BIOINFORMATION
Discovery at the interface of physical and biological sciences

open access www.bioinformation.net Hypothesis
Volume 8(10)

Genetic algorithm solution for double digest problem

Mohammad Ganjtabesh[1, 2*], H Ahrabian[1], A Nowzari-Dalini[2] & Z Razaghi Kashani Moghadam[3]

[1]Center of Excellent in Biomathematics, School of Mathematics, Statistics, and Computer Science, University of Tehran, Tehran, Iran; [2]Laboratoire d'Informatiques (LIX), Ecole Polytechnique, Palaiseau Cedex 91128, France; [3]Institute of Biochemistry and Biophysics (IBB), University of Tehran, Tehran, Iran; Mohammad Ganjtabesh – E mail: mgtabesh@ut.ac.ir; Phone: +98-21-66412178; *Corresponding author

Received December 26, 2011; Accepted December 28, 2011; Published May 31, 2012

Abstract:
The strongly NP-Hard Double Digest Problem, for reconstructing the physical map of DNA sequence, in now using for efficient genotyping. Most of the existing methods are inefficient in tackling large instances due to the large search space for the problem which grows as a factorial function $(a!)(b!)$ of the numbers a and b of the DNA fragments generated by the two restriction enzymes. Also, none of the existing methods are able to handle the erroneous data. In this paper, we develop a novel method based on genetic algorithm for solving this problem and it is adapted to handle the erroneous data. Our genetic algorithm is implemented and compared with the other well-known existing algorithms. The obtained results show the efficiency (speedup) of our algorithm with respect to the other methods, specially for erroneous data.

Background:
Usually, a DNA sequence is too long to be studied entirely and it must be broken into a set of small fragments by performing the digest experiment. In such experiment, restriction enzymes are used to cut the DNA sequence at specific positions. The length of these small fragments can be easily measured by using the gel-electrophoresis and their corresponding sequences become easier to determine in biological laboratory **[1]**. Given the length of these fragments, the DNA restriction mapping deals with the problem of determining the original ordering of these fragments (digest problem). The fragment lengths resulting from a single digest experiment can not yield any information about the ordering of the fragments. For this reason, double digestion experiment is performed where two different enzymes are used as follows. First a set of clones of DNA sequences are digested by an enzyme, say A. Then a second set of the same clones are digested by another enzyme, say B. Finally, the third set of the same clones are digested by a mix of both enzymes A and B, which we refer as C **(Figure 1)**. The goal is to reconstruct the original ordering of the fragments in the DNA sequences. The NP-Hardness of DDP has been proved by Goldstein and Waterman **[2]**. Cieliebak has shown that DDP is Strongly NP-Hard, even if the two enzymes always cut at disjoint restriction sites. Moreover, he has shown that for partial cleavage errors the problem of finding a solution with minimum number of errors is hard to approximate **[3]**. Due to the basic difficulty of DDP, Sur-Kolay et. al. **[4]** present a genetic algorithm to solve the errorless version of this problem. Prior to this algorithm, several approaches including the restriction site mapping based on separation theory **[5]** and computer-assistant interactive strategies **[6]** have been proposed in order to tackle this problem. In 2008, Wu and Zhang **[1]** formulated this problem as a mixed-integer linear program and by using the integer programming techniques, they can solve randomly generated large instances of DDP. Also a molecular solution for this problem is presented in **[7]**. None of the mentioned approaches could handle the erroneous data. With respect to the above discussion and the difficulty of DDP, it is desirable to solve this problem by using heuristic techniques such as Genetic Algorithms. In this paper, we present a novel

genetic algorithm for solving *DDP*, and we extend it to handle the erroneous data. Our algorithm is implemented and compared with the other well-known existing algorithms on two different data types, random and real data. The obtained results show the efficiency (speedup) of our genetic algorithms with respect to the other methods. For large input data size or erroneous data, the efficiency of our algorithm becomes more and more obvious.

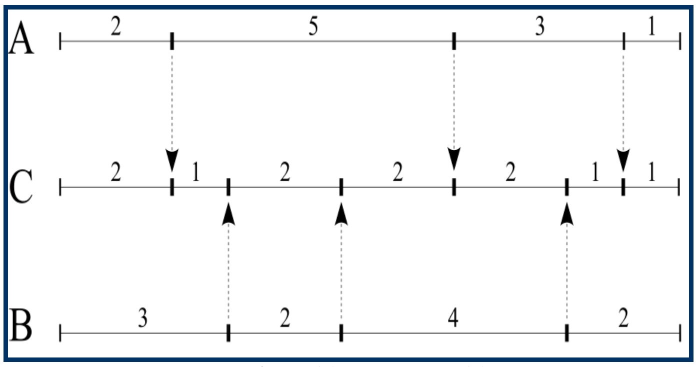

Figure 1: An instance of Double Digest Problem.

Methodology:
DDP can be formally defined as follows **[3]**, where $sum\ (S)$ denotes the sum of the elements in the multiset S and $dist\ (P)$ is the multiset of all distances between the adjacent points in the set P of points on a line.

Definition: Given three multisets A, B, and C of positive integers with $sum\ (A) = sum\ (B) = sum\ (C)$, find three ordered sets P^A, P^B, and P^C of points on a line, such that $dist\ (P^A) = A$, $dist\ (P^B) = B$, $dist\ (P^C) = C$, and $P^A \cup P^B = P^C$ (the point 0 is the smallest point in each set). In reality, the data obtained by digest experiment may contains different types of errors **[3]**. Here, we consider partial cleavage errors, where an enzyme fails to cut at some restriction sites. In this case, the optimization criterion for DDP is to minimize $|P^C - (P^A \cup P^B)| + |(P^A \cup P^B) - P^C|$ which is the absolute number of partial cleavage errors in a solution. The corresponding optimization problem, Min Absolute Error DDP, is formally defined as follows.

Definition: Given three multisets A, B, and C of positive integers such that $sum\,(A) = sum\,(B) = sum\,(C)$, find three ordered sets P^A, P^B, and P^C of points on a line, such that $dist\ (P^A) = A$, $dist\ (P^B) = B$, $dist\ (P^C) = C$, and $|(P^A \cup P^B) - P^C| + |P^C - (P^A \cup P^B)|$ is minimum (the point 0 is the smallest point in each set).

Genetic Algorithm
Genetic algorithm (GA) is a global optimization method inspired from biological evolution. The GA searches for an optimal solution from a population of candidate solutions according to an objective function. Genetic operations (selection, crossover, and mutation) are used on a given generation to produce the solutions for next generation. These operations are designed to preserve the most successful aspects of solutions until the best possible one is attained. The general scheme of our genetic algorithm is presented in **(Figure 2).**

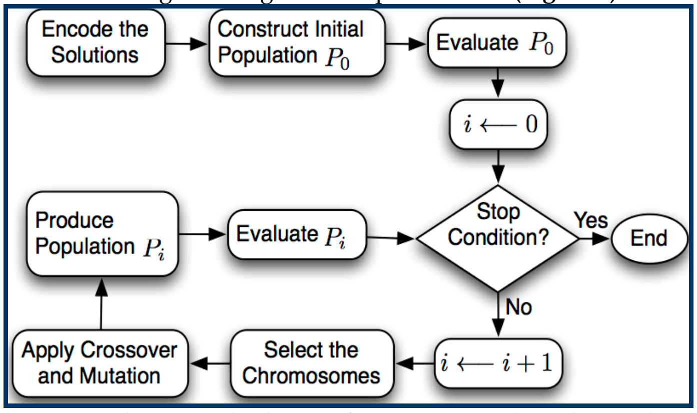

Figure 2: The general scheme of our Genetic Algorithm and its components**.**

Problem Encoding
To design a genetic algorithm, the solutions must be encoded in such a way that the genetic operations could be applied on them. To encode the solutions of *DDP*, we use the permutation of the multisets A and B as chromosomes. The initial population is randomly generated to represent the initial solutions. Each solution contains two chromosomes, one from the population X (corresponding to the multiset A) and the other one from the population Y (corresponding to the multiset B). For any chromosome $X_i = (x_{i_1}, x_{i_2}, \cdots, x_{i_n})$ and $Y_j = (y_{j_1}, y_{j_2}, \cdots, y_{j_m})$ in the population X and Y, respectively, the multiset Z is simply constructed by mapping the elements of X_i and Y_j on a line and computing the distances between adjacent points. The fitness function for these chromosomes is considered as $f(Z) = |Z \cap C|$. Our genetic algorithm tries to maximize the fitness function and it is terminated when $f(Z) = |C|$ or the maximum number of iterations is reached.

Extension for Erroneous Data
To consider the partial cleavage errors, we adopt the fitness function of our genetic algorithm in such a way that the solution with less error value has more opportunity to be selected for producing the next population. This error can be estimated by $|P^C - (P^A \cup P^B)| + |(P^A \cup P^B) - P^C|$. In this case, our genetic algorithm is terminated when the estimated error becomes 0 or the maximum number of iterations is reached.

Genetic Operations
In developing of our algorithm, the Roulette-Wheel method is chosen as a selection operation, where a chromosome with better fitness value is selected with higher probability **(for equation see supplementry material).**

Discussion:
Our algorithm is implemented in C#.Net Framework 2.0 (the source code is available by email request to the corresponding author). The size of the population is considered as 100, the

maximum iterations is set to $|A||B|$, $p_c = 0.8$, $p_m = 0.01$, and the higher (lower) fitness value represents the encoding of the best solution for errorless (erroneous) instances. In what follows, the cpu time (in second) is considered as a measure of comparison. In order to compare the efficiency of our algorithm with Sur-Kolay's genetic algorithm **[4]**, different data sets have been generated randomly. The size of the generated instances are shown on the $X-$axis of (**Figure 3**). For each size, 10 different instances are generated and both algorithms have been executed on them. The presented times (in second) are the average running time over 10 different executions. As it is clear in this figure, the efficiency of our genetic algorithm is better than the Sur-Kolay's genetic algorithm for all instances. For erroneous data, since none of the mentioned approaches could handle them, we compare our algorithm with backtracking algorithm.

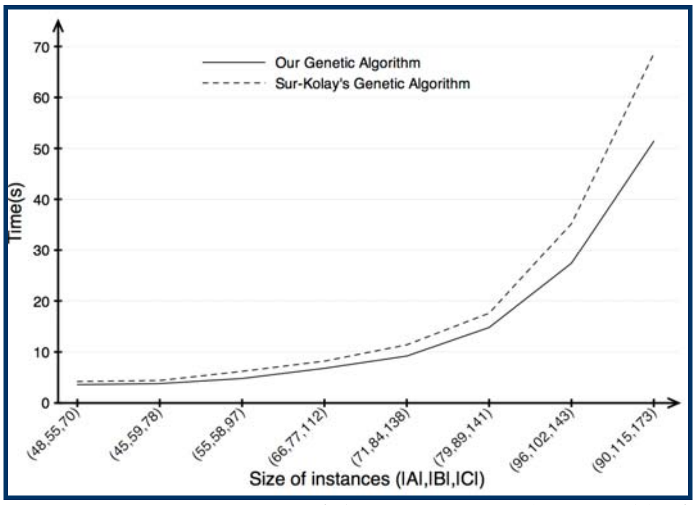

Figure 3: Time comparison of the running time (in seconds) of our genetic algorithm and Sur-Kolay's genetic algorithm for Double Digest Problem without error.

The results of this comparison are presented in **Table 1 (see supplementry material)**. Again, for each size, 10 different instances are generated randomly with different error bounds. The times presented in this table are the average running times of 10 different executions of these algorithms. The efficiency (speedup) of our algorithm is obvious from this table. To analyze the behavior of our genetic algorithm for different error bounds, our algorithm is executed on erroneous DDP instances with different error bounds and the results are presented in **(Figure 4)**. As we can see, the running time of our genetic algorithm is almost independent from the different error bounds. Also to show the accuracy of our genetic algorithm, we took real data sets for *DDP* form **[8].** The data is related to digesting Lambda DNA sequence by using the different pairs of enzymes (*HindIII*, *EcoR*) and (*BamHI*, *HindIII*), which have (8, 6) and (3, 8) different fragments, respectively. By using the both enzymes at the same time for each pair, we gain 13 and 10 different fragments, respectively. By performing our genetic algorithm, the correct order of the fragments is obtained.

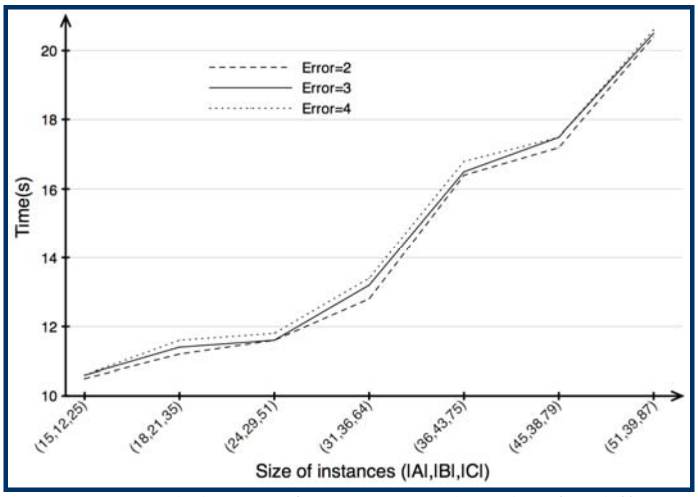

Figure 4: Time behavior of our genetic algorithm for different error bounds.

Conclusion:

A novel genetic algorithm is designed for solving double digest problem. This algorithm help us to obtain the accurate result in a reasonable amount of time for a computationally hard problem. Specially, for erroneous data, our genetic algorithm leads us to a satisfactory solution. The experimental results of our algorithm confirm its efficiency.

References:

[1] Wu Z & Zhang Y, *Int J Bioinform Res Appl.* 2008 **4**: 351 [PMID: 19008180]
[2] Goldstein L & Waterman MS, *Adv Appl Math.* 1987 **8**: 194
[3] Cieliebak M *et al. J Bioinform Comput Biol.* 2005 **3**: 207 [PMID: 15852501]
[4] Sur-Kolay S *et al. Int J Bioinform Res Appl.* 2009 **5**: 570 [PMID: 19778871]
[5] Allison L & Yee CN, *Comput Appl Biosci.* 1988 **4**: 97 [PMID: 2838139]
[6] Wright LW *et al. Comput Appl Biosci.* 1994 **10**: 435 [PMID: 7804876]
[7] Ganjtabesh M *et al. Comput. Inform.* 2009 **28**: 599
[8] *http://biology.clc.uc.edu/fankhauser/labs/genetics/DNA_Digestion/*DNA_Digestion.htm.

Edited by P Kangueane
Citation: **Ganjtabesh *et al.*** Bioinformation 8(10): 453-456 (2012)

Supplementary material:

Genetic Operations:
In developing of our algorithm, the Roulette-Wheel method is chosen as a selection operation, where a chromosome with better fitness value is selected with higher probability.

To perform the crossover operation on $X_i = (x_{i_1}, x_{i_2}, \cdots, x_{i_n})$ and $X_j = (x_{j_1}, x_{j_2}, \cdots, x_{j_n})$ from X, the following procedure is used and two new chromosomes O_i and O_j are produced. Also the parameter p_c $(0 \le p_c \le 1)$ is used to control the application probability of crossover operation. The same procedure is applied on the chromosomes of the population Y.

```
Procedure Crossover( $X_i$, $X_j$, $O_i$, $O_j$ )
for $k \leftarrow 1$ to $n$ do
if $x_{i_k} = x_{j_k}$ then $o_{i_k} \leftarrow x_{i_k}$; $o_{j_k} \leftarrow x_{i_k}$;
else if $rnd \le 0.5$ and $x_{j_k} \notin O_i$ and $x_{i_k} \notin O_j$ then $o_{i_k} \leftarrow x_{j_k}$; $o_{j_k} \leftarrow x_{i_k}$;
else if $x_{i_k} \notin O_i$ and $x_{j_k} \notin O_j$ then $o_{i_k} \leftarrow x_{i_k}$; $o_{j_k} \leftarrow x_{j_k}$;
end if.
end for.
if some positions in $O_i$ ($O_j$) are left blank then
randomly fill them with the elements in $X_i \setminus O_i$ ($X_j \setminus O_j$);
end if.
end.
```

To escape converging to a local optima and also to regenerate the lost chromosomes, the mutation operation should be considered. This operation is performed with probability p_m on a single chromosome in two different ways as follows:
The sub-array which is chosen randomly is reversed in place.
The chosen sub-array is circularly shifted to left or right by one position.

Table 1: The comparison of the running time (in seconds) of backtracking algorithm and our genetic algorithm for erroneous randomly generated instances of different sizes.

Size of instance $(\lvert A\rvert, \lvert B\rvert, \lvert C\rvert)$	Average running time for backtracking algorithm	Average running time for our genetic algorithm
(10,11,18)	775.9	10.8
(10,12,19)	1124.2	10.5
(15,18,28)	6495	11.2
(20,21,35)	18770.3	11.0
(23,25,46)	24128	11.2
(24,29,51)	∞	11.6
(31,36,64)	∞	12.8
(36,43,75)	∞	16.4
(45,38,79)	∞	17.2
(51,39,87)	∞	20.4