International Journal of Advance Research in Science and Engineering
Vol. No.6, Issue No. 04, April 2017
www.ijarse.com

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

A HYBRID GENETIC ALGORITHM FOR THE LONGEST COMMON SUB-SEQUENCE OF MULTIPLE SEQUENCES: A REVIEW

Aman Rawat[1], Vishal Thakuria[2], Pradeep Gehlot[3], Deepak Garg[4]

[1,2,3,4]*Dept. of Computer Applications, National Institute of Technology Kurukshetra (India)*

ABSTRACT

The k-LCS (longest common sub-sequence)problem is to find the LCS of k sequences. The k-LCS is difficult when the length and the number of sequences are significant. For solving this problem, in this paper, we have reviewed Expansion algorithm, Best next for maximal available symbol algorithm, Genetic algorithm and Ant colony optimization algorithms. By taking good features of above algorithms we have proposed a hybrid method, which is a combination of Genetic algorithm and Hill climbing algorithm. In our review, we compare our method with expansion algorithm, best next for maximal available symbol algorithm, GA and ACO algorithm.We have introduced a local search heuristic, hill climbing, with the GA to solve the above problem.

Keywords: Crossover, Expansion Algorithm,Genetic Algorithm, Hill Climbing, Mutation.

1. INTRODUCTION

Given k number of sequences, we need to find LCS(Longest Common Sub sequence) from the given sequences in k-LCS problem. Suppose we have a set S of input sequences, S={ s1,s2,….,sk }. If P is a sub-sequence of all S_i for $1 \leq i \leq k$, then P is said to be a common sub-sequence (CS) of set S. For example, consider the set S ={s1 = DPFIPDF, s2 = FPIPDIF, s3 = IPPFDPF}.The sequences DF, FPF and PPDF are common sub-sequences in s1, s2 and s3. If P is the longest in all of the common sub-sequences of S, then P is the LCS of S. LCS problem is a well-known problem in computer science [4, 5, 8, 13]. If the size of S is an arbitrary number, k, where k > 2, the k-LCS problem is an NP-hard problem even over binary alphabet. Using exhaustive search to find the all the CS in k sequences when k is significant is a difficult task. Dynamic programming based solution requires O(n^2) space and time for solving 2-LCS[17]. Heuristic algorithms have been proposed for solving this problem where effective evaluation techniques are used to determine the quality of CS.The expansion algorithm (EA) is proposed by Bonizzoni et al. [1]. The algorithm uses streams to denote each sequence in the input set *S*, and then find the LCS of these streams. A stream is a sequence without contiguously same symbol. However, the performance is not satisfactory. Huang et al. propose a best next for maximal available symbols (BNMAS) algorithm [7] for determining highly possible symbols as elements of LCS based on the occurrence frequency of each common symbol in the input sequences. It is an intuitive method to select common characters from sequences. Various evolutionary algorithms are there to simplify the searching limitations. Ant colony optimization (ACO) algorithm for the k-LCS problem was proposed by Shyu and Tsai [11]. It uses the characteristics of ant searching for food and uses this to find the common sub-sequence in S [3]. It uses adaptation by relieving on the concentration of pheromone and probabilities to decide next symbol to choose to get superior results. The ACO remembers better solutions that have been recovered, then decides the character that is best common character and then appends it to the CS. Genetic Algorithm (GA) relies on the bio-inspired process such as crossover, selection and mutation to solve the optimization problem. An algorithm was proposed by Chiang et al. for solving the k-LCS problem [2]. The chromosomes are regarded as CS and are evolved in further generations using crossover and mutation operators to find the better common sub-sequence. Based on the fitness function the better results from the chromosomes are retained to form the new generation and are used for evolution in the subsequent generations[12]. As the good part of the result is retained, this helps find a better solution in next generations.

III. RELATED WORK

3.1 Dynamic Programming Based Solution for 2 LCS

Needleman and Wunsch [17] used Dynamic Programming to calculate 2-LCS on a matrix of (m+1)×(n+1) .Let C = $c_1c_2 \ldots c_i \ldots c_m$ and D = $d_1d_2 \ldots d_j \ldots d_n$ be the two input strings. Let L(i, j) denote the length of string C and string D, where 0 <=i <=m and 0 <=j <=n. For every entry L(i,j), there will be three cases as stated in the figure 2.1 Based on them, to calculate L(i,j), first L(i-1,j), L(i,j-1) and L(i-1, j-1) needs to be calculated. With each calculation of L(i, j), where 0<=i <=m, 0<=j<=n, we can eventually obtain L(m,n), the LCS length of C and D. So Needleman and Wunsch's algorithm for 2-LCS formula can be represented as follows.

$$L(i,j) = max \left\{ \begin{array}{ll} & if\ i = 0\ or\ j = 0; \\ L(i-1, j-1) + 1 & if\ a_i = b_j; \\ max \left\{ \begin{array}{l} L(i-1,j) \\ L(i,j-1) \end{array} \right\} & otherwise \end{array} \right\} \quad (2.1)$$

Fig. 2.1 Equation for the Algorithm

Algorithm: Dynamic Programming Algorithm of 2-LCS Input: Two sequences *C* and *D*

```
Output: The length of LCS ($C$,$D$)
{Step 1. calculate $L(i,j)$ based on Equation 2.1}
 for $i$ = 0 to|$C$| do
   for $j$ = 0 to |$D$| do
     if $i$ = 0 or $j$ = 0 then
       $L(i,j) \leftarrow 0$
     else if $C[i] = D[j]$ then
       $L(i,j) \leftarrow L(i - 1, j - 1) + 1$
     else
       $L(i, j) \leftarrow \max\{L(i, j - 1), L(i - 1, j)\}$
     end if
end for
 end for
{Step 2. Return the length of $LCS(C, D)$}
$LCS(C, D) \leftarrow L(|C|, |D|)$
```

2.2 The Expansion Algorithm

Expansion Algorithm(EA) was proposed by Bonizzoni et al. [4] for solving the k-LCS problem.The streams are used to represent the sequence in S and then are used to find the LCS. A stream has non contiguous symbols. Initially, common sub-streams having length more than 2 and a longest common sub-stream T of all sequences in S are found . Further, all sub-strings are expanded to obtain the approximate long common sub-sequence of S. Consider S_1 = XXXXYYYXXXXYYX and S_2 = XXXYYYYXXXXYYY. After encoding, $S_1 = X^4Y^3X^4Y^2X$ and $S_2 = X^3Y^4X^4Y^3$. One sub-stream ss of S is XYXY. We expand the initial X, and ss now is X^2YXY. Then examine if ss is a sub-sequence of both the sequence S_1 and S_2. The subsequently expanded sub-streams stated are XY^2XY, XYX^2Y, $XYXY^2$, X^4YXY, X^2Y^2XY, and so on. Finally, we get ss = $X^2Y^2X^4Y^2$. Then, each symbol is expanded with the maximal value using binary search in a way that the expanded substream is the common sub-sequence of S_1 and S_2. Finally we get the ss tas $X^3Y^3X^4Y^2$. For k-LCS when the length of input sequence is n the time complexity is $O(kn^3 \log n)$, . The EA cannot be used when the input sequence is significantly large, it becomes impractical. A better minimum-spanning-tree-based(MSTG) [14]algorithm was proposed by Tsai and Hsu that can be used to find better common stream.

2.3 The Best Next for Maximal Available Symbols

Best Next for Maximal Available Symbols (BN-MAS) was proposed by Huang *et al*. [15] for solving the *k*-LCS problem. In this algorithm, a set *S* having *k* sequences over finite letter Σ. The main aim of the algorithm is to list the count of each different character before a_i in every sequence, for $1 \le i \le n$, and select the one that has the most available characters as the common sub-sequence. After that, remove the selected character and the characters that are beyond the selected character from each sequence, and again write the available symbols till there is no such common character in all the sequences . Available characters can be shows as count of each and every characters before the *i* th character For example, S_1 = XYYXVVXZXXYZ, the available characters of $S_1[12]$(= Z) are two V, three Y , five X and two Z (including symbol Z itself.) So $v[Z,1] = [2, 5, 3, 2]$. In this Algo., we first record the $v(\sigma, j)$ as the available characters of symbol σ in sequence *j*, and we are only focus about the symbols that are located at the rightmost position. For example, consider four sequences in set *S*, that are {S_1 = XYYXVVXZXXYZ, S_2 = VXYZVYVXYXYZ S_3 = VXVYVZXZVYXV, S_4 = VYXYVVXZXVXY}.To calculate $v(Z)$, we have to calculate $v(Z, 1) = [2, 5, 3, 2]$, $v(Z, 2) = [3, 3, 3, 3]$, $v(Z, 3) = [3, 2, 1, 2]$, and $v(Z, 4) = [3, 2, 2, 1]$ initially. Then $v(Z) = \min\{v(Z,1),v(Z,2),v(Z,3),v(Z,4)\} = [2, 2, 1, 1]$. We will Repeat it again then the steps, we can get $v(V) = [2, 1, 2, 0]$, $v(X) = [2, 3, 2, 1]$, and $v(Y) = [2, 2, 2, 1]$. We shows $Sum(v[\sigma])$ as the sum of all elements in $v[\sigma]$. Then we choose the σ with the maximum $Sum(v[\sigma])$ into the common sub-sequence. In this example, we choose the X into the common sub-sequence. After that, we eliminate the character X and all the characters after X in each sequence, we can get S_1' = XYYXVVXZX, S_2' = VXYZVYVXZ, S_3' = VXVYVZXZVY, and S_4' = VYXYVVXZXV. Repeat again the above steps, we will get the common subs-equence as XYVXZX. The time complexity of BNMAS is $O(\sigma^2 kn + \sigma^3 n)$, where $\sigma = |\Sigma|$, $k = |S|$, and n is the length of the longest sequence in S. As the sequence length increases gradually this algorithm under-performs.

2.4 ACO for *k*-LCS

Ant Colony Optimization(ACO) algorithm for solving the *k*-LCS problem was proposed by Shyu and Tsai [21]. The main motive of the algorithm is to discover common characters that are at better positions in the sequences and can help compose a larger common sub-sequence. Each symbol is transformed to one state, and use the ants' pheromone and probabilities as the decision criteria to locate the symbol that must be chosen to form the common sub-sequence. Consider a set *S* having sequences: *S*1=VYYVXXVZVVYZ, *S*2=XVYZXYXVZGYZ, *S*3=XVXYXZVZXYVX.

Step 1: Randomly select *m* sequences ($m < |S|$) as artificial ants. Suppose $m = 1$ and the artificial ant is *S*2. Then each symbol in S2 is transformed into following state: XVYZXYXVZVYZ →(X,1), (V, 2), (Y, 3), (Z, 4), (X, 5), (Y, 6), (X, 7), (V, 8), (Z, 9), (V, 10),(Y, 11), (Z, 12).

Step 2: Randomly select a state as the common sub-sequence.Assume we choose (V,2) as the candidate for common sub-sequence, then we find the leftmost V in each sequence, and mark them as the common character V, common sub-sequence (CS) ← V.*S*1 = VYYVXXVZVVYZ *S*2 = XVYZXYXVZVYZ *S*3 = XVXYXZVZXYVX

Step 3: According to the probability function, select the next state *p* in *S*2 inside length *d*, and then find the closest *p* of V. Suppose $d = 3$, *p* may be (Y,3), (Z,4), (X,5). We select (X,5) as *p*. Then we find the closest X of G in *S*1 and *S*3, and mark them. *S*1 = VYYVXXVZVVYZ, *S*2 = XVYZXYXVZVYZ, *S*3 = XVXYXZVZXYVX

Step 4: Repeat Step 3, until the the generation-best result can't be improved by artificial ants in 100 consecutive generations.

2.5 Genetic Algorithm for *k*-LCS

Genetic Algorithm to solve k-LCS was developed by Chiang et al. [2]. As the part of the algorithm we choose the smallest sequence from set S that contains k sequences. The smallest one is our template sequence. In different words we can say a template pattern of binary string i.e 0 or 1 is randomly generated and if we write all the letters corresponding to 1 in template sequence, the final result is the template sub-sequence.For a template sequence *t* = *BAABCDGFHTBC* and a template pattern *tp* = 010110111011. Then the generated template sub-sequence *ts* =*ABCGFHBC*. For k sequences in set S we will randomly generate *tp* template patterns as our initial populations set. Then we will apply crossover on randomly choosing 2 different template patterns. Then again we will choose a template pattern randomly and mutation operation is applied by choosing a mutation point *j* and toggling the bit at that point from 0 to 1 or 1 to 0. Finally we use fitness function to determine whether the template pattern is good or not.

We use fitness function given in 2.2 where $|S|$ denotes the input sequences in set S, P_j^m is the occurrences of pattern P_j in the sub-sequences *S*, P_j^v is the sum of letters that P_j is matched to all sequences in *S*, and $f(P_j)$ gives the fitness function of the pattern P_j.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

$$f(P_j) = \begin{cases} P_j^m \times P_j^v & \text{if } P_j^m = |S| \\ -1 \times (|S| - P_j^m) \times P_j^v & \text{otherwise.} \end{cases} \quad (2.2)[2]$$

Genetic Algorithm

```
Output: The CS of S
{Step 1. Initialize population g}
 Produce p template patterns P , choose the last sequence  as the template sequence $S_{last}$
{Step 2. Reproduce the template sub-sequences $Sub_i$}
  for i = 0 to p do
 for j = 0 to n do
if $P_i[j] = 1$ then
$Sub_i \leftarrow S_{last}[j]$
end if
end for
 end for
{Step 3. Compare template sub-sequences with input sequences}
The Fitness function($P_j$)
{Step 4. Reproduce new $P_j$}
 $Parent_1 \leftarrow random(P)$; $Parent_2 \leftarrow random(P)$
Crossover($Parent_1$ , $Parent_2$)
$Parent_3 \leftarrow random(P)$
Mutation($Parent_3$)
{Step 5.After repeat Step 2 to Step 4, return the CS}  Termination condition: G generations are reached or $f(P_{highest})$ is not changed in 10 consecutive generations }.  CS $\leftarrow S_1$
```

III. COMPARISON TABLE.

Table 1. Comparison table for different algorithms on k-LCS

	DP	EA	BNMAS	ACO	GA
Population	Single	Single	Single	Single	Multiple
Iterative	No	No	No	Yes	Yes
Fitness Function	No	No	No	Yes	Yes
Crossover	No	No	No	No	Yes
Mutation	No	No	No	No	Yes
No. Of Parameters	Less	Less	Less	More	More
Accuracy	Good	Average	Average	Good	Best

VI. PROPOSED ALGORITHM

The Genetic Algorithm comes out to be efficient in terms of the sequence length and accuracy when compared to other heuristics explained above. We hybridize the Genetic Algorithm by combining Hill Climbing in the selection of next chromosome. This will add local optimal search to the GS. The mutation and crossover operation ensures that the search is spread throughout the population space and hill climbing will ensure that we reach the local optima resulting in better quality solution than the GA. By this we extract the goodness of Hill Climbing as well as Genetic Algorithm thus helping in achieving better CS with comparatively less execution time.

VI. HYBRID GENETIC ALGORITHM

```
Output: The CS of $S$
{Step 1. Initialize population $g$}
 Produce $p$ template patterns $P$ , choose the last sequence  as the template sequence $S_{last}$
{Step 2. Reproduce the template sub-sequences $Sub_i$}
for i = 0 to p do
for j = 0 to n do
if $P_i[j] = 1$ then
$Sub_i \leftarrow S_{last}[j]$
end if
end for
end for
{Step 3. Compare template sub-sequences with input sequences}
The Fitness function($P_j$)
{Step 4. Reproduce new $P_j$}
$Parent_1 \leftarrow random(P)$; $Parent_2 \leftarrow random(P)$
Crossover($Parent_1$ , $Parent_2$)
$Parent_1$=Hill_Climbing($Parent_1$)
$Parent_2$=Hill_Climbing($Parent_2$)
$Parent_3 \leftarrow random(P)$
Mutation($Parent_3$)
{Step 5.After repeat Step 2 to Step 4, return the CS}  Termination condition: $G$ generations are reached or
$f(P_{highest})$ is not changed in 10 consecutive generations }.  CS $\leftarrow S_1$
```

V. CONCLUSION

The longest common sub-sequence problem is a NP-hard problem when the sequence is large and k is not fixed. We have reviewed various algorithms for solving the LCS problem and finally came out with the hybrid approach of our own. We are positive to obtain a comparatively larger common sub-sequence than any other algorithm reviewed in this paper by extracting the best from Genetic Algorithm and Hill Climbing. Their

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

hybridization will yield in better solution quality and less execution time. The local search optimization using Hill Climbing gets the best out of Genetic Algorithm when used together. In similar manner other heuristics as well as meta-heuristic algorithms can also be hybridized with Genetic Algorithm to achieve significantly improved results.

VI. FUTURE SCOPE

For the future work, the algorithm will be tested on large k having a significantly long length and to optimize the performance of the algorithm in terms of solution length as well as the execution time. Hybridization with many other heuristics and meta-heuristics will help achieve this. Furthermore implementation of the algorithm using Cuckoo Search and Firefly Algorithm will help compare the solution generation process and that will aid in even better hybridization of the algorithms to get good quality solutions for our problem as well as various other such problems.

REFERENCES

1. Bonizzoni, P., Vedova, G.D. and Mauri, G.: Experimenting an approximation algorithm for the LCS In: Discrete Applied Mathematics,Vol. 110, No. 1, pp. 13–24 (2001).
2. Dorigo, M., Maniezzo, V. and Colorni, A.: Ant system- Optimization by a colony of cooperating agents. In: IEEE Transactions on Systems, Man, and Cybernetics-Part B,Vol. 26, No. 1, pp. 29–41 (1996).
3. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequence. In: Communications of the ACM,Vol. 18, No. 6, pp. 341–343, (1975).
4. Hirschberg, D. S.: Algorithms for the longest common subsequence problem. In: Journal of ACM, Vol. 24, pp. 664–675 (1977).
5 . Huang, K.-F, Yang, C.-B. and Tseng, K.-T: An efficient algorithm for multiple sequence Alignment. In: Proc. of the 19th Workshop on Combinatorial Mathematics and Computation Theory, Kaohsiung, Taiwan, pp. 50–59 (2002).
6. Huang, K.-S., Yang, C.-B. and Tseng, K.-T: Fast algorithms for finding the common subsequence of multiple sequences. In: Proceedings of International Computer Symposium, Taipei, Taiwan, pp. 90–95 (2004).
7. Hunt, J.W. and Szymanski, T.G.: A fast algorithm for computing longest common subsequences. In: Communications of the ACM,Vol. 20, No. 5, pp. 350–353 (1977).
8. Lee, R.C.T., Chang, R.C., Tseng, S.S., and Tsai, Y.T.: Introduction to the Design and Analysis of Algorithm - a strategic approach.ISBN 007-124346-1. In: McGraw Hill (2005)
9. Maier, D.: The complexity of some problems on subsequences and supersequences. In: Journal of the ACM, Vol. 25, No. 2, pp. 322–336 (1978).
10. Shyu, S.-J., and Tsai, C.-Y.: Finding the longest common subsequence for multiple biological sequences by ant colony optimization. In: Computers & Operations Research, Vol. 36, pp. 73–91, (2009).
11. Smith, R.E., Dike, B.A., and Stegmann, S.A.: Fitness inheritance in genetic algorithms. In: Proceedings of the 1995 ACM symposium on Applied computing, Nashville, TN, USA, pp. 345–350 (1995).

12. Wagner, R.A., and Fischer, M.J.: The stringto-string correction problem. In: Journal of the ACM, Vol. 21, No. 1, pp. 168–173 (1974).

13. Tsai, Y.T., and Hsu, J.T.: An approximation algorithm for multiple longest common subsequence problems. In: Proceeding of the 6thWorld Multiconference on Systemics, Cybernetics and Informatics, SCI, pp. 456-460 (2002).

14. Kuo-Si Huang, C.-B.Y., and Tseng, K.-T.: Fast algorithms for Finding the common subsequence of multiple sequences. In: Proc. of International Computer Symposium, Taipei, Taiwan, Dec. pp.15-17 (2004)

15. Julstrom, B.A., and Hinkemeyer, B.: Starting from scratch: Growing longest common subsequences with evolution In: Proceedings of the 9th In-ternational Conference on Parallel Problem Solving From Nature (PPSN IX), Lecture Notes in Computer Science 4193, Springer Berlin / Hei-delberg, pp. 930-938 (2006.)

16. Needleman, D.B., and Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. In: Journal of Molecular Biology, Vol. 48, No. 3, pp. 443-453 (1970).