Submitted to the Workshop of Algorithms and Computational Molecular Biology

A New Greedy Algorithm for Longest Common Subsequence Problems

Yin-Te Tsai*
Department of Computer Science and Information Management
Providence University
Shalu, Taiwan 433, R.O.C.
Email:yttsai@pu.edu.tw
Tel:+886-4-26328001x13411
Fax:+886-4-26324045

Jeh-Ting Hsu
Department of Computer Science and Information Management
Providence University
Shalu, Taiwan 433, R.O.C.
Email:s8511041@algo.cs.pu.edu.tw

Abstract

The longest common subsequence (LCS) problem, a famous NP-Hard problem, is to find the common subsequence with maximum length on a set S of n strings of length m over an alphabet set Σ. One of important applications of this problem is to measure the similarity of biological sequences in molecular biology. This paper introduces a new algorithm for LCS problems, which uses the ant colony optimization approach and runs in $O(nm^4 \log m + Rlmn)$ time when $m > n$ and $m > |\Sigma|$, where R denotes the number of iterations and l denotes the number of ants. Our algorithm is also a greedy algorithm based on Hamiltonian cycles. In this paper, we prove that this algorithm has performance ratio of $|\Sigma|$ and show that the algorithm obtains better solutions than previous algorithms from our initial experimental results.

Keywords: Longest common subsequence problems, Approximation algorithms, Greedy algorithms, Ant colony optimization, Hamiltonian cycles

*Corresponding author: Yin-Te Tsai

1 Introduction

Let $S = \{s_1, s_2, s_3, \ldots, s_n\}$ be a set of n strings of length m over an alphabet set Σ, where $m > n$ and $m > \Sigma$. String s is called a common subsequence of s_is' iff s can be obtained by deleting zero or more symbols from each s_i, where i=1, 2, ..., n. The longest common subsequence (LCS) problem is to find a common subsequence of s_is' of maximal length.

LCS has many important applications in different areas. In molecular biology, LCS is an appropriate method for measuring the similarity of biological sequences [14]. When we want to know how homologous those DNA or protein sequences are, we can calculate the maximum number of identical symbols among them. That is exactly the longest common subsequence of them. Moreover, we can even compare the evolutionary distances of those sequences only by using a special LCS algorithm [1].

The LCS problem on two strings (2LCS) is polynomial-time solvable and has received much attention. Many authors have tried for years to use the dynamic programming technique on the 2LCS problem [16, 8, 13, 2]. But most of them still needed $O(mn)$ in the worse case. However, the LCS problem on n strings was shown to be NP-hard [12] (even on a binary alphabet). Exact algorithms using dynamic programming for solving LCS problems appear in [7, 9].

Jiang and Li [10] showed that there exists a constant $\delta > 0$ such that, if LCS has a polynomial-time approximation algorithm with performance ratio n^δ, then P=NP. They proposed Long Run algorithm for approximating LCS problems. This algorithm finds the longest common subsequence consisting of only one distinct symbol over Σ. For example, given strings `aaabc`, `bbbaac` and `ccaa`, Long Run will output `aa`, because `aa` is the longest common subsequence with a single symbol.

Bonizzoni, Vedova and Mauri [3] proposed another approximation algorithm called Expansion Algorithm (EA). Given a set S of strings, EA performs the following four steps: (1) Compute all streams of S of maximum length 2. (2) Compress each sequence into its longest stream. (3) Use

the Huffman-like greedy algorithm to obtain a common stream T of S by merging pairwise streams into a common subsequence. (4) Expand all streams of S of maximum length 2 and all substrings of T for determining the final common subsequence.

The performance ratio of an approximation algorithm A for LCS problems is the ratio between the length of longest common subsequence and the length of common subsequence found by A. It was proved that the performance ratio of EA and Long Run are both $|\Sigma|$.

Tsai and Hsu [15] proposed an approximation algorithm MSTG for LCS problems. This algorithm is a greedy algorithm based on a minimal spanning tree (MST). Although MSTG has the same performance ratio and time complexity as that of EA, the authors showed that MSTG have better solutions from their experimental results. Table 1 summaries recent results for LCS problems.

Ant Colony Optimization(ACO) is a general-purpose heuristic algorithm that can be applied to solve different combinatorial optimization problems [6, 5]. The ACO algorithm imitates the foraging behavior of real ant colonies. When ants search for the sources of food or from food sources back to the nest, they lay a chemical substance called pheromone. The closer trail will have more pheromone than the farther one will have, accordingly, the ants can reach the closer trail much quicker. The ants smell pheromone and choose the stronger pheromone trail to go. This is the exact way that they find the shortest path between the food and the nest. ACO was proposed firstly by Dorigo [4] and can be applied to combinatorial problems by following four steps:

1. Define an appropriate graph of the problem that can be searched by ants.
2. Define the autocatalytic feedback process.
3. Define a heuristic algorithm that can work on the graph.
4. Define a constraint satisfaction method.

ACO has already been adapted to solve a number of different combinatorial problems and

network routing problems, such as the shortest common supersequence problem (e. g. traveling salesman, quadratic assignment, job-shop scheduling, vehicle routing, sequential ordering, graph coloring, etc.) [5].

In this paper, one approximation algorithm for solving the LCS problem is proposed. This algorithm is an Hamiltonian-cycle-based greedy algorithm, which searches for better candidate sequences of LCS by finding better Hamiltonian cycles. Our algorithm HCG uses Ant Colony Optimization (ACO) approach for Hamiltonian cycles and has time complexity $O(nm4\log m + Rlmn)$, where R is the user-defined number of iteration and l is the number of ants. HCG algorithm has performance ratio of $|\Sigma|$, and has better experimental results compared with the previous approximation algorithms.

The remainder of this paper is organized as follows. Section 2 will introduce our HCG algorithm and analyze its time complexity. The experimental results are given in Section 3. Future research directions and works are provided in Section 4.

2 The Algorithm

In this section, we shall introduce our new approximation algorithm for LCS problem. Let σ^i denote a string composed of i consecutive σs' for $\sigma \in \Sigma$, where i is a positive integer. String $\sigma_1\sigma_2\ldots\sigma_k$ with $\sigma_j \in \Sigma$, $1 \le j \le k$, is a stream of string s if $\sigma_i \neq \sigma_{i+1}$ for all $1 \le i < k$ and $\sigma_1^{i_1}\sigma_2^{i_2}\ldots\sigma_k^{i_k} = s$ for positive integers i_j's. String $s = \sigma_1^{j_1}\sigma_2^{j_2}\ldots\sigma_k^{j_k}$ with $\sigma_l \in \Sigma$, $1 \le l \le k$, is called the *expand* of stream $\sigma_1\sigma_2\ldots\sigma_k$ with respective to a set S of strings if s is the common subsequence of S and j_ls' are maximized for $1 \le l \le k$.

The minimum-spanning-tree-based greedy (MSTG) algorithm, proposed in [15], uses a MST as a guide tree for merging the common subsequences. The details of this algorithm is described in Figure 1. In MSTG, phase 0 runs in $O(mn)$ time, while Phase I needs $O(n^2m^2) + O(n^2\log n) + O((n-1)m^2) = O(n^2m^2)$ time, where Step 2 runs in $O(n^2m^2)$ time, $O(n^2\log n)$ time is needed in

Algorithm: MSTG
Input: A set $S = \{s_1, \ldots, s_n\}$ of n strings of length m
Output: A common subsequence for S

1. // Phase 0
   Compute $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ where $\mathbf{s_i}$ is the longest stream of s_i.

2. // Phase I
   Compute the longest common subsequence $L_{\mathbf{s}_i,\mathbf{s}_j}$ for $\mathbf{s}_i$ and $\mathbf{s}_j$, where $1 \leq i < j \leq n$.

3. Let $G = (V, E)$ be a complete graph with cost function $c : E \to \mathbb{R}$, where $V = \{1, 2, \ldots, n\}$ is the vertex set, E is the edge set and $c_{ij} = m - |L_{\mathbf{s}_i,\mathbf{s}_j}|$ for $1 \leq i \neq j \leq n$.
   Let $\{e_1, e_2, \ldots, e_{n-1}\}$ be the accepted edge sequence in constructing a minimum spanning tree using Kruskal's algorithm.

4. Let r_i and l_i denote the two end vertices of edge e_i.
   Let $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$, where T_j is the labeled tree containing root $\mathbf{s}_j$ only, where $1 \leq j \leq n$.
   Let i=1.

5. Let T' and T'' denote the labeled trees in $\mathcal{T}$ containing $\mathbf{s}_{r_i}$ and $\mathbf{s}_{l_i}$, respectively.
   Let s' and s'' denote the labels of roots of T' and T'' respectively.
   Create a new node y with label $L_{s',s''}$.
   Construct a tree T''' rooted at y by linking y with the roots of T' and T''.
   Let $\mathcal{T} = \mathcal{T} - \{T', T''\} + \{T'''\}$.
   Let $i = i + 1$.

6. Repeat Step 5 n-2 times.

7. // Phase II
   Let $\mathcal{S}$ denote the all streams of S of length ≤ 2.
   Add to $\mathcal{S}$ all substrings of the label of root of tree in $\mathcal{T}$.

8. Find the longest *expand* of $w \in \mathcal{S}$ with respect to S.

9. Report the result sequence.

Figure 1: MSTG algorithm

Step 3, and Step 5 is repeated n-1 times of $O(m^2)$ time each. Phase II which is the same process as **ExpandArbitray** function of EA in [3] runs in $O((|\Sigma|^2 + m^2)nm^2 \log m)$ time, where it takes $O(nm^2 \log m)$ time to get an *expand* and there are at most $|\Sigma|^2 + m^2$ steams in $\mathcal{S}$. When $m > n$ and $m > |\Sigma|$, the total time complexity is $O(nm^4 \log m)$.

Our algorithm assumes that there are l ants, denoted by a_1, ..., a_l. Let $e(i,j)$ be the edge linking i and j. Let c_{ij} be the cost of edge $e(i,j)$. Let η_{ij} be the quantity of visibility on edge $e(i,j)$. Let τ_{ij} be the quantity of pheromone on edge $e(i,j)$. Let $\Delta\tau_{ij}^k$ be the quantity of pheromone laid on edge $e(i,j)$ by ant a_k. Let N_k be a set of unvisited vertices by ant a_k. Let α denote the relative importance of the trail. Let β denote the relative importance of the visibility. Let ρ denote the trail persistence and $1-\rho$ the trail evaporation. Note that $\alpha \geq 0$, $\beta \geq 0$ and $0 \leq \rho < 1$. Let P_{ij}^k be the probability that ant a_k chooses edge $e(i,j)$ to be on it's trail, which is defined as follows:

$$P_{ij}^k = \begin{cases} \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{v \in N_k} (\tau_{iv})^\alpha (\eta_{iv})^\beta} & \text{if } j \in N_k; \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Our algorithm is the Hamiltonian-cycle-based greedy (HCG) algorithm for LCS problems. The complete description of it is listed in Figure 2. The step 4 uses the ACO heuristics described in [6] for Hamiltonian cycles. In HCG, phase 0 runs in $O(mn)$ time to compute streams. Phase I needs $O(n^2m^2) + O(n^2) + O(Rlmn) = O(n^2m^2 + Rlmn)$ time, where Steps 2, 3 and 4 run in $O(n^2m^2)$, $O(n^2)$ and $O(Rlmn)$ time, respectively. Phase II is the same process as Phase II of MSTG and runs in $O(nm^4 \log m)$ time for $m > n$ and $m > |\Sigma|$. Therefore, it is $O(nm^4 \log m + Rlmn)$ time in total for $m > n$ and $m > |\Sigma|$, where R is the number of iterations. The time complexity becomes $O(nm^4 \log m)$ if $R = O(m)$ and $l = O(m)$.

Long Run algorithm proposed in [10] had shown that it has performance ratio of Σ. It is easy to know that the common subsequence generated by our HCG algorithm has at least the same length of that of Long Run algorithm. Therefore, we have the following result immediately.

Theorem 1 *HCG algorithm has performance ratio of* $|\Sigma|$.

Algorithm: HCG
Input: A set $S = \{s_1, \ldots, s_n\}$ of n strings of length m, α, β, ρ, l and R
Output: A common subsequence for S

1. // Phase 0
   Compute $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ where $\mathbf{s_i}$ is the longest stream of s_i.

2. // Phase I
   Compute the longest common subsequence $L_{\mathbf{s}_i,\mathbf{s}_j}$ for $\mathbf{s}_i$ and $\mathbf{s}_j$, where $1 \le i < j \le n$.
   Let $G = (V, E)$ be a complete graph with cost function $c : E \to \mathbb{R}$, where $V = \{1, 2, \ldots, n\}$ is the vertex set, E is the edge set and $c_{ij} = m - |L_{\mathbf{s}_i,\mathbf{s}_j}|$ for $1 \le i \ne j \le n$.

3. Set $\tau_{ij} = 1$ for $1 \le i \ne j \le n$.
   Set $\eta_{ij} = 1/c_{ij}$ for $1 \le i \ne j \le n$.
   Set $\mathcal{D}$ to be an empty sequence.

4. Reapeat the following steps R times

   4.1 Set $\Delta\tau_{ij}^k = 0$ for $1 \le i \ne j \le n$ and $1 \le k \le l$.
   Set $N_k = \{1, \ldots, m\} - \{k\}$ for $1 \le k \le l$.
   Place ant a_k to a random vertex v_k for $1 \le k \le l$.
   Set $t_k =< v_k >$.

   4.2 Repeat the following steps $m - 1$ times.

   4.2.1 For $1 \le k \le l$,
   (a) find the j_k such that $P_{v_k j_k}^k = \max\{P_{v_k j}^k | j \in N_k\}$;
   (b) set $N_k = N_k - \{j_k\}$;
   (c) set $v_k = j_k$;
   (d) append v_k to the vertex sequence t_k.

   4.3 For $1 \le k \le l$,
   (a) use GCS algorithm (see Figure 3) to compute the candidate sequence d_k for t_k with respect to $\mathbf{S}$;
   (b) if $|d_k| > |\mathcal{D}|$, set $\mathcal{D} = d_k$.

   4.4 For $1 \le i \ne j \le n$ and $1 \le k \le l$,
   (a) set $\Delta\tau_{ij}^k = 0$;
   (b) if $e(i, j) \in d_k$, set $\Delta\tau_{ij}^k = |d_k|/|\mathcal{D}|$.
   Set $\Delta\tau_{ij} = \sum_{1 \le k \le l} \Delta\tau_{ij}^k$ for $1 \le i \ne j \le n$.
   Set $\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}$ for $1 \le i \ne j \le n$.

5. // Phase II
   Let $\mathcal{S}$ denote the all streams of S of length ≤ 2.
   Add to $\mathcal{S}$ all substrings of $\mathcal{D}$.

6. Find the longest *expand* of $w \in \mathcal{S}$ with respect to S.

7. Report the result sequence.

Figure 2: HCG algorithm

Algorithm: GCS
Input: A set $S = \{s_1, \ldots, s_n\}$ of n strings of length m, and a vertex sequence $Q = \{q_1, \ldots, s_n\}$
Output: A candidate sequence

1. If $|S| = 1$, return s_1.
2. Set $D =$ the LCS of s_1 and s_2.
3. For $i = 3$ to n do
   Set $D =$ the LCS of D and q_i;
4. Return D.

Figure 3: GCS algorithm

3 Experimental Results

We shall give our initial experimental results and the comparative study with EA in [3] and MSTG in [15]. Our experiments can be divided into two parts. In the first part, we test DNA and protein sequences selected from some conserved regions of bacteria and generated from an evolution process according to the one-parameter substitution scheme, Jukes-Cantor model [11]. Besides, we also discuss the performance in sequences of four different lengths of average lengths 250, 500, 750, and 1000. In the second part, we test three groups of sequences with different divergence. All of our experimental data are obtained from National Center for Biotechnology Information (NCBI) and sequence lengths are all ranged between 400 and 500. As for parameters of HCG algorithm, the setting values are α= 2, β=2, ρ= 0.3, $R = n$ and $l = n$.

Table 2 shows the length of common subsequences found by EA, MSTG and HCG algorithms on 10 sets of DNA sequences and 10 sets of protein sequences. This table demonstrates that HCG almost has better solutions in the data sets.

In addition, the experimental results on strings generated by JC model are listed in Table 2. Four sets of probabilistic DNA and protein sequences are tested, where two sets of 5 sequences and two sets of 10 sequences. The results also demonstrate that HCG has better solutions than others in all data sets.

Table 4 and Table 5 show the length of LCS found by EA, MSTG, and HCG in four groups of sequences of average lengths 250, 500, 750, and 1000, where each group consists of 5 sequences. These results show that HCG algorithm outperforms other algorithms.

Finally, we consider three groups of sequences with different divergence. Given a set of sequences of length m, the divergence degree is defined by the following formula:

$$(LLCS - SLCS)/m,$$

where $LLCS$ is the length of the longest one of all pairwise LCSs and $SLCS$ is the length of the shortest one of all pairwise LCSs. Each group contains 10 sequences with the same divergence. The experimental results for convergence degrees 10%, 20%, and 30% are shown in Table 6, Table 7 and Table 8, respectively. The results of HCG are better than others, especially on DNA sequences with higher divergence and protein sequences.

4 Concluding Remarks

This paper have proposed a new approximation algorithm HCG for LCS problems, which is a greedy algorithm based on Hamiltonian cycles and runs in $O(nm^4 \log m + Rlmn)$ time when $m > n$ and $m > |\Sigma|$, where R denotes the number of iterations, l denotes the number of ants. We have proven that HCG has performance ratio of $|\Sigma|$ and shown that the algorithm obtains better solutions than previous algorithms from our initial experimental results.

More experiments must be performed to further determine the practicability of HCG. One research direction is to find the tight performance ratio for HCG. To design better approximation algorithms for LCS problems by new approach is worthy to exploit.

References

[1] L. Bergroth, H. Hakonen, and T. Raita. New approximation algorithms for longest common subsequences. In *SPIRE*, pages 32–40, 1998.

[2] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *SPIRE*, pages 39–48, 2000.

[3] P. Bonizzoni, G. D. Vedova, and G. Mauri. Experimenting an approximation algorithm for the lcs. *Discrete Applied Mathematics*, 110(1):13–24, 2001.

[4] A. Colorni, M. Dorigo, and V. Maniezzo. An investigation of some properties of an ant algorithm. In *the Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92)*, pages 509–520, 1992.

[5] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, 1999.

[6] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:29–41, 1996.

[7] K. Hakata and H. Imai. The longest common subsequence problem for small alphabet size between many strings. In *Lecture Notes in Computer Science 650, Springer Verlag*, pages 469–478, 1992.

[8] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24:664–675, 1977.

[9] R. W. Irving and C. B. Fraser. Two algorithms for the longest common subsequence of three (or more) strings. In *Lecture Notes in Computer Science 644, Springer Verlag*, pages 214–229, 1992.

[10] T. Jiang and M. Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM Journal on Computing*, 24(5):1122–1139, 1995.

[11] W. H. Li. *Molecular Evolution.* Sinauer Assoc., 1997.

[12] D. Maier. The complexity of some problems on subsequences and supersequences. *J. ACM*, 25:322–336, 1978.

[13] W. J. Masek and M. S. Paterson. A faster algorithm computing string edit distances. *J. Comput. System Sci.*, 20:18–31, 1980.

[14] T. Smith and M. Waterman. Indentification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[15] Y. T. Tsai and J. T. Hsu. An approximation algorithm for multiple longest common subsequence problems. In *the Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002), Vol XII*, pages 456–460, 2002.

[16] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21:168–173, 1974.

Problem	**Algorithm**	**Results**
2LCS	[16]	$O(nm)$ time
	[8]	$O(rn + n \log n)$ time
	[13]	$O(nm/\log n)$ time
LCS	[7]	ea: $O(m\|\Sigma\|n + \lambda\|\Sigma\|n(\log^{n-3} m + \log^{n-2} \|\Sigma\|))$ time
	[9]	ea: $O(nr(m-r)^{n-1} + n\|\Sigma\|m)$ time
	[10]	aa, pr: $\|\Sigma\|$, $O(mn)$ time
	[3]	aa, pr: $\|\Sigma\|$, $O(nm^4 \log m)$ time
	[15]	aa, pr: $\|\Sigma\|$, $O(nm^4 \log m)$ time
	this paper	aa, pr: $\|\Sigma\|$, $O(nm^4 \log m + Rlnm)$ time

Table 1: Previous algorithms and results, where r denotes the length of LCS, λ denotes the number of dominant matches, R denotes the number of iterations, l denotes the number of ants, 'pr' stands for 'performance ratio', 'ea' stands for 'exact algorithm', and 'aa' stands for 'approximation algorithm'.

Data No	$\|\Sigma\|$	n	EA	MSTG	HCG
1			169	178	179
2			175	181	179
3	4	5	176	176	181
4			175	177	182
5			179	179	177
6			137	148	150
7			127	131	141
8	4	10	123	131	140
9			128	136	141
10			136	139	141
11			66	75	79
12			47	61	62
13	20	5	69	75	79
14			61	64	66
15			58	69	66
16			43	47	48
16			36	41	41
18	20	10	38	36	42
19			36	36	45
20			34	34	42

Table 2: Experiments on DNA and protein sequences of length 400∼500.

Data No	$\|\Sigma\|$	n	EA	MSTG	HCG
1	4	5	210	211	213
2	4	10	182	192	215
3	20	5	167	183	183
4	20	10	102	113	119

Table 3: Experiments with sequences generated by JC model of length 400∼500.

Data No	Length	EA	MSTG	HCG
1	250	92	91	95
2	500	169	178	179
3	750	292	303	302
4	1000	363	370	373

Table 4: Experiments on DNA sequences of different lengths.

Data No	Length	EA	MSTG	HCG
1	250	32	34	35
2	500	66	75	79
3	750	87	97	105
4	1000	108	118	126

Table 5: Experiments on protein sequences of different lengths.

Data No	$\|\Sigma\|$	n	EA	MSTG	HCG
1	4	10	378	379	378
2	20	10	76	77	82

Table 6: Experiments on DNA and protein sequences with divergence degree 10%

Data No	$\|\Sigma\|$	n	EA	MSTG	HCG
1	4	10	164	165	168
2	20	10	50	51	58

Table 7: Experiments on DNA and protein sequences with divergence degree 20%

Data No	$\|\Sigma\|$	n	EA	MSTG	HCG
1	4	10	160	170	173
2	20	10	48	56	55

Table 8: Experiments on DNA and protein sequences with divergence degree 30%