RESEARCHERID
THOMSON REUTERS

[Rawat* *et al.*, 6(10): October, 2017]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

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

Aman Rawat, Pradeep Gehlot,Vishal Thakuria And Deepak Garg.
* Department of Computer Applications, NIT Kurukshetra, India

DOI: 10.5281/zenodo.1012555

ABSTRACT

Finding the longest common subsequence for multiple sequences are knows as k-LCS. For significant number and longer length sequences the k-LCS turns out to be a NP-hard problem. To solve this problem, we have developed a hybrid genetic algorithm that can be used to find an optimal solution in comparatively less time. The results obtained by our solution are comparatively better than that of EA (Expansion-algorithm), BNMAS(Best next maximal available symbol algorithm.), GA(Genetic algorithm) and ACO(Ant colony optimization algorithm). Using Hill Climbing along with Genetic Algorithm proved to be a successful development in finding the more optimal solution with less time complexity. As Genetic Algorithm has been used, the number or length of the sequences doesn't post to be a problem and a better subsequence is obtained with each iteration.

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

I. INTRODUCTION

We have as set S of sequences such that S= {s1, s1,….,s_k}, among these k-sequences we need to find the longest common subsequence. If subsequence X is present in all the sequences s_i ($1 \leq i \leq k$) then X is a CS (Common Subsequence) for S. E.g. lets consider a set S ={s1 = GTACTGA, s2 = ATCTGCA, s3 = CTTAGTA}.The sequences GA, ATA and TTGA are common sub-sequences in the strings s1, s2 and s3. TTGA is the longest one in all three of the common sub-sequences, so TTGA is the LCS of S. Longest Common subsequence is a popular problem in Computer Science [4, 5, 8, 13]. When no. of the sequences k is greater than 2 the k-LCS is NP-hard even over the binary input [10].

When k is significantly large then using the exhaustive search to search all the Common Subsequence in k sequences becomes challenging. Using DP (Dynamic programming) based algorithm needs $O(n^2)$ time and space in solving 2-LCS[17]. Many hheuristic algorithms has also been given to solve k-LCS problem..Bonizzoni et al. [1] proposed the EA (Expansion Algorithm). However, it wasn't satisfactory in terms of performance. BNMAS (best next for maximal available symbols) algorithm [7] is given by Huang et al. That is based upon the frequency of occurrence of common characters in the input sequence. Several evolutionary algorithms have also been proposed to ease the searching limitations. ACO (Ant colony optimization)algorithm for the k-LCS IS given by Shyu-and-Tsai [11]. The characteristics of ant searching for food are used to find the common sub-sequence in S [3].

Genetic Algorithm (GA) uses the crossover, mutation and selection which are the biologically-inspired process to solve the k-LCS optimization problem. An algorithm was given by Chiang et al. for solving the k-LCS problem [2]. The Common subsequence can be regarded as chromosomes and are evolved in further subsequent generations using mutation and crossover process to find the improved common sub-sequence. Fitness function is used as an evaluation mechanism based on it the better solutions from the chromosomes are preserved to create the new generation and then its utilized for evolution in the next generations [12].

In our hybrid Genetic Algorithm we are optimizing the process of creation of initial population as well as using hill climbing as a optimizing technique in the process of crossover and mutation. This will help us create a faster and more optimal algorithm to find the solution to k-LCS.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

II. GENETIC ALGORITHM

Chiang et al. [2] developed Genetic Algorithm to solve the problem of k-LCS. In this algorithm the sequence with the smallest length is chosen from the set containing the k sequences. We choose the smallest one as our template sequence. In another words we randomly generate a template pattern of 1 or 0 is and by writing all the characters corresponding to 1 in template sequence, the get the final result that is the template subsequence. For a template sequence t = *GATCTCGAGCAT* and a template pattern tp = 101000011001, the generated template sub-sequence till be ts =*GTAGT*.

The initial population set can be generated by taking k template patterns tp for k sequences in set S. All these k template patterns tp are then evaluated against the fitness function. The template pattern having the best fitness value i.e the highest non-negative value is chosen as the best solution for that generation, in this case for the 1st generation. The template patterns are then applied with mutation and crossover to generate the new template pattern to be evaluated at the subsequent generations. The crossover operation are applied in pairs and a point is chosen and the template pattern are swapped starting at the chosen point and ending at the length of the string. After this remaining template patterns are chosen randomly and mutation is applied to them by changing the 0 to 1 and 1 to 0 in the template patterns. In our algorithm we are taking 80% template patterns for crossover and the remaining 20% for mutation.

Figure 2.1 below gives the fitness function for our algorithm. $|S|$ indicates the number of input sequences in the set S, P_j^m is number of times the pattern P_j occurred in the sub-sequences of S, P_j^v is the total number of times that P_j is successfully matched to all sequences in S, and finally $f(P_j)$ will give the value of fitness function corresponding to the pattern P_j in sequence set S.

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

Fig (2.1)

As an improvement, in our hybrid genetic algorithm we are using hill climbing in the process of mutation and crossover to produce good quality template patterns for next generations. For every template pattern pair we are choosing a distinct crossover points that are randomly generated. In case of mutation again different mutation points have been chosen for better template pattern generation. Using the Genetic algorithm ensures that the search for the solution is spread throughout the sample space and Hill Climbing ensures finding local optima. By combining these we extract the best out of both the algorithms and derive the good quality solution.

Hybrid Genetic Algorithm

```
Output: The CS of S
{Step 1. Initialize population g}
 Produced p template patterns P , choose the last sequence  as the template
  sequence S_last
{Step 2. Reproduced 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 ← 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. Reproduced new P_j}
   Parent_1 ← random(P ); Parent_2 ← random(P )
   Crossover(Parent_1 , Parent_2)
                Parent_1=Hill_Climbing(Parent_1)
```

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

$Parent_2$=Hill_Climbing($Parent_2$)
$Parent_3$←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 ← S_1

Hybrid Genetic Algorithm MATLAB Code

```
tic;
s1=fileread('s1.txt');
s2=fileread('s2.txt');
s3=fileread('s3.txt');
s4=fileread('s4.txt');
s5=fileread('s5.txt');
s6=fileread('s6.txt');
loo=0;
totalinput=6;
s=[s1; s2; s3; s4; s5; s6];
n=length(s1);
for itr=1:1
    count=0;
    ntp=10000
    phighest=0;
    maxlen=power(2,20);
    ra=randperm(maxlen);
    ra=ra(1:ntp/2);
    p1 = de2bi(ra,n);
    largesttp=repmat(char(0),1,n);
    p2= randi([0 1], ntp/2,n);
    p=cat(1,p1,p2)
      for xxxx=1:100
                sub = repmat(char(0),ntp,n);
    p;
    for i=1:
          ntp
          cnt=1;
    for j=1:
          n
          if p(i,j)==1
                   m=s1(j);
    sub(i,cnt)=strcat(sub(i,cnt),m);
    cnt=cnt+1;
    end
    end
    end
    pv= zeros(1,ntp);
    pm=zeros(1,ntp);
    fp=zeros(1,ntp);
    for i=1:
          ntp
          v=sub(i,:);
    len=0;
    len=length(deblank(v));
    for j=1:
          totalinput
          x=s(j,:);
```

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

```
cnt=1;
for k=1:
      n
      if x(k)==v(cnt)
               cnt=cnt+1;
end
if (cnt-1)==len
    pm(i)=pm(i)+1;
end
if cnt>len
break;
end
end
pv(i)=pv(i)+cnt-1;
end
end
for i=1:
      ntp
    if pm(i)==totalinput
        fp(i)=pm(i)*pv(i);
    else
        fp(i)=(-1)*(totalinput-pm(i))*pv(i);
end
end
max=fp(1);
maxtp=repmat(char(0),1,n);
for i=1:
      ntp
      if fp(i)>0
               v=sub(i,:);
end
if fp(i)>=max
    max=fp(i);
    maxtp=sub(i,:);
end
end
if max>phighest
   phighest=max;
   largesttp=maxtp;
end
count=count+1
   largesttp
   LengthOfOutput=length(deblank(largesttp))
   r=randperm(ntp);
ntpp=(ntp*4)/5 ;
mt=ntp-ntpp;
r=r(1:ntpp+mt);
if rem(ntpp,2)==0
   ntppp=ntpp/2;
else
   ntppp=(ntpp-1)/2;
end

for i=1:
    ntppp
    d=randi([1 n],1,1);
d1=randi([1 n],1,1);
```

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

```
    if d>d1
    tmp=d;
    d=d1;
    d1=tmp;
    end

    for j=d:
          d1
          tmp=p(r(i),j);
    p(r(i),j)=p(r(ntppp+i),j);
    p(r(ntppp+i),j)=tmp;
    end
    end
    nn=10;
    for i=ntpp+1:
          ntpp+mt
          r1=randperm(n);
    r1=r1(1:nn);
    for j=1:
          nn
          p(r(i),r1(j))=1- p(r(i),r1(j));
    end
    end
    end
largesttp
end
toc;
```

III. RESULTS AND DISCUSSION

Tables:

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
Time Complexity	$O(N^2)$	$O(kn^3 \log n)$	$O(\sigma^2 kn + \sigma^3 n)$	-	$O(Gpk(n+\lvert P_j \rvert))$

Where *k = denotes Number of String.
*n = Length of Strings.
* $\sigma = |\Sigma|$
*G = Number of Generation

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

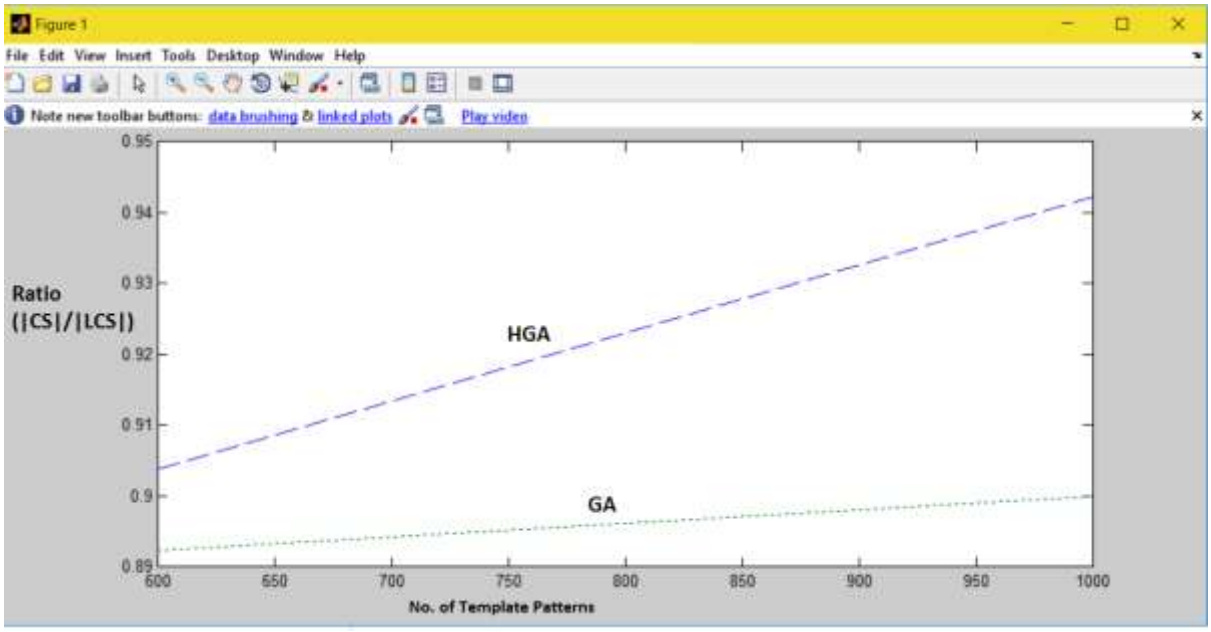
As we can clearly see from the above table Genetic Algorithm provides the best time complexity when compared to other algorithms. Now we will be comparing the accuracy of our Hybrid Genetic Algorithm with respect to Genetic Algorithm.

Table 2: Comparing the accuracy of HGA and GA

Sequence Length	Template patterns	GA(|CS|/|LCS|)	HGA(|CS|/|LCS|)
100	600	.8923	.9038
	1000	.9000	.9423
500	600	.5250	.9269
	1000	.5610	.9403
1000	600	.5490	.9182
	1000	.5817	.9211

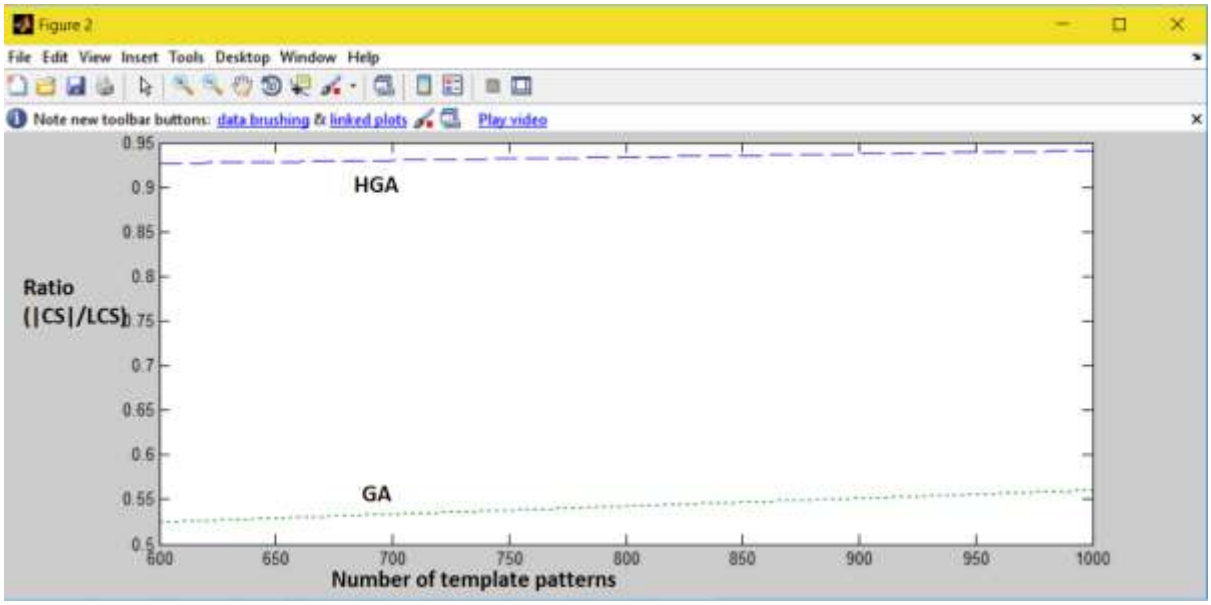
The table 2 shows comparison in results of HGA and GA for string of length 100, 500 and 1000. In input is a DNA sequence that has been randomly generated. Initial population is generated to be 600 and 1000 in all three cases. |CS| denotes the length of common subsequence generated by GA and HGA and |LCS| denotes the actual LCS for the input sequences. The ratio (|CS|/|LCS|) will give the accuracy of the result obtained(highest being 1). The k is taken to be 6 i.e there are 6 sequences. We can clearly see that HGA produces better results than GA in all the cases and in some cases the result is significantly better than the GA.

The graphical results are as follows:

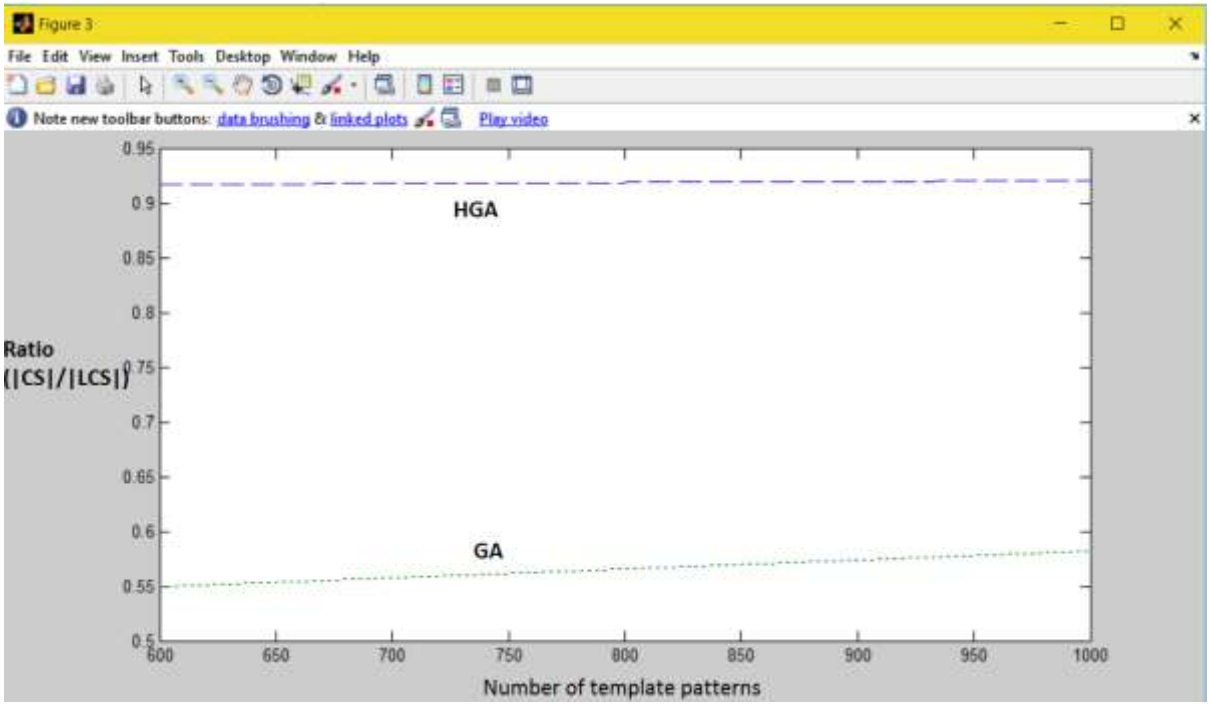


Graph 1: Sequence length is 100

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7



Graph 2: Sequence length is 500



Graph 3: Sequence length is 1000

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IV. CONCLUSION

The Hybrid Genetic Algorithm (GA + Hill Climbing) has turned out to be better in case of better quality subsequence generation when compared to the Genetic Algorithm. GA ensures that solution is optimized over the whole solution space and hill climbing helps in optimizing the local optima. The Bio-inspired process of GA ensures that we get better quality solutions at each iteration. After practically implementing the code we found the results were far more optimal than the GA. In contrast to other Algorithms GA produces results at very early stage though that might not be the optimal solution and it keeps on improving the solutions at subsequent iterations or generations. This comes out to be useful in cases where we do not want a 100% match but significantly low percentage of match is required, which in case of other algorithms require 100% execution of the program and can turn out to be very time consuming.

V. 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 co-operating 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).
17. Chung-Han Chiang: A genetic Algorithm for the Longest Common Subsequences of Multiple Sequences, January (2009).