

# On the Topic of Sequence Assembly Problems

*C. J. Lin and J. P. Lu*

## 1. General Sequences Assembly Problem (GSAP)

**Input:** A set of strings  $S = \{s_1, s_2, \dots, s_n\}$ .

**Output:** A target string  $w$ .

The input data is usually obtained by the method of shotgun. For example, given a string  $A$  and we utilize the shotgun method to break the string  $A$  many times:

The string  $A$ :      ATCTGGCTACCATGCTAGCTAGTTAGCTAGGGCTCTAT

First cut:            ATCTGGCTACCATGCTAGCTAGTTAGCTAGGGCTCTAT

Second cut:         ATCTGGCTACCATGCTAGCTAGTTAGCTAGGGCTCTAT

In this case, the string  $A$  is cut 2 times. At the first cut, we obtain three substrings of  $A$ :

ATCTGGCTAC,  
CATGCTAGCTAGTT, and  
AGCTAGGGCTCTAT

At the second cut, we obtain four substrings of  $A$ :

ATCTG,  
GCTACCATGCTA,  
GCTAGTTAGCTAGG, and  
GCTCTAT

The input of this problem is above 7 substrings. Our job is to reconstruct a target string  $w$  in which each string  $s_i$  in  $S$ ,  $i = 1, \dots, n$ , is a substring of  $w$ . **Note that here we ignore the problem of repeating groups and errors produced during the process of shotgun.**

Let  $\lambda$  denote the length of a substring and  $n$  denote the number of substrings. Because we ignore the problem of repeating groups, we'll obtain the time complexity of  $O(n^2 \cdot \lambda \log \lambda)$ .

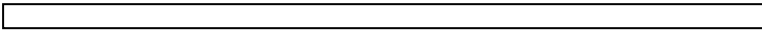
## 2. Double Digest Problem (DDP)

**Input:** Three sets of *lengths*, A, B and C.

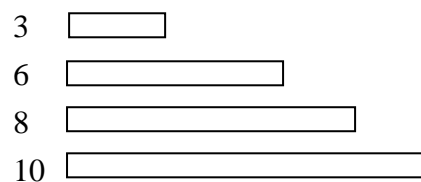
**Output:** *Orders* of above three sets.

We state this problem by the following example.

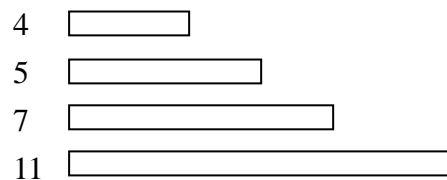
Given a DNA sequence  $R$ . At the first step, we use an enzyme  $A$  to break  $R$  into pieces of lengths  $\{3, 6, 8, 10\}$ . At the second step, we use an enzyme  $B$  to break  $R$  into pieces of lengths  $\{4, 5, 7, 11\}$ . At last, we apply both enzymes  $A$  and  $B$  simultaneously to cut  $R$ . The sequence  $R$  will be broken into pieces of lengths  $\{1, 2, 3, 3, 5, 6, 7\}$ . The process above is shown as follows:

R: 

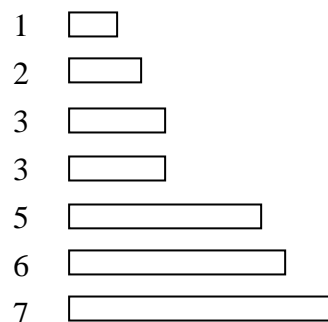
Applying enzyme  $A$ :



Applying enzyme  $B$ :



Applying both enzyme  $A$  and  $B$  simultaneously:



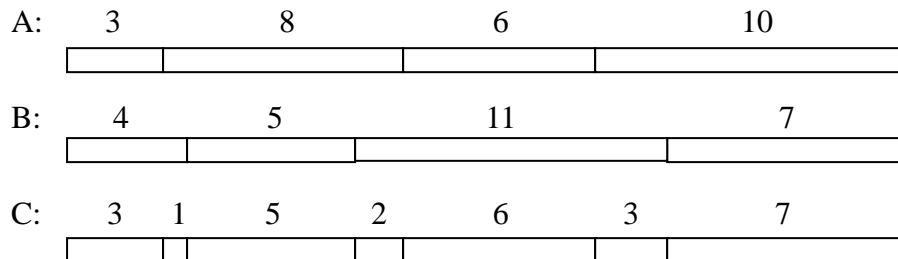
In this case,

The input set A will be  $\{3, 6, 8, 10\}$ .

The input set B will be  $\{4, 5, 7, 11\}$ .

The input set  $C$  will be  $\{1, 2, 3, 3, 5, 6, 7\}$ .  
 Our job is to order above three sets,  $A$ ,  $B$  and  $C$ .

A feasible solution is as follows:



There may have many solutions. In DDP, we discuss the order of lengths sets. DDP is an  $NP$ -hard problem which was proved in 1987 [GW87]. The reason why it is very hard is that we have to figure out a feasible solution, which makes the orders of these three input sequences satisfy the relation between  $A$  and  $C$  and the relation between  $B$  and  $C$ . The result of  $NP$ -hard can be seen from the following reference:

[GW87] **Mapping DNA by stochastic relaxation**, Goldstein, L., and Waterman, M. S., **Adv. In Appl. Math.**, Vol. 8, 1987, pp. 194-207.

[S78] **Inferfing DNA structure from segmentation data**, Stefik, M., **Artificial Intelligence**, Vol. 11, 1978, pp. 85-114.

[GM90] **Mapping DNA by stochastic relaxation : a new approach to fragment sizes**, Grigorjev, A. V., and Mironov, A. A., **CABIOS**, Vol. 6, 1990, pp. 107-111.

[GM91] **Mapping DNA by stochastic relaxation : a schedule for optimal annealing**, Grigorjev, A. V., and Mironov, A. A., **J. DNA Mapping and Sequencing**, Vol. 1, 1991, pp. 221-226.

[FSR83] **Mapping the order of DNA restriction fragments**, Fitch, W. M., Smith, T. F., and Ralph, W. W., **Gene**, Vol. 22, 1983, pp. 19-29.

[NMS84] **Plasmid mapping computer program**, Nolan, G. P., Maina, C. V., and Szalay, A. A., **Nucleic Acids Res.**, Vol. 12, 1984, pp. 717-729.

[B88] **Construction of restriction maps**, Bellon, B., **CABIOS**, Vol. 4, 1988, pp. 111-115.

[P92] **DNA Physical Mapping, Flows in Networks and Minimum Cycles Mean in Graphs**, Pevzner, P. A., **DIMACS Series in Discrete Mathematics and Theoretical Computer Science**, Vol. 8, 1992, pp. 99-112.

[PM87] **An efficient method for physical mapping of DNA molecules**, Pevzner, P. A., and Mironov, A. A., **Molek. Biol.**, Vol. 21, 1987, pp. 788-796.

[HAY90] **Restriction site mapping for three or more enzymes**, Ho, S. T. S., Allison, L., and Yee, C. N., **CABIOS**, Vol. 6, 1990, pp. 195-204.

[DK88] **Errors between sites in restriction site mapping**, Dix, T. I., and Kieronska, D. H., **CABIOS**, Vol. 4, 1988, pp. 117-122.

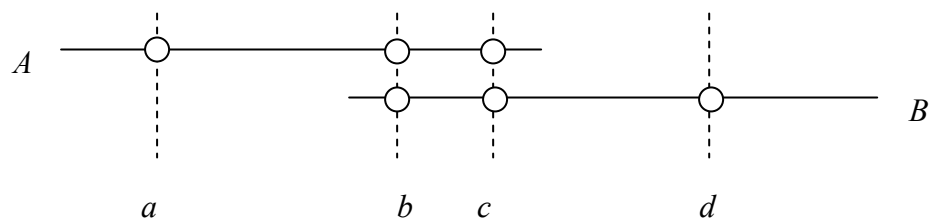
[DB84] **An efficient program to construct restriction maps from experimental data with realistic error levels**, Durand, R., and Bregegere, F., **Nucleic Acids Res.**, Vol. 12, 1984, pp. 703-716.

### **3. Consecutive Ones Problem (C1P)**

**Input:** A 1/0 matrix( $n \times m$ ).

**Output:** A target matrix.

Suppose there are two DNA sequences,  $A$  and  $B$ . If we know that probes  $a$ ,  $b$  and  $c$  bound to sequence  $A$ , probes  $b$ ,  $c$  and  $d$  bound to sequence  $B$ , we believe that sequence  $A$  and  $B$  are related.



In this case, we can utilize a 1/0 matrix to represent above graph as follows:

	$a$	$b$	$c$	$d$
$A$	1	1	1	0
$B$	0	1	1	1

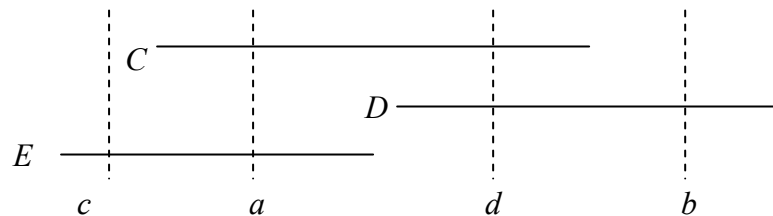
In the above 1/0 matrix, all 1's in each row are consecutive. The matrix is said to have the **consecutive ones property**. Consider the following case:

	$a$	$b$	$c$	$d$
$C$	1	0	0	1
$D$	0	1	0	1
$E$	1	0	1	0

The columns of above matrix can be permuted to reach the consecutive ones property. Our job is to find the matrix which has the consecutive ones property or not. The following matrix is satisfied the consecutive ones property by permuted some columns of above matrix.

	$c$	$a$	$d$	$b$
$C$	0	1	1	0
$D$	0	0	1	1
$E$	1	1	0	0

We can find the relation between these three sequences as follows:



**There exist many solutions of each instance of C1P.**

The C1P is not an *NP*-Complete problem. There are many algorithms running

within polynomial time, even linear time. The results come from the following references:

[FG65] **Incidence Matrices and Interval Graphs**, Fulkerson, D. R. and Gross, O. A., **Pacific J. Math.**, Vol. 15, 1965, pp. 835-855. <Polynomial time>

[BL76] **Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithm**, Booth, K. S. and Lueker, G. S., **Journal of Comput. Systmes Sci.**, Vol. 13, 1976, pp. 335-379. <Linear time>

[H2002] **A Simple Test for the Consecutive Ones Property**, Hsu, W. L. , **Journal of Algorithms**, Vol. 43, 2002, pp. 1-16.

<  $O(n + m + r)$ ;  $r$  : total number of ones; Matrix  $M = n \times m$  >

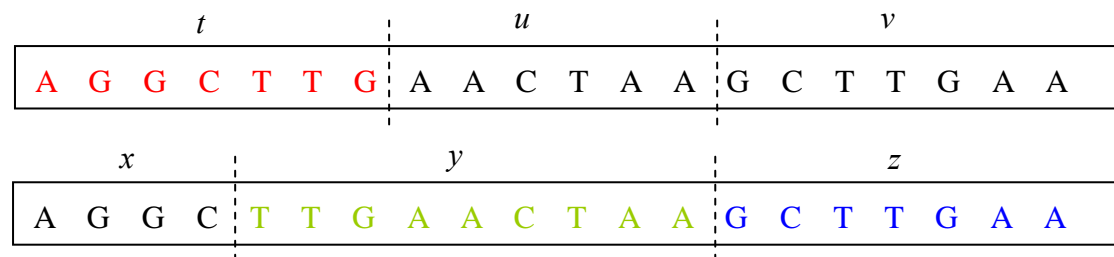
## **4. Shortest Common Superstring Problem (SCSP)**

**Input:** A substring-free set of strings  $S = \{s_1, s_2, \dots, s_n\}$ .

**Output:** A target string  $w$  with shortest length.

**Input can be given by cutting a DNA Sequence.** Given a set of strings  $S = \{s_1, s_2, \dots, s_n\}$ . A common superstring of  $S$  is a string containing each  $s_i \in S$  as a contiguous substring. The shortest common superstring  $w$  is a common superstring of shortest length.

For example, given strings  $t, u, v, x, y$  and  $z$ , where  $x, y$  and  $z$  are obtained by cutting a DNA sequence as follows:



Original DNA sequence  $\Psi = \text{AGGCTTGA} \text{ACTAAGCTTGAA}$

$t = \text{AGGCTTG}$

$u = \text{AACTAA}$

$v = \text{GCTTGAA}$

$x = \text{AGGC}$

$y = \text{TTGAACTAA}$

$z = \text{GCTTGAA}$

Since  $x$  is a substring of  $t$ ,  $u$  is a substring of  $y$ , and  $v$  is a substring of  $z$ . we ignore  $x$  and  $u$ . Now we obtain a substring-free set  $S = \{t, y, z\}$ .

$t = \text{AGGCTTG}$

$y = \text{TTGAACTAA}$

$z = \text{GCTTGAA}$

Finally, we can find that:

$w = \text{AGGCTTGAACCTAA}$  is a shortest common superstring of  $S$ .

Yet, this is not the original DNA sequence is

$\Psi = \text{AGGCTTGAACCTAAGCTTGAA}$ .

We may say that the shortest common superstring  $w$  contains every given string  $s_i$  which is cut from the original DNA sequence  $\Psi$ . However, the length of the shortest common superstring may be less than the original DNA sequence. Actually,  $w$  is a subsequence of  $\Psi$ . In addition, we intuitively have the feeling that SCSP is quite related to the topic of data compression.

Why does this situation occur? This is due to the problem of repeating groups (i.e., not substring-free). **In addition, SCSP doesn't consider to recover the original sequence. It considers to reduce length of the assembled string. Therefore, if we ignore the problem of repeating groups and use subsequences cut from the original DNA sequence, GSAP is equivalent to SCSP. At this case, we clearly know that SCSP must have a polynomial time algorithm to solve it as GSAP does.**

The shortest common superstring problem, i.e., SCSP is an *NP*-Complete problem which has been proved on this paper [GMS80].

[GMS80] **On Finding Minimal Length Superstring** , Gallant, J., Maier, D. and Storer, J. , **Journal of Computer and System Sciences** , Vol. 20 , 1980 , pp. 50-58.