when, assuming $i < l$ it holds that: $1 \leq i < l < j < h \leq |S|$ or $1 \leq i < l < h < j \leq |S|$. Note that at most one of two crossing pairs can belong to a letter-dupplicated subsequence of S. Now, we define the problem we are interested into.

Problem 1. Max-LL-DUP
Input: *A string S on alphabet Σ.*
Output: *A letter-duplicated subsequence S' that contains the maximum number of letters in Σ.*

Since the objective function of Max-LL-DUP is the number of letters appearing in a solution and not the length of the subsequence, we assume in the following that each solution of Max-LL-DUP contains at most one pair for each letter in Σ. We denote by x-Max-LL-DUP, $1 \leq x \leq |S|$, the restriction of Max-LL-DUP where each letter appears at most x times in the input sequence.

3. APX-Hardness of 4-Max-LL-DUP

In this section, we show that 4-Max-LL-DUP is APX-hard. We prove the result by giving a reduction from the Maximum Independent Set on cubic graphs (3-MIS). Recall that in a cubic graph each vertex has degree 3 and that, given a cubic graph $G = (V, E)$, 3-MIS asks for a subset $V' \subseteq V$ of maximum cardinality such that no two vertices in V' are adjacent.

Given an instance $G = (V, E)$ of 3-MIS we define a corresponding instance of 4-Max-LL-DUP as follows. We start by defining the alphabet Σ and the string S. The alphabet Σ is defined as follows:

$$\Sigma = \{a_i, b_i : v_i \in V, 1 \leq i \leq |V|\} \cup \{e_{i,j}, e_{j,i} : \{v_i, v_j\} \in E, 1 \leq i, j \leq |V|\}.$$

Given a vertex $v_i \in V$, the letters in $\{a_i, b_i, e_{i,w}, e_{i,z}, e_{i,j} \in \Sigma : v_i \in V\}$ are called the letters *related to* v_i. Now, S consists of different substrings. For each v_i, $1 \leq i \leq |V|$, S contains the following substrings $S_1(v_i)$, $S_2(v_i)$:

$$S_1(v_i) = a_i b_i a_i b_i \quad S_2(v_i) = a_i e_{i,j} e_{i,j} e_{i,z} e_{i,z} e_{i,w} e_{i,w} a_i.$$

For each $\{v_i, v_j\} \in E$, with $1 \leq i < j \leq |V|$, the following substring $S(i,j)$ is defined:

$$S(i,j) = e_{i,j}e_{j,i}e_{i,j}e_{j,i}.$$

The input string S is obtained by concatenating first the substrings $S_1(v_i)$, $1 \leq i \leq |V|$, in lexicographic order, then $S_2(v_i)$, $1 \leq i \leq |V|$, in lexicographic order, then $S(v_i, v_j)$, where $\{v_i, v_j\} \in E$ and $1 \leq i < j \leq |V|$. First, we show that S is indeed an instance of 4-Max-LL-DUP.

Lemma 1. *Let G be an instance of* 3-MIS *and S be a corresponding string built by the reduction, then each letter of Σ has at most four occurrences in S.*

Proof. We show that each letter in Σ has at most four occurrences in S. Letter b_i, $1 \leq i \leq |V|$, has two occurrences in S, since it appears only in substring $S_1(v_i)$. Letter a_i, $1 \leq i \leq |V|$, has four occurrences in S, since it appears (twice) only in substrings $S_1(v_i)$, $S_2(v_i)$. Letter $e_{i,j}$, $1 \leq i, j \leq |V|$, has four occurrences in S, since it appears only in substrings $S_2(v_i)$ (twice) and $S(i,j)$ (twice). □

Now, we prove that starting from an independent set of G, we can compute in polynomial time a solution of 4-Max-LL-DUP on instance S.

Lemma 2. *Let G be an instance of* 3-MIS *and S be the corresponding instance of* 4-Max-LL-DUP*. Given an independent set V' of G, we can compute in polynomial time a solution of* 4-Max-LL-DUP *on instance S that contains $5|V'| + 4|V \setminus V'|$ pairs.*

Proof. Given an independent set V' of G, define a solution S' of 4-Max-LL-DUP as follows:

- For each $v_i \in V'$, add to S' pair b_ib_i in $S_1(v_i)$, and a_ia_i in $S_2(v_i)$; moreover, for each j (assume w.l.o.g $i < j$), add to S' the pair $e_{i,j}, e_{i,j}$, in $S(i,j)$. Hence five pairs of letters related to v_i are in S' (recall that G is a cubic graph).

- For each $v_i \in V \setminus V'$, add to S' the pair a_ia_i in $S_1(v_i)$, and the pairs $e_{i,j}, e_{i,j}$, $e_{i,z}, e_{i,z}$ $e_{i,w}, e_{i,w}$ in $S_2(v_i)$, where $\{v_i, v_j\}$, $\{v_i, v_h\}$, $\{v_i, v_w\}$ are the edges of G incident in v_i. Hence in this case four pairs of letters related to v_i are in S'.
S' contains $5|V'| + 4|V \setminus V'|$ pairs of letters, thus concluding the proof. □

Lemma 3. *Let G be an instance of* 3-MIS *and S be a corresponding instance of* 4-Max-LL-DUP*. Given a solution of* 4-Max-LL-DUP *on instance S that contains $5q + 4(|V| - q)$ pairs, we can compute in polynomial time an independent set V' of G with $|V'| \geq q$.*

Proof. Consider a solution S' of 4-Max-LL-DUP that contains $5q + 4(|V| - q)$ pairs. First, we notice that there exists a set $V' \subseteq V$ of vertices in G such that $|V'| \geq q$ and such that, for each $v_i \in V'$, S' contains all the five pairs of letters related to v_i, that is b_ib_i, a_ia_i and for each j, with $\{v_i, v_j\} \in E$ (assume w.l.o.g $i < j$), $e_{i,j}, e_{i,j}$. Indeed, assume aiming at a contradiction that this is not the case. Since an instance of 4-Max-LL-DUP contains 5 pairs of letters related to each vertex $v_i \in V$, then S' can contain at most $5(q-1) + 4(|V| - q + 1) < 5q + 4(|V| - q)$ pairs. Consider now $v_i \in V'$. Then S' contains 5 pairs of letters related to v_i. We claim that for each $\{v_i, v_j\} \in E$ (we assume w.l.o.g. $i < j$), S' contains one pair $e_{i,j}e_{i,j}$ in each substring $S(i,j)$.

Assume this is not the case, then S' must contain a pair $e_{i,j}e_{i,j}$ in a substring of S different from $S(i,j)$, that is in $S_2(v_i)$. Thus S' cannot contain pair $a_i a_i$ in $S_2(v_i)$, since it is crossing with pair $e_{i,j}e_{i,j}$. Since S' can contain at most one of the crossing pairs $a_i a_i$, $b_i b_i$ of $S_1(v_i)$, it follows that S' contains at most 4 pairs of letters related to v_i, which contradicts the assumption that $v_i \in V'$.

Consider two vertices $v_i, v_z \in V'$, with $1 \leq i < z \leq |V|$. Since S' contains 5 pairs of letters related to each of v_i, v_z, then for $x \in \{i, z\}$ S' must contain a pair $e_{x,j}e_{x,j}$ in each $S(x,j)$. Then $\{v_i, v_z\} \notin E$, otherwise the pairs $e_{i,z}e_{i,z}$ and $e_{z,i}e_{z,i}$ would be crossing in $S(i,z)$ and could not be both in S'. Thus no edge $\{v_i, v_z\} \in E$ and V' is an independent set of G of size at least q. □

We can conclude with the main result of this section.

Theorem 1. *4-Max-LL-DUP is APX-hard.*

Proof. Due to the results of Lemma 2 and in Lemma 3, we have described an L-reduction from 3-MIS to 4-Max-LL-DUP (see [5] for details on L-reductions). Indeed, an optimal solution of 3-MIS, denoted by OPT(3-MIS) contains at least $\frac{|V|}{4}$ vertices (a greedy algorithm that selects a vertex v, adds it to the independent set and remove v and its neighbors, computes an independent set of size at least $\frac{|V|}{4}$). Hence, denoted by OPT(4-Max-LL-DUP) an optimal solution of 4-Max-LL-DUP, by Lemma 2 there exists a constant α such that:

$$OPT(\text{4-Max-LL-DUP}) \leq \alpha\, OPT(\text{3-MIS}).$$

Moreover, given an approximated solution of 4-Max-LL-DUP on instance S of value APX(4-Max-LL-DUP) by Lemma 3 we can compute in polynomial time an approximated solution of 3-MIS on instance G containing APX(3-MIS) vertices, such that there exists a constant β and it holds that

$$OPT(\text{3-MIS}) - APX(\text{3-MIS}) \leq \beta\,(OPT(\text{4-Max-LL-DUP}) - APX(\text{4-Max-LL-DUP})).$$

It follows that have designed an L-reduction from 3-MIS to 4-Max-LL-DUP. Since 3-MIS is APX-hard [6], then also 4-Max-LL-DUP is APX-hard. □

4. A Linear-Time Algorithm for 3-Max-LL-DUP

We now show that 3-Max-LL-DUP is solvable in linear time. We present how to compute the value of an optimal solution of 3-Max-LL-DUP, the algorithm can be easily adapted to compute the actual solution.

Consider a function $D[j]$, with $0 \leq j \leq |S|$, that represents the length of an optimal solution of 3-Max-LL-DUP on instance $S[1,j]$. Function $D[j]$ can be computed with the following recurrence. For $j \in 0, 1$, $D[j] = 0$; for $j \geq 2$, we have that:

$$D[j] = \max \begin{cases} D[j-1] & \\ D[r_a - 1] + 1 & \text{where } S[j] = a \text{ and } r_a \text{ is the rightmost} \\ & \text{occurrence of } a \text{ in } S[1, j-1]. \end{cases} \quad (1)$$

We start by considering the correctness of the recurrence.

Lemma 4. *There exists a solution of* 3-Max-LL-DUP *on* $S[1, j]$ *consisting of* h *pairs if and only if* $D[j] = h$.

Proof. We prove the lemma by induction. In the base case, for $j = 0, 1$, there is no pair in $S[1, 1]$ and by definition, $D[0] = 0$ and $D[1] = 0$. Assume that the lemma holds for a string $S[1, z]$, with $z < j$, we show that it holds for j. Consider a solution S' of 3-Max-LL-DUP on instance $S[1, j]$ of size h. If S' does not include $S[j]$, then S' is a solution of 3-Max-LL-DUP on instance $S[1, j - 1]$ of size h and by induction hypothesis $D[j - 1] \geq h$, thus $D[j] \geq h$. Assume that S' includes $S[j]$, then it must include also another occurrence of $a = S[j]$ in $S[1, j - 1]$ we can assume that it is $S[r_a]$. Thus S' contains $h - 1$ pairs of $S[1, r_a - 1]$, thus by induction hypothesis $D[r_a - 1] \geq h - 1$ and, by Recurrence 1, $D[j] \geq h$. Note indeed that, since S contains at most three occurrences of each letter, then no solution of 3-Max-LL-DUP on instance $S[1, r_a - 1]$ contains a.

Assume that $D[j] = h$, as computed by Recurrence 1. If $D[j] = D[j - 1] = h$, then by induction hypothesis there exists a solution S' of 3-Max-LL-DUP on instance $S[1, j - 1]$, hence also on $S[1, j]$, of size at least h. Assume that $D[j] = D[r_a - 1] + 1$, then $D[r_a - 1] = h - 1$. It follows by induction hypothesis that there exists a solution S'' of 3-Max-LL-DUP on instance $S[1, r_a - 1]$ of size at least $h - 1$. By adding the pair $(S[r_a], S[j])$ to S'' we obtain a solution of 3-Max-LL-DUP on instance $S[1, j]$ of size at least h. □

Next, we show that D can be computed in linear time. Notice that D contains a linear number of entries. Each $D[j]$ can be updated in constant time as described in the following. First, assume that we have computed in linear time an array A of size $|\Sigma|$ and indexed by Σ, that for each $a \in \Sigma$ stores the positions of the two leftmost occurrences of a (recall that each letter has at most three occurrences in S; note that we can assume that each letter of Σ has at least two occurrences in S, otherwise we can delete it from S). Array A can be computed in linear time, scanning S from left to right. Using array A, given $a = S[j]$, we can compute in constant time the value of r_a, since it is the maximum value smaller than j contained in the entry of A associated with a. Then $D[j]$ can be computed in constant time, since we have to consider two values, namely $D[j - 1]$, that can be computed in constant time, and $D[r_a - 1]$, that can be computed in constant time using array A. Finally, the length of an optimal solution of 3-Max-LL-DUP on instance S is stored in $D[|S|]$.

5. Conclusion

We have introduced Max-LL-DUP, a problem that asks for a letter-duplicated subsequence of an input string that contains the maximum number of letters of the alphabet. An interesting future direction is to further study the approximation complexity of the problem, in particular, the design of constant-factor approximation algorithms.

References

[1] S. Schrinner, M. Goel, M. Wulfert, P. Spohr, K. Schneeberger, G. W. Klau, Using the longest run subsequence problem within homology-based scaffolding, Algorithms

Mol. Biol. 16 (2021) 11. URL: https://doi.org/10.1186/s13015-021-00191-8. doi:10.1186/S13015-021-00191-8.

[2] R. Dondi, F. Sikora, The longest run subsequence problem: Further complexity results, in: P. Gawrychowski, T. Starikovskaya (Eds.), 32nd Annual Symposium on Combinatorial Pattern Matching, CPM 2021, July 5-7, 2021, Wrocław, Poland, volume 191 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 14:1–14:15. URL: https://doi.org/10.4230/LIPIcs.CPM.2021.14. doi:10.4230/LIPICS.CPM.2021.14.

[3] W. Lai, A. Liyanage, B. Zhu, P. Zou, Beyond the longest letter-duplicated subsequence problem, in: H. Bannai, J. Holub (Eds.), 33rd Annual Symposium on Combinatorial Pattern Matching, CPM 2022, June 27-29, 2022, Prague, Czech Republic, volume 223 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 7:1–7:12. doi:10.4230/LIPIcs.CPM.2022.7.

[4] M. Lafond, W. Lai, A. Liyanage, B. Zhu, The longest subsequence-repeated subsequence problem, in: W. Wu, J. Guo (Eds.), Combinatorial Optimization and Applications - 17th International Conference, COCOA 2023, Hawaii, HI, USA, December 15-17, 2023, Proceedings, Part I, volume 14461 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 446–458. URL: https://doi.org/10.1007/978-3-031-49611-0_32. doi:10.1007/978-3-031-49611-0_32.

[5] D. P. Williamson, D. B. Shmoys, The Design of Approximation Algorithms, Cambridge University Press, 2011. URL: http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE.

[6] P. Alimonti, V. Kann, Some apx-completeness results for cubic graphs, Theor. Comput. Sci. 237 (2000) 123–134. URL: https://doi.org/10.1016/S0304-3975(98)00158-3. doi:10.1016/S0304-3975(98)00158-3.