Control and Cybernetics

vol.**33** (2004) No. 1

On the complexity of the Double Digest Problem[1]

by

J. Błażewicz[2], E. Burke,[3] M. Jaroszewski[2], M. Kasprzak[2]
B. Paliswiat[4], P. Pryputniewicz[4]

[2] Institute of Computing Science, Poznań University of Technology,
Piotrowo 3A, 60-965 Poznań, Poland.
Institute of Bioorganic Chemistry, Polish Academy of Sciences,
Noskowskiego 12/14, 61-704 Poznań, Poland.

[3] University of Nottingham

[4] Institute of Computing Science, Poznań University of Technology,
Piotrowo 3A, 60-965 Poznań, Poland.

Abstract: The problem of genome mapping is considered. It is showed that the Double Digest Problem (DDP) is NP-hard in its search version, while its decision version is trivially easy.
Keywords: computational complexity, DNA restriction maps.

1. Introduction

Creation of physical maps is one of the basic steps in genome studies. Such a map of a DNA strand contains information about locations of markers, which are short, specific subsequences. There are many ways of constructing a physical map. One of them is based upon splitting a target strand into many shorter ones, called clones, so that they would mutually overlap. Next, information about each clone is received. This information consists of knowledge about the sets of short and unique DNA fragments, called probes, that bind to each clone during the hybridization process. The methods and algorithms based on the foregoing approach are presented, in particular, in Alizadeh et al. (1995), Setubal and Meidanis (1997).

Another way of creating a physical map, consists in the digestion of DNA with restriction enzymes. These enzymes cut DNA molecules within specific,

[1]The work was partially supported by the KBN (Polish State Committee for Scientific Research) grant. This paper was written while the first author was at the University of Nottingham under ESPRC grant, whose help is greatly appreciated.

short patterns of nucleotides called restriction sites. After the digestion, the lengths of obtained fragments are measured and the original ordering of these fragments must be reconstructed, and this is the place where computer science methods come to the effect. In practice, several variants of this approach are used. Two of the best known are *the partial digest* and *the double digest*.

In the former variant only one restriction enzyme is used, see Skiena et al. (1990), Skiena and Sundaram (1994). After multiplying, the target DNA is divided into a few sets. Molecules from each set are digested by the same enzyme but the time span allowed for digestion differs among sets. The reaction time spans should be determined in such a way that, in the case of ideal experiment, in one of the sets most of DNA strands are cut at most once (of course, the site need not be the same for different molecules), in the other set - exactly twice, and so on. The longest reaction time span must suffice to let the enzyme cut all molecules at all occurrences of the restriction site. As a result, one gets three collections of restriction fragments, whose lengths are then measured during a gel electrophoresis process. The problem of restriction map construction based upon the presented biochemical experiment is known as PDP (the Partial Digest Problem). The computational complexity of PDP is an open question (Setubal and Meidanis, 1997). Recently, an approach has been designed which reduces the complexity of the biochemical stage to two digestions only (the very short time span and the very long one), thus resulting in definition of the so called Simplified Partial Digest Problem (SPDP), see Błażewicz et al. (2001). The method compares favorably with the approaches existing so far.

In the latter variant (i.e. the double digest) two restiction enzymes are used for digestion. They are labeled as a and b. The target DNA is multiplied in the PCR reaction and the obtained copies are divided into three sets. Molecules from the first set are digested by one enzyme (a), molecules from the second set are digested by the other enzyme (b), while molecules from the third one are cut by both enzymes. All digestions are complete, since the time span of each reaction is long enough to let the enzyme cut the target strands in each occurrence of the restriction site. As a result one gets three collections of shorter DNA fragments that correspond to the three digestion processes. At the end the lengths of obtained fragments are measured during a gel electrophoresis process and recorded as three multisets. On the basis of this data the original locations of restriction sites on the target DNA should be reconstructed. The combinatorial problem based upon the described ideal biochemical experiment is known as the Double Digest Problem (DDP). The proof of the NP-completeness of DDP (its decision version) has been presented in Goldstein and Waterman (1987), however, it turned out to be incorrect and the decision version of the problem appears to be trivially easy.

In this paper, we present an appropriate proof of the NP-hardness of the search version of Double Digest Problem assuming that restriction sites recognized by enzymes used for digestion processes do not cover one another. Thus, both enzymes cut DNA molecules in different loci.

In Section 2 mathematical formulations of DDP in decision and search versions are given. In Section 3 the proof of the NP-hardness is presented. Section 4 concludes the paper.

2. Problem formulation

Let n denote the number of restriction sites recognized by enzyme a and let m denote the number of restriction sites recognized by enzyme b. Moreover, let l denote a length of a target DNA molecule. Next let $A = \{a_1, \ldots, a_{n+1}\}$, $B = \{b_1, \ldots, b_{m+1}\}$ and $C = \{c_1, \ldots, c_{m+n+1}\}$ denote the non-decreasingly ordered multisets of the lengths of DNA fragments obtained as a result of digestion with, respectively, enzyme a, enzyme b, and enzymes a and b combined. We will formulate now the Double Digest Problem in both versions: the decision and the search version.

PROBLEM 1 *Double Digest Problem in the decision version:*
Instance: *A, B, C and l, m, n as defined above.*
Question: *Is it possible to find any permutation σ of multiset A and any permutation μ of multiset B such that the list of fragment lengths, obtained on the segment of length l as the result of composing σ and μ, ordered non-decreasingly, would be identical to the list $\langle c_1, \ldots, c_{m+n+1}\rangle$?*

PROBLEM 2 *Double Digest Problem in the search version:*
Instance: *A, B, C and l, m, n as defined above.*
Goal: *Find any permutation σ of multiset A and any permutation μ of multiset B such that the list of fragment lengths, obtained on the segment of length l as the result of composing σ and μ, ordered non-decreasingly, would be identical to the list $\langle c_1, \ldots, c_{m+n+1}\rangle$.*

In what follows, we will denote by DDP(d) and DDP(s), respectively, the decision and search versions of the Double Digest Problem. Obviously, DDP(d) is easy, since for any error-free input data coming from the biochemical experiment there must exist permutations σ and μ for which the desired property holds, because the input data reflect a real, existing physical map. More formally:

$$D_{DDP(d)} = Y_{DDP(d)}, \tag{1}$$

where $D_{DDP(d)}$ denotes the set of all instances of DDP(d) and $Y_{DDP(d)}$ denotes the set of all instances of DDP(d) with the answer 'yes'. As a result, the NP-completeness proof of DDP(d) given in Goldstein and Waterman (1987) is not correct. On the other hand, the problem of finding permutations σ and μ obeying the assumptions of the problem considered is computationally hard. This will be proved in the next section.

3. Proof of the NP-hardness of the search version of the Double Digest Problem

At the beginning we will formulate some problems that will be used in the following proofs.

PROBLEM 3 *Partition (only decision version):*
Instance: *set* $S = \{s_1, s_2, \ldots, s_k\}$ *of* k *elements and multiset* $F = \{f(s_1), f(s_2), \ldots, f(s_k)\}$ *of sizes, where* $f(s_i) > 0$.
Question: *Is there a subset* $S' \subset S$ *such that the following dependence holds:*

$$\Sigma_{s \in S'} f(s) = \Sigma_{s \in S - S'} f(s)?$$

Let n_q denote the number of restriction sites recognized by enzyme a_q and let m_q denote the number of restriction sites recognized by enzyme b_q. Moreover, let l_q denote the length of a target DNA molecule. Next, let $A_q = \{a_{q1}, \ldots, a_{q(n_q+1)}\}$ and $B_q = \{b_{q1}, \ldots, b_{q(m_q+1)}\}$ denote, non-decreasingly ordered, multisets of the lengths of DNA fragments obtained as a result of digestion with, respectively, enzymes a_q and b_q. In addition, let $C_q = \{c_{q1}, \ldots, c_{q(m_q+n_q+1)}\}$ denote any multiset, sorted in non-decreasing order. The quasi Double Digest Problem in both versions: the decision one and the search one is formulated below.

PROBLEM 4 *quasi Double Digest Problem in the decision version (qDDP(d) for short):*
Instance: A_q, B_q, C_q *and* l_q, m_q, n_q *as defined above.*
Question: *Is it possible to find any permutation* σ_q *of multiset* A_q *and any permutation* μ_q *of multiset* B_q *such that the list of fragment lengths, obtained on the segment of length* l_q *as the result of composing* σ_q *and* μ_q*, ordered non-decreasingly, would be identical to the list* $\langle c_{q1}, \ldots, c_{q(m_q+n_q+1)} \rangle$*?*

PROBLEM 5 *quasi Double Digest Problem in the search version (qDDP(s) for short):*
Instance: A_q, B_q, C_q *and* l_q, m_q, n_q *as defined above.*
Goal: *Find any permutation* σ_q *of multiset* A_q *and any permutation* μ_q *of multiset* B_q *such that the list of fragment lengths, obtained on the segment of length* l_q *as the result of composing* σ_q *and* μ_q*, ordered non-decreasingly, would be identical to the list* $\langle c_{q1}, \ldots, c_{q(m_q+n_q+1)} \rangle$.

We see that due to the form of C_q, the following equation holds for both qDDP(d) and qDDP(s):

$$|C_q| = |A_q| + |B_q| - 1. \tag{2}$$

Now, our aim is to prove that the Double Digest Problem in its search version is NP-hard. We will do this by proving the following facts:

$$PARTITION \propto_T qDDP(s) \propto_T DDP(s), \tag{3}$$

where $\propto_T$ stands for a polynomial Turing reduction (Garey and Johnson, 1979). Since Partition is NP-hard, (Karp, 1972), we obtain the result.

LEMMA 3.1 *PARTITION* $\propto_T$ *qDDP(s).*

Proof. In order to prove the lemma, one has to construct an algorithm A for solving Partition that uses a polynomial number of times a procedure P which solves qDDP(s).

Let us consider any procedure P that solves the quasi Double Digest Problem in the search version:

$$P(I_{qDDP(s)}) = \text{solution} \in Z(I_{qDDP(s)}) \text{ whenever } Z(I_{qDDP(s)}) \neq \emptyset, \tag{4}$$

$$P(I_{qDDP(s)}) = \text{'no' whenever } Z(I_{qDDP(s)}) = \emptyset, \tag{5}$$

where $Z(I)$ denotes the set of all solutions for any given instance I of the problem.

Our goal is to construct an algorithm A for solving Partition that uses procedure P a polynomial number of times. With this end in view, let the algorithm A be as follows:

$A(I_{PART})$ = generate the set $G(I_{PART})$ of corresponding instances of qDDP(s) and call procedure P for each generated instance.
$A(I)$ = 'yes' $\Leftrightarrow \exists_{J \in G(I)}$ $(P(J) = \text{solution})$, where $G(I)$ denotes the set of corresponding instances of qDDP(s) generated by algorithm A for instance I.

Let us consider any instance $I_{PART} = (k, S, F)$ of the Partition problem. The generation of set $G(I_{PART})$ is as follows: $n_q = k - 2$, $m_q = 1$, $l_q = \Sigma_{s \in S} f(s)$, $C_q = F$, $B_q = \{\frac{1}{2} \cdot l_q, \frac{1}{2} \cdot l_q\}$ for all generated instances. The only difference relates to multiset A_q, as for the i-th generated instance $I_{qDDP(d)i} \in G(I_{PART})$ the following holds:

$$A_{qi} = F - \{f(s_u), f(s_w)\} \cup \{f(s_u) + f(s_w)\}, \tag{6}$$

where $i = 1, \ldots, \frac{1}{2} \cdot (k^2 - k)$, $1 \leq u \leq k$, $1 \leq w \leq k$ and $u \neq w$.

Obviously, for each generated instance, another pair of elements of F is being removed, thus resulting in $\frac{1}{2} \cdot (k^2 - k)$ different instances. Also note that $\mu_{qi} = \langle \frac{1}{2} \cdot l_q, \frac{1}{2} \cdot l_q \rangle = \mu_q$ for each element of $G(I_{PART})$.

Algorithm A solves the decision problem Π iff:
1. for each instance $I \in Y_\Pi$ the answer 'yes' is generated,
2. for each instance $I \in D_\Pi - Y_\Pi$ the answer 'no' is generated.

Thus, we have to prove that:
1. $\forall_{I \in D(PART)} : (I \in Y_{PART} \Rightarrow A(I) = \text{'yes'})$,
2. $\forall_{I \in D(PART)} : (I \notin Y_{PART} \Rightarrow A(I) = \text{'no'})$.

First, let us consider any instance $I = (k, S, F) \in Y_{PART}$. Note that $A(I) =$ 'yes' $\Leftrightarrow \exists_{J \in G(I)}(P(J) = \text{solution})$, where $G(I)$ denotes the set of $\frac{1}{2} \cdot (k^2 - k)$

instances generated by algorithm A for instance I. Because the answer for I is 'yes', there exists a subset S' such that $\Sigma_{s\in S'} f(s) = \Sigma_{s\in S-S'} f(s)$. Thus, for any instance $J_i \in G(I)$, $i = 1, \ldots, \frac{1}{2}\cdot(k^2-k)$, the elements of multiset C_{qi} can be divided into two sets with equal sums of sizes. Moreover, there exists at least one instance $J_i \in G(I)$, for which one can construct a permutation σ_{qi} of A_{qi}, $i = 1, \ldots, \frac{1}{2}\cdot(k^2-k)$ such that the composition of σ_{qi} and μ_{qi} results in the list with the desired property.

Second, let us consider any instance $I = (k, S, F) \notin Y_{PART}$. Note that $A(I) =$ 'no' $\Leftrightarrow \forall_{J\in G(I)}(P(J) =$ 'no'), where $G(I)$ denotes the set of $\frac{1}{2}\cdot(k^2-k)$ instances generated by algorithm A for instance I. Because the answer for I is 'no', there does not exist a subset S' such that $\Sigma_{s\in S'} f(s) = \Sigma_{s\in S-S'} f(s)$. Thus, for any instance $J_i \in G(I)$, $i = 1, \ldots, \frac{1}{2}\cdot(k^2-k)$, the elements of multiset C_{qi} cannot be divided into two sets C'_{qi} and $C_{qi} - C'_{qi}$ with equal sums of sizes, while $B_{qi} = B_q = \{\frac{1}{2}\cdot l_q, \frac{1}{2}\cdot l_q\}$ and $\mu = \langle \frac{1}{2}\cdot l_q, \frac{1}{2}\cdot l_q\rangle$. Due to that fact, there does not exist any permutation σ_{qi} of A_{qi} for any instance $J_i \in G(I)$, $i = 1, \ldots, \frac{1}{2}\cdot(k^2-k)$ such that the composition of σ_{qi} and μ_q would result in the list with the desired property. ■

Let us note that as a result of the LEMMA we have proven that qDDP(s) is NP-hard. Now, we will prove the second part of our reduction process.

LEMMA 3.2 *$qDDP(s) \propto_T DDP(s)$.*

Proof. Again, in order to prove the lemma, one has to construct an algorithm A for solving problem qDDP(s) that uses a polynomial number of times a procedure P which solves DDP(s). The most important part of the proof consists in proving the following dependence:

$$Y_{DDP(d)} = Y_{qDDP(d)}. \tag{7}$$

With this end in view, let us consider any instance $I_{DDP(d)} = (n, m, l, A, B, C)$ of DDP(d). Clearly, since $D_{DDP(d)} = Y_{DDP(d)}$ (let us recall that this fact follows from the trivial answer to any ideal biochemical experiment), there must exist permutations σ and μ for which the answer is 'yes'. The corresponding instance $I_{qDDP(d)} = (n_q, m_q, l_q, A_q, B_q, C_q)$ of qDDP(d) can be easily constructed as follows: $n_q = n$, $m_q = m$, $l_q = l$, $A_q = A$, $B_q = B$, $C_q = C$, questions of the two instances being the same. Obviously, an answer to the question of $I_{qDDP(d)}$ is 'yes' and holds for permutations $\sigma_q = \sigma$ and $\mu_q = \mu$.

Now, let us consider any instance $I_{qDDP(d)} = (n_q, m_q, l_q, A_q, B_q, C_q)$ of qDDP(d) for which the answer is 'yes' and holds for the permutations σ_q and μ_q. Thus, the non-decreasingly ordered list of elements obtained out of the composition of σ_q and μ_q is identical to the list $\langle c_{q1}, \ldots, c_{q(m_q+n_q+1)}\rangle$. Adding the fact that (2) holds, we can construct the corresponding instance $I_{DDP(d)} = (n, m, l, A, B, C)$ of DDP(d) as follows: $n = n_q$, $m = m_q$, $l = l_q$, $A = A_q$, $B = B_q$, $C = C_q$. Obviously, an answer to the question of $I_{DDP(d)}$ is 'yes' and holds for permutations $\sigma = \sigma_q$ and $\mu = \mu_q$.

Thus, we have shown that any instance of DDP(d) is also an instance of qDDP(d) and that any instance of qDDP(d) for which the answer is 'yes' is also an instance of DDP(d). Adding the fact that $D_{DDP(d)} = Y_{DDP(d)}$, we have proven the following dependence:

$$Y_{DDP(d)} = Y_{qDDP(d)}. \tag{8}$$

Now, let us consider any procedure P that solves the Double Digest Problem in the search version:

$$P(I_{DDP(s)}) = \text{solution} \in Z(I_{DDP(s)}), \tag{9}$$

where $Z(I) \neq \emptyset$ denotes the set of all solutions for given instance I of the problem.

Our goal is to construct an algorithm A for solving qDDP(s) that uses procedure P a polynomial number of times. With this end in view, let the algorithm A be as follows:

$A(I_{qDDP(s)}) = P(I_{DDP(s)})$, where P denotes a procedure that solves DDP(s) and $I_{DDP(s)}$ denotes the corresponding instance of DDP(s).

Thus, $A(I_{qDDP(s)}) =$ 'solution' whenever the following dependence holds:

$$I_{qDDP(d)} \in Y_{qDDP(d)}, \tag{10}$$

while $A(I_{qDDP(s)}) =$ 'no' otherwise. ■

Now, we may prove the main result.

THEOREM 3.1 *Problem DDP(s) is NP-hard.*

Proof. Follows immediately from the lemmae and the fact that Partition is NP-hard. ■

4. Summary

In the paper, the Double Digest Problem in the search version was proved to be NP-hard under the assumption that the restriction enzymes cut DNA molecules in different loci. It was also shown that the number of equivalent solutions grows exponentially with the length of a DNA strand being mapped (see Waterman, 1995; Goldstein and Waterman, 1987; Schmitt and Waterman, 1991). A good alternative is, thus, the Simplified Partial Digest Problem (SPDP in short), presented in Błażewicz et al. (2001).

References

ALIZADEH, F., KARP, R.M., WEISSER, D.K. and ZWEIG, G. (1995) Physical mapping of chromosomes using unique end-probes. *Journal of Computational Biology* **2**, 159-184.

Błażewicz, J., Formanowicz, P., Jaroszewski, M., Kasprzak, M. and Markiewicz, W.T. (2001) Construction of DNA restriction maps based on a simplified experiment. *Bioinformatics* **5**, 398-404.

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability. A Guide to the theory of NP-Completeness.* W.H. Freeman and Company, San Francisco.

Goldstein, L. and Waterman, M.S. (1987) Mapping DNA by stochastic relaxation. *Advanced Applied Mathematics* **8**, 194-207.

Karp, R.M. (1972) *Reducibility among combinatorial problems.* In: R.E. Miller and J.W. Thatcher, eds., *Complexity of Computer Computations.* Plenum Press, New York, 85-103.

Rosenblatt, J. and Seymour, P. (1982) The structure of homeometric sets. *SIAM J. Alg. Disc. Math.* **3**, 343-350.

Schmitt, W. and Waterman, M.S. (1991) Multiple solutions of DNA restriction mapping problem. *Advanced Applied Mathematics* **12**, 412-427.

Setubal, J. and Meidanis, J. (1997) *Introduction to Computational Biology.* PWS Publishing Company, Boston.

Skiena, S.S., Smith, W.D. and Lemke, P. (1990) Reconstructing sets from interpoint distances. In: *Proc. Sixth ACM Symp. Computational Geometry.* 332-339.

Skiena, S.S. and Sundaram, G. (1994) A partial digest approach to restriction site mapping. *Bulletin of Mathematical Biology* **56**, 275-294.

Waterman, M.S. (1995) *Introduction to Computational Biology. Maps, Sequences and Genomes.* Chapman & Hall, London.