An Algorithm for the Shortest Superstring Problem: A Dynamic Programming Approach

Pattara Rujeerapaiboon
Alongkot Burutarchanai
Prabhas Chongstitvatana

Department of Computer Engineering, Chulalongkorn University
Bangkok, 10330, Thailand
Email: Prabhas@chula.ac.th

Abstract

The Shortest Superstring Problem, also known as SSP, is a computational problem to construct the shortest string that contains every string in the given set. SSP is an important optimization problems as it has many applications in various fields including computing science, information technology and bioinformatics. SSP is proven to be NP-complete, thus, it cannot be solved by any efficient algorithms unless P=NP. Recently, machine learning and greedy procedures are major paradigms used for handling the problem. In this paper, the authors propose a dynamic programming algorithm which is accurate and computationally simple. Also, the algorithm can be adjusted or extended to handle the problem in specific domains.

Key Words: SSP, shortest superstring problem, data compression, genome assembly, fragment assembly

1. Introduction

SSP has posed immediate applications in many classical domains for a long time. For example, data compression, which is the process of encoding information with fewest information bearing units, in computing science and information technology. Also, the problem is of increasing importance in recent years as it is the generalization of *fragment assembly* in the newly explored bioinformatics domains.

DNA or Deoxyribonucleic Acid is a nucleic acid consisting of four different nucleotides, namely A or Adenine, T or Thymine, C or Cysteine and G or Guanine. Due to technological limits, an accurate long DNA molecule cannot be directly identified. Instead, a whole molecule is reconstructed from several smaller parts of it, which are up to 600 bases long for best accuracy according to current techniques [1]. The reconstruction is named fragment assembly and is a special case of SSP. However, some assumptions have to be made upon the SSP approach when dealing with fragment assembly as some of DNA molecules consist of repetitive parts called repeats.

SSP is proven to be NP-complete which shows that the problem itself is inherently computationally complex. Many attempts are devoted to invent an acceptably accurate algorithm [3-7]. In this paper, the authors develop an accurate and efficient graph algorithm on the basis of dynamic programming, which is adapted from Floyd-Warshall algorithm to some extent.

2. Algorithms

2.1 Floyd-Warshall Algorithm

Floyd-Warshall algorithm [2] is an algorithm, on the basis of dynamic programming, to find the shortest path between all pairs of vertices in a directed graph that has no negative-length cycles. That is, given a directed graph of n vertices, with no negative-length cycles, in the form of $n \times n$ distance matrix d_{ij}, the algorithm is based on the following recurrent relation for $i, k = 1, \dots, n$ and $i, k \neq j$

$$d_{ij} = min\{d_{ik}, d_{ij} + d_{kj}\} \quad (1)$$

The time complexity of a trivial implementation of the algorithm is $O(n^3)$

2.2 Preliminaries to the Proposed Algorithm

In this section, the authors will introduce functions, sets, graphs and other terminologies used to describe the algorithm.

Given a finite set of non-null strings $S = \{s_1, s_2, \dots, s_n\}$ where $s_i \in \sum^*$ for $1 \leq i \leq n$, the

reduced set $S^{'}$ is the largest set that preserves the following conditions.

1. $\forall x \in S^{'}(x \in S)$

2. $\forall a, b \in S^{'} \wedge a \neq b$
$\left(\forall q, r, s, t \in \sum^{*} \begin{pmatrix} qar \neq b \\ sbt \neq a \end{pmatrix}\right)$

Let μ be the overlap function. $\mu: \sum^{*} \times \sum^{*} \rightarrow I^{+} \cup \{0\}$ preserves the following condition.

$$\exists x, y, z \begin{pmatrix} a = xy \wedge b = yz \wedge |y| = w \wedge \\ \forall r(a = qr \wedge b = rs \rightarrow |r| \leq w) \end{pmatrix} \rightarrow \mu(a, b) = w$$

Let ρ be the concatenating function. $\rho: \sum^{*} \times \sum^{*} \rightarrow \sum^{*}$ preserves the following condition.

$$\exists x, y, z\big(a = xy \wedge b = yz \wedge |y| = \mu(a, b)\big) \rightarrow \rho(a, b) = xyz$$

Let ρ^{*} be the composite concatenating function. Given $n \geq 3$ and $a_i \in \sum^{*}$ for $i \in I^{+}$, ρ^{*} preserves the following conditions.

1. $\rho^{*}(a_1, a_2) = \rho(a_1, a_2)$

2. $\rho^{*}(a_1, a_2, \dots, a_n) = \rho(\rho^{*}(a_1, a_2, \dots, a_{n-1}), a_n)$

Let $\alpha \in [0,1]$ and $\beta \in I^{+}$ be parameterized variables. Given α, β, a, b where $a, b \in S^{'}$ and $a \neq b$ and γ *is* the length of the shortest string in $S^{'}$, let P_{ab} be the possibility set under the reduced set $S^{'}$. P_{ab} preserves the following conditions.

1. $P_{ab} = \{p_{ab1}, p_{ab2}, \dots, p_{ab\beta}\}$

2. $\forall i \in \{1,2, \dots, \beta\}\left(\begin{matrix} \exists x, y\left(a = xy \wedge |y| = \frac{\gamma \alpha i}{\beta + 1}\right) \\ \rightarrow h_{abi} = y \end{matrix}\right)$

3. $\forall i \in \{1,2, \dots, \beta\}\left(\begin{matrix} \exists q, r, s \\ \left(\begin{matrix} b = qrs \wedge \\ |q| = \mu(a, b) \wedge \\ |r| = min \\ \left(\begin{matrix} \gamma \alpha \\ \left(\left(1 - \frac{i}{\beta + 1}\right), \right) \end{matrix}\right) \\ |b| - |q| \end{matrix}\right) \\ \rightarrow h_{abi} = r \end{matrix}\right)$

4. $\forall i \in \{1,2, \dots, \beta\}(p_{abi} = h_{abi} + t_{abi})$

Let $G = (V, E)$ be the complete directed graph (that is there are edges from every vertices to every other vertices in the same graph) under the reduced set $S^{'}$ called validity graph. Each vertex in G represents different strings in $S^{'}$ and G preserves the following conditions.

1. $|S^{'}| = m \rightarrow V = \{v_1, v_2, \dots, v_m\}$

2. $\exists y \in P_{ab}, x, z(xyz \in S^{'}) \rightarrow (v_a, v_b) = 1$

3. $\forall y \in P_{ab}, x, z \in \sum^{*}\left(xyz \notin S^{'}\right) \rightarrow (v_a, v_b) = 0$

Let σ be the repeat eliminating function under the validity graph G. Given $a, b, c \in S^{'}$, σ preserves the following conditions.

1. $\sigma(a, b) = (a, b)$

2. $\rho(a, b) = x \wedge \exists y, z(ycz = x) \rightarrow \sigma(a, b, c) = (a, b)$

3. $\rho(a, b) = x \wedge \forall y, z \in \sum^{*}(ycz \neq x) \rightarrow \sigma(a, b, c) = (a, b, c)$

Let σ^{*} be the composite repeat eliminating functions under the validity graph G. Given $n \geq 4$ and $a_i \in S^{'}$ for $i \in I^{+}$, σ^{*} preserves the following conditions.

1. $\sigma^{*}(a_1, a_2) = \sigma(a_1, a_2)$

2. $\sigma^{*}(a_1, a_2, a_3) = \sigma(a_1, a_2, a_3)$

3. $\sigma^{*}(a_1, a_2, \dots, a_{n-1}) = (a_1^{'}, a_2^{'}, \dots, a_l^{'}) \wedge \rho^{*}(a_1^{'}, a_2^{'}, \dots, a_l^{'}) = x \wedge \exists y, z(ya_n z = x) \rightarrow \sigma^{*}(a_1, a_2, \dots, a_n) = (a_1^{'}, a_2^{'}, \dots, a_l^{'})$

4. $\sigma^{*}(a_1, a_2, \dots, a_{n-1}) = (a_1^{'}, a_2^{'}, \dots, a_l^{'}) \wedge \rho^{*}(a_1^{'}, a_2^{'}, \dots, a_l^{'}) = x \wedge \forall y, z \in \sum^{*}(ya_n z \neq x) \rightarrow \sigma^{*}(a_1, a_2, \dots, a_n) = (a_1^{'}, a_2^{'}, \dots, a_l^{'}, a_n)$

Let τ be the scoring function under the validity graph G. Given $n \geq 2$ and $a_i \in S^{'}$ for $i \in I^{+}$ and $a_i \neq a_j$ if $i \neq j$, τ preserves the following condition.

$$\sigma^{*}(a_1, a_2, \dots, a_n) = (a_1^{'}, a_2^{'}, \dots, a_l^{'}) \rightarrow \tau(a_1, a_2, \dots, a_n) = \frac{\sum_{i=1}^{l-1}\left(v_{a_i^{'}, v_{a_{i+1}^{'}}}\right)}{l-1}$$

2.3 The Algorithm

Given a finite set of non-null strings $S = \{s_1, s_2, \dots, s_n\}$, the algorithm follows the sequence of

instructions.

1. Let the best superstring B be null.
2. Reduce S into a reduced set S' by repeatedly eliminating a string of S which is a substring of one or more other strings (rather than itself) until no such string exists. Let m be the size of S'.
3. Construct a validity graph G from S' where each vertex represents different strings in S'. For each edge, the weight is 1 if collapsing its initial vertex with terminal vertex produces a valid link (a portion of resulting string around the overlapping region which is a substring of some members of S'). Otherwise, the edge is 0.
4. Find all-pair most accurate acyclic path by exploiting Floyd-Warshall algorithm with slight modification (instead of the path's length, the scoring function τ is used as the metric).
5. Transform each $m(m-1)$ path into a superstring (by orderly using the composite repeat eliminating function σ^* and the composite concatenating function ρ^*.
6. Choose the best superstring in step 4 by selecting the string that contains most number of strings in S' (if there are more than one such strings, choose the shortest one). Compare the chosen string with B using the same method and assign the better string to B.
7. For each vertex in G find the most accurate acyclic path (the scoring function τ is used as the metric) that begins with the vertex (if there are more than one such paths, choose one that represents the longest superstring) and transform it into a superstring. At this point, we get m superstrings. Let S, whose size is m, be the set of these string
8. Continue step 1 until the exit condition (such as maximum number of iterations or maximum number of iterations that B is not changed) is reached.
9. Return B.

3. Performance

A string of length 1000 over a set of alphabet whose size is 4 is sheared into 50 strings whose length are between 100-120. Parameters α and β are accordingly set to 1.0 and 3. The maximum number of iterations and the maximum number of iterations that the best solution does not improve are 5 and 2 respectively. Levenshstein distance, which is the minimum number of operations {character insertion, deletion and substitution}[8] required to transform one string to another string, together with an original string length which the authors define as l, is used to measure similarity between an original string and a string constructed by the algorithm.

In the following performance evaluation, a noise-free environment is assumed (exploitation under a less ideal environment will be further discussed). Ten experiments are conducted. The results are presented in table 1.

No.	Levenshtein Distance (d)	Accuracy $\left(1-\frac{d}{l}\right)\times 100\%$
1.	0	100.00
2.	72	92.80
3.	0	100.00
4.	78	92.20
5.	1	99.90
6.	0	100.00
7.	34	96.60
8.	74	92.60
9.	8	99.20
10.	0	100.00

Table 1 Experimental results

From experiments, the average accuracy of the algorithm is 97.33%. By a statistical hypothesis testing, the algorithm can reconstruct an original string from its corresponding smaller strings with not less than 99.00% average accuracy over a 0.05 significant level.

4. Time Complexity Analysis

For the sake of simplicity, in this analysis, the authors will not take time complexity of string processing into account as it varies greatly depends on which string processing algorithms are used.

The time complexity of the algorithm is dominated by step 4. As previously discussed in section 2.1, Floyd-Warshall algorithm's time complexity is $O(n^3)$ where n is a number of vertices in a graph under consideration. Step 4 requires circuit check, which is performed every time each current shortest path is about to be updated, in additional to normal Floyd-Warshall algorithm. A fast sorting algorithm whose time complexity is $O(n\,log\,n)$ and searching for consecutive identical vertices can be exploited to identify whether the path contains circuits. By using previous technique and the fact that each current shortest path has no more than n vertices, the time complexity of the algorithm is $O(n^4\,log\,n)$. However, this is a loose time complexity approximation and the exact time complexity is expected to be substantially less.

5. Conclusions

The algorithm is computationally simple and accurate. Also, it can be adjusted or extended to handle SSP in specific domains like computing science, information technology or bioinformatics. Moreover, the algorithm can be exploited even in the presence of noise. That is if maximum error rate in

strings is known, the string matching condition can be adjusted according to that rate.

The algorithm may yield better performance if an overlap graph is used instead of the validity graph. In this context, the overlap graph is the complete directed graph. An edge from vertex v_a to vertex v_b is $\frac{\mu(a,b)}{min\,(|a|,|b|)}$ where a and b are strings represented by v_a and v_b respectively. Further investigation should be conducted to prove or disprove this hypothesis.

Future works focus on enhancing the algorithm to handle several or long repetitive parts in strings.

6. References

[1] M. Pop, S.L. Salzberg, and M. Shumway, "Genome sequence assembly: algorithms and issues", *IEEE Computer*, 35, 2002, pp.47-54.

[2] C.H., Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, 1998.

[3] A. Blum, et al., "Linear approximation of shortest superstring", *Journal of the Association for Computing Machinery*, 41, 1994, pp. 630-647.

[4] D. Breslauer, T. Jiang, and Z. Jiang, "Rotations of periodic strings and short superstring", *Journal of Algorithms*, 24, 1997, pp. 340-353.

[5] C. Armen and C. Stein, "A $2\frac{2}{3}$ superstring approximation algorithm", *Discrete Applied Mathematics*, 88, 1998, pp. 29-57.

[6] Z. Sweedyk, "A $2\frac{1}{2}$- approximation algorithm for shortest superstring", *SIAM Journal on Computing*, 29, 1999, 954-986.

[7] M.K. Goldberg and D.T. Lim, "A learning algorithm for the shortest superstring problem", in: *Proceedings of the Atlantic Symposium on Computational Biology and Genome Information Systems and Technology, Durham, NC*, 2001.

[8] N.C. Jones and P.A. Pevzner, *An Introduction to Bioinformatics Algorithms*, MIT Press, 2004.