JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 1, Number 3, 1994
Mary Ann Liebert, Inc., Publishers
Pp. 235–239

An Exponential Example for a Partial Digest Mapping Algorithm

ZHENG ZHANG

ABSTRACT

The partial digest problem for small-scale DNA physical mapping is known in computer science as the turnpike reconstruction problem. Although no polynomial algorithm for this problem is known, a simple backtracking algorithm of Skiena *et al.* works well in practice. Weiss raises the question whether an exponential example exists for this algorithm. This paper presents such an exponential example for this backtracking algorithm.

Key words: DNA; physical mapping; partial digest algorithm

INTRODUCTION

THE PARTIAL DIGEST MAPPING PROBLEM (known in computer science as the turnpike problem) is to reconstruct the positions of n restriction sites from the list of all the distances between every pair of these n restriction sites. Rosenblatt and Seymour (1982) give a pseudopolynomial algorithm for this problem using polynomial factoring. The running time analysis for Rosenblatt and Seymour's algorithm is due to Lemke and Werman (1988). In fact, Rosenblatt and Seymour's method can solve a more general problem when n points of the set to be determined are in n-dimensional space (this problem is motivated by phase retrieval of spectroscopic analysis). In practice, one often uses a simple backtracking algorithm due to Skiena *et al.* (1990). Experiments show the algorithm works well, and Skiena and Sundaram (1994) generalized this algorithm to deal with noisy data. No polynomial bound for the Skiena *et al.* backtracking algorithm is known. However, no exponential counter-example has been found (Weiss, 1992). Moreover, Weiss raises the question if such an exponential example exists: "... if there is no backtracking, the running time is $O(n^2 \log n)$. Of course, backtracking happens, and if it happens repeatedly, then the performance of the algorithm is affected. No polynomial bound on the amount of backtracking is known, but on the other hand, there are no pathological examples that show that backtracking must occur more than $O(1)$ times. Thus, it is entirely possible that this algorithm is $O(n^2 \log n)$." (quoted from Weiss 1992, p. 400). In this paper, I will construct an exponential example for the backtracking algorithm of Skiena *et al.*

PARTIAL DIGEST PROBLEM

We first review the partial digest problem and the Skiena *et al.* backtracking algorithm.

Definition 1: *Let X be a finite set of points of $\mathbb{R}$. $\Delta(X)$ is defined to be the list of all distances between all pairs of points of X.*

Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802.

Definition 2: *Let X and Y be two finite sets of points of* $\mathbb{R}$. $\Delta(X,Y)$ *is defined to be the list of all distances between a point of X and a point of Y.*

The turnpike problem is: Given a list L, find a point set X such that $\Delta(X) = L$. The Skiena *et al.* backtracking algorithm is outlined as follows (see Weiss, 1992 and Skiena *et al.*, 1990 for details): First find the longest distance in L, which decides the two outermost points of the set, and then delete this distance from L. Then repeatedly position the longest remaining distance of L. Because for each step the longest distance in L must be realized from one of the outermost points, there are only two possible positions (left or right) to put the point. At each step, for each of the two positions, we need to check if all the distances from the position to the points already selected are in L. If it is true, delete all those distances before going to the next step. Backtrack if it is false for both of the two positions. A solution has been found when L is empty.

For example, suppose $L = \{2,2,3,3,4,5,6,7,8,10\}$. Because L includes all the pairwise distances, then $|L| = n(n-1)/2$, where n is the number of points in the solution. Now $|L| = 10$ so $n = 5$. First set $x_1 = 0$. Since 10 is the largest distance in L, it is clear that $x_5 = 10$. Removing distance $x_5 - x_1 = 10$, we obtain

$$X = \{0,10\} \qquad L = \{2,2,3,3,4,5,6,7,8\}.$$

The largest remaining distance is 8. Now we have two choices, either $x_4 = 8$ or $x_2 = 2$. Because those two cases are mirror images of each other, without loss of generality, we can assume $x_2 = 2$. After removal of distances $x_5 - x_2 = 8$ and $x_2 - x_1 = 2$ from L, we obtain

$$X = \{0,2,10\} \qquad L = \{2,3,3,4,5,6,7\}.$$

Since 7 is the largest remaining distance, we have either $x_4 = 7$ or $x_3 = 3$. If $x_3 = 3$, distance $x_3 - x_2 = 1$ should be in L, but it is not, so we can only set $x_4 = 7$. After removing distances $x_5 - x_4 = 3$, $x_4 - x_2 = 5$, and $x_4 - x_1 = 7$ from L, we obtain

$$X = \{0,2,7,10\} \qquad L = \{2,3,4,6\}.$$

Now 6 is the largest remaining distance. Once again we have two choices, either $x_3 = 4$ or $x_3 = 6$. If $x_3 = 6$, the distance $x_4 - x_3 = 1$ must be in L, but it is not. So it leaves us as the only choice $x_3 = 4$, and provides a solution $\{0,2,4,7,10\}$ of the partial digest problem. Figure 1 gives the pseudocode for the algorithm. Here the function Delete_Max(L) returns the maximum value of L and removes it from list L, and two global variables X and *width* are used. Note that at each step the choice of putting a point at the left or right position is arbitrary.

This algorithm runs in $O(n^2 \log n)$ expected time if L arises from real points in general positions, because in this case at each step one of the two choices will be pruned with probability of 1 (Skiena *et al.*, 1990). Is this algorithm polynomial in the worst case? In the next section, we show the answer is no, by describing an exponential example for this algorithm.

EXPONENTIAL EXAMPLE

Denote by $\oplus$ the merge (union) operation of two lists of distances. And for a set D, denote $D^* = \{1 - a | a \in D\}$.

Proposition 1: *Assume* $0 < \varepsilon < \frac{1}{4}n$. *Let* $D = \{x_1, x_2, \ldots, x_n\}$, where $x_i = (n+i)\varepsilon$ or $x_i = 1 - (n+i)\varepsilon$. Let $E = \{x'_1, x'_2, \ldots, x'_n\}$, where $x'_i = (n+i)\varepsilon$ or $x'_i = 1 - (n+i)\varepsilon$. If $D \neq E$ and $D \neq E^*$, then $\Delta(D) \neq \Delta(E)$.

Proof Since $\Delta(P) = \Delta(P^*)$ for any point set P, without loss of generality, we can assume $x_1 = x'_1 = (n+1)\varepsilon$. Now suppose $x_i = x'_i$ for all $i < j$ and $x_j \neq x'_j$. By the definition of x_j and x'_j, we can assume $x_j = 1 - (n+j)\varepsilon$ and $x'_j = (n+j)\varepsilon$. Let $X = \{x_1, x_2, \ldots, x_{j-1}\}$, $Y = D - X = \{x_j, x_{j+1}, \ldots, x_n\}$ and $Z = E - X = \{x'_j, x'_{j+1}, \ldots, x'_n\}$. Then

$$\Delta(D) = \Delta(X) \oplus \Delta(Y) \oplus \Delta(X,Y) \qquad \text{and} \qquad \Delta(E) = \Delta(X) \oplus \Delta(Z) \oplus \Delta(X,Z).$$

It is obvious that

$$\max(\Delta(Y) \oplus \Delta(X,Y)) \geq |x_j - x_1| = 1 - (2n + j + 1)\varepsilon.$$

Since $\max(Z) \leq 1 - (n + j + 1)\varepsilon$ and $\min(Z) = (n + j)\varepsilon$, we have

$$\max(\Delta(Z)) \leq 1 - (2n + 2j + 1)\varepsilon,$$

```
set $X$;
int $width$;

procedure Partial_Digest(List $L$)
    $width = $ Delete_Max($L$);
    $X = \{0, width\}$;
    place($L$);
end;

procedure place(List $L$)
    if $L = \emptyset$ then
        output solution $X$;
        exit;
    endif
    $y = $ Delete_Max($L$);
    if $\Delta(\{y\}, X) \subset L$ then
        $X = X \cup \{y\}$;
        place($L - \Delta(\{y\}, X)$);          /* place a point at right position */
        $X = X - \{y\}$;                      /* Backtracking */
    endif
    if $\Delta(\{width - y\}, X) \subset L$ then
        $X = X \cup \{width - y\}$;
        place($L - \Delta(\{width - y\}, X)$);      /* place a point at left position */
        $X = X - \{width - y\}$;                /* Backtracking */
    endif
end
```

FIG. 1. Pseudocode for the Skiena *et al.* backtracking algorithm.

and similarly, noticing that $\max(X) \leq 1 - (n + 2)\varepsilon$ and $\min(X) = (n + 1)\varepsilon$, we have

$$\max(\Delta(X,Z)) = \max\{\max(X) - \min(Z), \max(Z) - \min(X)\} \leq 1 - (2n + j + 2)\varepsilon.$$

So, $\Delta(Y) \oplus \Delta(X,Y) \neq \Delta(Z) \oplus \Delta(X,Y)$ and thus $\Delta(D) \neq \Delta(E)$. □

Proposition 2: *Assume* $0 < \varepsilon < \frac{1}{12} n$. *Let* $A_1 = \{1 - \varepsilon, 1 - 2\varepsilon, \ldots, 1 - n\varepsilon\}, A_2 = \{\varepsilon, 2\varepsilon, \ldots, n\varepsilon\}$ $A_3 = \{(n + 1)\varepsilon, (n + 2)\varepsilon, \ldots, 2n\varepsilon\}, A_4 = \{(2n + 1)\varepsilon, (2n + 2)\varepsilon, \ldots, 3n\varepsilon\}, A_5 = \{1 - (2n + 1)\varepsilon, 1 - (2n + 2)\varepsilon, \ldots, 1 - 3n\varepsilon\}$, *and* $D = F \cup G$ *where F and G are disjoint point sets satisfying* $F \cup G^* = A_3$ *(so D is just the same as that of Proposition 1). Let* $A = A_1 \cup A_2 \cup A_4 \cup A_5 \cup \{0,1\} \cup D$. *We can choose D such that, giving* $\Delta(A)$ *to the Skiena et al. backtracking algorithm, it will take it at least* $\Omega(2^{n-1})$ *time to find A. (Fig. 2 shows the arrangement of A.)*

Proof It is clear that there are 2^n different choices of D. For the first $2n + 1$ steps of the algorithm, we will find all the points of set A_1 and A_2 correctly. For the ith step of the next n steps, it is easy to see that the largest distance we are positioning is $1 - (n + i)\varepsilon$. This shows that for each of the next n steps, we can place a point at two possible places. It is obvious that there are 2^n different ways we can put these n

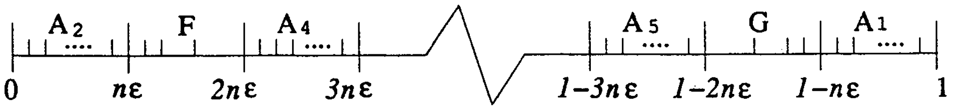

FIG. 2. Arrangement of set A.

points. Now we prove that none of the 2^n cases conflicts with $\Delta(A)$.

Let E be the set of the n points we pick in the next n steps and let $B = E \cap [(n+1)\varepsilon, 2n\varepsilon]$ and $C = E \cap [1 - 2n\varepsilon, 1-(n+1)\varepsilon]$. (Note E is also in the form of E of Proposition 1.) For any arbitrary choice of D and E, it is clear that

$$\Delta(A_1 \cup A_2 \cup \{0,1\}, D) = \Delta(A_1 \cup A_2 \cup \{0,1\}, A_3) = \Delta(A_1 \cup A_2 \cup \{0,1\}, E), \tag{1}$$

$$\Delta(E \cup A_1 \cup A_2 \cup \{0,1\}) = \Delta(E) \oplus \Delta(A_1 \cup A_2 \cup \{0,1\}) \oplus \Delta(A_1 \cup A_2 \{0,1\}, E), \tag{2}$$

and

$$\begin{aligned}\Delta(A) &= \Delta(A_1 \cup A_2 \cup D \cup A_4 \cup A_5 \cup \{0,1\}) \\ &\supset \Delta(A_1 \cup A_2 \cup \{0,1\}) \oplus \Delta(A_4) \oplus (A_1 \cup A_2 \cup \{0,1\}, D) \oplus \Delta(A_1, A_4).\end{aligned} \tag{3}$$

It is also straightforward to check that $\Delta(B) \oplus \Delta(C) = \Delta(B) \oplus \Delta(C^*) \subset \Delta(B \cup C^*) = \Delta(A_3) = \Delta(A_4)$, and $\Delta(B,C) \subset \Delta(A_1,A_4)$. Therefore, $\Delta(E) \subset \Delta(A_4) \cup \Delta(A_1,A_4)$. This along with (1)–(3) proves that $\Delta(E \cup A_1 \cup A_2 \cup \{0,1\}) \subset \Delta(A)$. So we cannot exclude any of the 2^n cases. If symmetric solutions do not count, we still have 2^{n-1} cases. By the proof stated in the following paragraph, we know that for each example there is only one solution. Any wrong choice of B and C in the algorithm will result in backtracking. So no matter what order we try to put a point at each step, we can always find an example from the above group that makes the algorithm go through all the 2^{n-1} wrong choices before finally getting to the right one, thus the algorithm needs at least $\Omega(2^{n-1})$ time to find the solution.

To see why there is only one solution for each example, first notice that in $\Delta(A)$ all the distances are in $(0,3n\varepsilon] \cup [1-6n\varepsilon,1]$. So all the points must be in $[0,3n\varepsilon] \cup [1-3n\varepsilon,1]$. In particular, we observe that after placing the first $3n+2$ points (after we find E), the largest remaining distance is $1-(2n+1)\varepsilon$, so the last $2n$ points must be in $[1-3n\varepsilon, 1-(2n+1)\varepsilon] \cup [(2n+1)\varepsilon, 3n\varepsilon]$. Because of the discreteness of the distances in $\Delta(A)$, there are only $2n$ positions in the above intervals and no two points can have the same position, so the last $2n$ points must be chosen the same as $A_4 \cup A_5$. When the choice of B and C is wrong (*i.e.*, $E \neq D$), suppose, by the purpose of deriving a contradiction, that we have a solution. This solution must be $S := A_1 \cup A_2 \cup A_4 \cup A_5 \cup \{0,1\} \cup E$, and by Proposition 1, we can easily check that $\Delta(S) \neq \Delta(A)$, which is a contradiction. So there is no solution when the choice of B and C is wrong. And when B and C are correct, there is only one solution (since the last $2n$ positions are fixed). This shows there is a unique solution for each example. □

OPEN PROBLEMS

This example shows that the backtracking algorithm for the turnpike reconstruction problem is exponential. The question remains whether the turnpike reconstruction problem is NP-complete? Finding a strong polynomial algorithm or proving it is NP-complete is still an open problem.

ACKNOWLEDGMENT

The author wishes to thank Pavel Pevzner and Webb Miller for suggesting the problem and for help in writing this paper.

REFERENCES

Lemke, P., and Werman, M. 1988. On the complexity of inverting the autocorrelation function of a finite integer sequence, and the problem of locating n points on a line, given the unlabeled distances between them. (Unpublished manuscript).

Rosenblatt, J., and Seymour, P.D. 1982. The structure of homometric sets. *SIAM J. Algebraic Discrete Methods* 3, 343–350.

Skiena, S.S., and Sundaram, G. 1994. A partial digest approach to restriction site mapping. *Bull. Mathemat. Biol.* 56, 275–294.

Skiena, S.S., Smith, W.D., and Lemke, P. 1990. Reconstructing sets from interpoint distances. Proc. *Sixth ACM Symp. Computat. Geom.* 332–339, ACH Press, New York.

Weiss, M.A. 1992. *Data Structures and Algorithm Analysis*, Benjamin/Cummings Publishing Company, Inc., Redwood City, California.

Address reprint requests to:
Dr. Zheng Zhang
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802

Received for publication August 25, 1993; accepted October 12, 1993.