Longest Common Subsequence using Chemical Reaction Optimization

Md. Rafiqul Islam
Computer Science and Engineering Discipline, Khulna University, Khulna - 9208, Bangladesh.
e-mail: dmri1978@gmail.com

Zarrin Tasnim Asha
Computer Science and Engineering Discipline, Khulna University, Khulna - 9208, Bangladesh.
e-mail: zarrin.ku@gmail.com

Rezoana Ahmed
Computer Science and Engineering Discipline, Khulna University, Khulna -9208, Bangladesh.
e-mail: sheme.cse@gmail.com

Abstract— **Longest Common Subsequence (LCS) is a well-known optimization problem. It is the problem to find out a longest subsequence of each member of a given set of sequences. It is an NP-hard problem and has applications in data compression, FPGA circuit minimization and bioinformatics. Chemical Raction Optimization (CRO) is a new meta-heuristic method that is being widely used in solving optimization problem. In this paper we have proposed an efficient Chemical Reaction Optimization technique to solve Longest Common Subsequence problem. The design strategies of elementary operators and correction method are shown to solve the problem. The proposed method is compared with two other methods and the experimental results show that it takes less execution time than that of others.**

Keywords—Longest common subsequence, Chemical Reaction Optimizatio, Metaheurictic, Molecule.

I. INTRODUCTION

Longest common subsequence is an optimization problem. Longest Common Subsequence (LCS) means the longest sequence of characters that appear left-to-right in a given set of strings. But it is not necessarily that it is adjacent block of characters. LCS is essential to occupy serial positions within the original sequences or strings. The longest common subsequence is the base of data comparison programs. Many problems or applications are solved by LCS such as the applications in data compression, FPGA circuit minimization, Bioinformatics [1]. A detail of literature review related to LCS problem and its solutions can be found in [1].One of the biological applications is to compare the DNA of two or more difference organisms. DNA is sequence which consists of A (Aclenine), G (Gaunine), C (Cytosine) and T (Thymin). Thus we are given two DNA sequences and wish to find DNA sequence which is common and longest in length and such a problem is LCS problem [2]. LCS problem for two strings can be solved using Dynamic Programming (DP) [2, 3], which gives optimum results. Chemical Reaction Optimization (CRO) is a simple and powerful meta-heuristic which is showing excellent performance in solving optimization problem. CRO mimics the interactions of molecules in chemical reactions to search for the global optimum [4]. CRO was designed as an optimization framework. It has been applied to solve many practical problems, e.g. quadratic assignment problem[5], population transition problem in peer-to-peer streaming[7], network coding optimization problem[8], standard continuous benchmark function[9], cognitive radio spectrum allocation problem[10], grid scheduling problem[11][12], stock portfolio selection problem [13],artificial neural network training [14], 0-1 Knapsack problem [3] etc. In this paper we have designed an algorithm to find LCS based on the concept of CRO. The proposed method and other two dynamic programming algorithms such as dynamic programming algorithm for LCS given in [2] and fast dynamic programming method depicted in [3] have been implemented and compared the results to show the performances.

II. LONGEST COMMON SUBSEQUENCE (LCS)

Common subsequence of two given sequences is a subsequence that exits in both the sequences. However, the Longest Common Subsequence (LCS) of two given sequences is a subsequence that exits in both the sequences and have longest or maximum length. For example, we have two sequences S_1=(E,F,G,H,E,F) and S_2=(F,H,G,E,F,E), the sequence S=(F,G,F,E) is an LCS. Since the subsequence is common in both in the sequence S1 and S2 and the sub sequence S have the length, |S| = 4, which is the maximum in common sub-sequences. The problem can be formulated as follows:
Let $X = [x_1, x_2 \dots x_n]$ be a sequence of n elements where X is a string. The elements of the string are members of a finite alphabet $\sum$, $x_n \in \sum$, $\forall i = 1, 2, 3, \dots, n$.
A sequence $Y = [y_1, y_2, \dots, y_m]$ is a subsequence of X if there exists an increasing sequence of indices $[i_1, i_2, \dots, i_k]$ such that $X[i_k] = Y[j]$ holds, $\forall j = 1, 2, \dots, k \le n$.

Given a finite set S of n strings $S^* = \{S_1, S_2, \ldots, S_n\}$, S is a common subsequence, if S subsequence S_i, $\forall i = 1, 2, 3, \ldots, n$.
The longest common subsequence of S* is the common subsequence of maximum length. The problem can be expressed as

Maximize |S|
subject to S subsequence S_i, $\forall i = 1, \ldots, n$.

with |S| being the length of the common subsequence S. The longest common subsequences are not always unique. More than one common subsequence with maximum length can be found there.

III. Chemical Reaction Optimization (CRO)

CRO is one of the newest optimization algorithms, which is inspired by the chemical reaction process [15]. CRO has been successfully applied to many other problems. In CRO, molecule (M) is the basic operating agent which represents the solution of optimization problem. To explore solution a set of molecules are controlled and manipulated by CRO. A molecule has potential energy (PE), kinetic energy (KE), hits number, and other characteristics that represent solution. CRO simulates four types of chemical reactions. The four types of elementary reactions are on-wall ineffective collision, decomposition, inter-molecular ineffective collision and synthesis. During the process of CRO the total energy remains constant which means CRO requires conservation of energy. The CRO algorithm includes three stages: initialization, iteration and the final stage. The initialization stage generates initial population (pop) along with PopSize, KElossRate, MoleColl, buffer, InitialKE and two thresholds(α and β). In iteration stage, one elementary reaction out of four reactions takes place in each iteration. Here, we have to determine whether uni-molecular or bi-molecular reaction is taken place. The type of reaction is determined by comparing a random number t [0, 1] against MoleColl. If t > MoleColl there will be unimolecular reaction. Otherwise a bi-molecular reaction occurs. At the end of the each iteration we have to check stopping criteria. Here the potential energy value of the newly obtained molecule(s) is compared with the original molecules. The new molecules will be accepted when the new values can satisfy the energy conservation conditions. Otherwise, the new molecules are discarded**.**

The pseudo code of CRO algorithm is depicted in Algorithm 1.

Algorithm 1: CRO Algorithm

```
Input: Objective function f and the parameter values
\\ Initialization
Set PopSize, KELossRate, MoleColl, buffer, InitialKE, α, and β
Create PopSize number of molecules or solutions
\\ Iterations
while the stopping criteria not met do
Generate b ∈ [0, |S1|]
if b > MoleColl then
Randomly select one molecule or solution S'
if Decomposition criterion met then
Trigger Decompos ()
else
Trigger Onwall ()
end if
else
Randomly select two molecules or solutions $S'_1$ and $S'_2$
if Synthesis criterion met then
Trigger Synthesis ()
else
Trigger Intermole ()
end if
end if
Check for any new minimum solution
end while
\\ The final stage
Output the best solution found and its objective function value
```

IV. Design CRO for LCS

Here we design CRO method for two strings. In LCS problem two sequences are given for finding the longest subsequence. Suppose we have two sequences such as S_1 (first sequence) of length n and S_2 (second sequence) of length m, where $m \leq n$ and $n = |S_1|$ and $m = |S_2|$. We have to find out the longest common subsequence of S_1 and S_2.
Example 1:
S_1 = "GCCCTAGCG".
S_2 = "GGCACTA".
The output sequence, S = "GCAC".
Here we will design Chemical Reaction Optimization (CRO) algorithm for LCS problem.

S_1 :

Character	G	C	C	C	T	A	G	C	G
Index	0	1	2	3	4	5	6	7	8

S_2:

Character	G	G	C	A	C	T	A
Index	0	1	2	3	4	5	6

A. Solution Generation

To generate solution first we take an array of length m=$|S_2|$. Then we generate m random numbers from 0 to n =$|S_1|$. Next we sort the numbers to get a solution sequence. The following Fig. 1 depicts the scenario according to the example 1.

Generate $|S_2|$ random numbers $k \in [0, n]$:

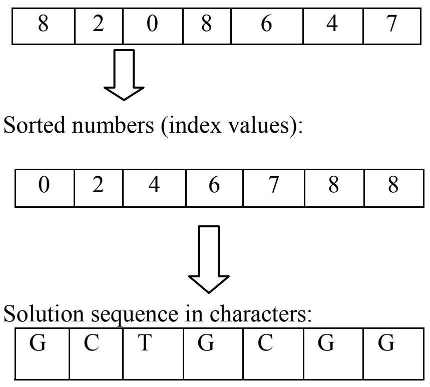
8 2 0 8 6 4 7
Sorted numbers (index values):
0 2 4 6 7 8 8
Solution sequence in characters:
G C T G C G G

Fig. 1. Solution Representation

B. *On-wall Ineffective Collision*

This elementary reaction corresponds to the basic local search operator for CRO. It changes the solution structure of one solution. Suppose that in an on-wall ineffective collision, the solution structure S is transformed to S'. The new solution structure is generated as S' = C(S), where C(S) is a correction operator that corrects new solution structure. The detail of C (S) function is explained at the end of this section.

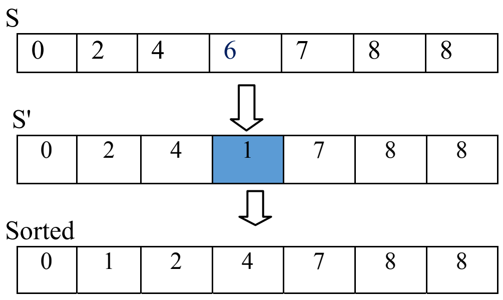

Fig. 2. Solution after On Wall Ineffective Collision

C. *Decomposition*

This elementary reaction generates two solutions from one solution. Decomposition often applies a vigorous change to the solution and the resultant solution possesses solution structures greatly different from the original one. Assume that the operator creates two solutions S'_1 and S'_2 from solution S, such as,

$$S \rightarrow S'_1 \cup S'_2$$

Firstly, S is copied to generate S'_1 and S'_2. After that, for n/2 positions in solutions S'_1 and S'_2 are changed randomly. Then we sort them. The function C (S) is invoked to make the output solution valid.

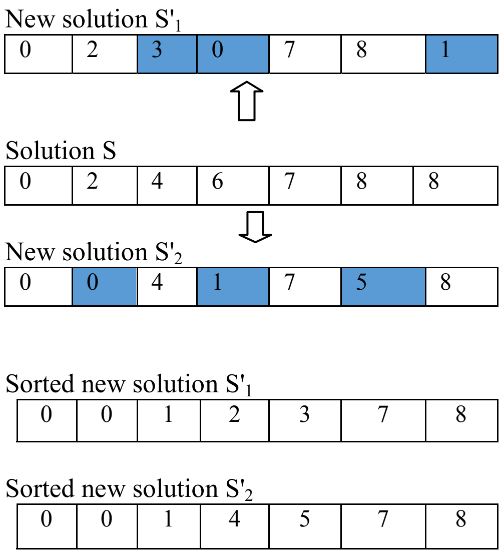

Fig. 3. Solution representation after decomposition

D. *Inter-molecular Ineffective Collision*

It changes the solution structure. The molecularity (assume two) remains unchanged before and after the process. Suppose that in an inter-molecular ineffective collision, the solution structure S_1 is transformed to S'_1 and S_2 is transformed to S'_2. It can be defined as follows:

$$S_1 \cup S_2 \rightarrow S'_1 \cup S'_2$$

Then the new solution structure is generated as S' = C (S).

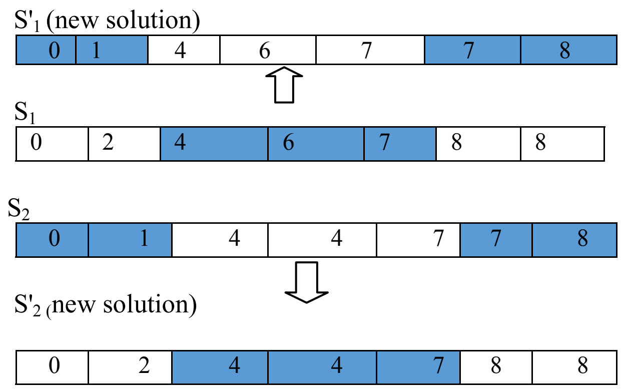

Fig. 4. Solution representation in Inter-molecular Ineffective Collision

E. *Synthesis Operator*

Synthesis takes two molecules as the inputs and combines them to generate a new molecule. In this paper we take two existing solution S'_1 and S'_2. Generate a new solution S from them. It can be represent by the following form.

$$S'_1 \cup S'_2 \rightarrow S$$

This new solution S is the longest common subsequence.

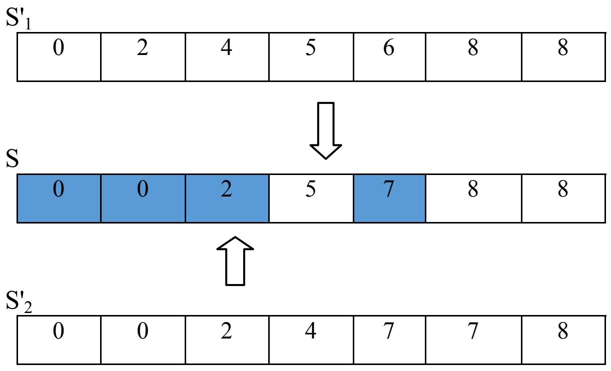

Fig. 5. Solution representation after Synthesis

F. *Correction Method*

In correction method, the input is solution, S. From the solution vector we get index of character in S_1. Next, we compare the character in position S[i] of S_1 with the character in position, i of S_2, where i = 0, 1……….n-1. If we find same character (character matching) in S_2 and in S_1, then we copy the value of S[i] to output vector, S'[i]. If different character is found (no character matching) then we do forward searching in S_1 starting at the position S[i] +1 to the position S[i+1]-1. If any character in position, j where j ∈ [S[i]+1 to S[i+1]-1] matches with the character in position, i of S_2, the value of j (which is the index value of S_1)is saved to S'[i]. In Fig 6, when i = 1, the character in position 1 of S_1 is C and the character in the same position of S_2 is G. So we do forward searching in S_1. Here S[i] +1 = 2 and S [i+1]-1 = 3, so we perform searching in positions 2 to 3 of S_1. In this case the character in position 3 of S_1 is G, which is same in position 1 of S_2. So we save 3 (position of S_1) to S'[1]. If character matching is not found, then backward searching is to be done in positions S[i]-1 to S [i-1] +1. In backward searching, if we find character matching in position, k of string1 where k ∈ [S [i-1] +1 to S[i]-1]. We save the value of k in S'[i]. In case, there is no character matching in forward searching and also in backward searching the value of S[i] to be copied in S'[i].

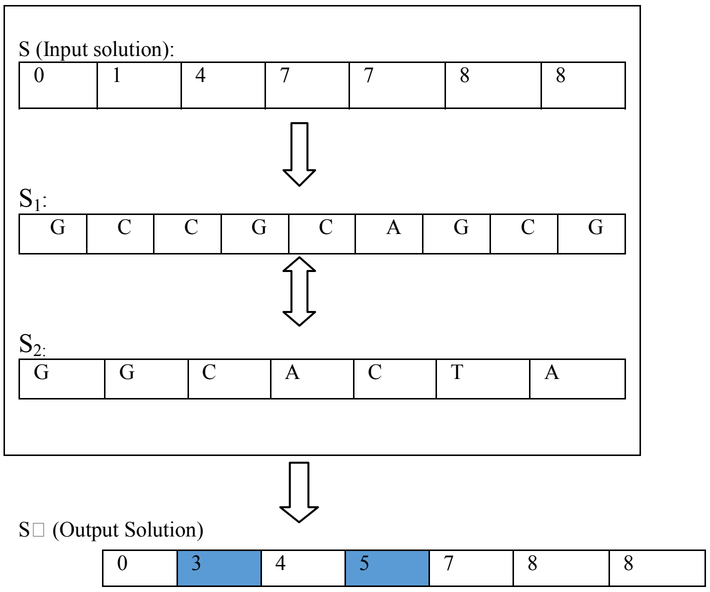

Fig. 6. Solution representation after Correction

V. Simulation Results

We compare proposed CRO method with Dynamic LCS [2] and Fast Dynamic LCS [3] algorithms to show their performances. During the experiments, we have used our own data sets in all test cases. For input string of different length generation different alphabets are used. All times the lengths of both input strings are same. As stated earlier the input sequences are generated randomly. In order to use of same input over and over, different input are used. The following alphabets: {0, 1}, {A, C, G, T} and English alphabet with 26 characters, i.e., {A….Z} or {a…z} are used in order to build input strings. Two strings of certain length are generated randomly based on certain alphabet and used those strings as an input for the algorithm. As we say in section 1(Introduction section) that DNA sequence is consists of A (Adenine), G (Guanine), C (Cytosine) and T (Thymine). For example, the DNA of any organism can be a string S=ACCTGT. So DNA can be expressed as a string that is consists of finite set {A, G, C, T}.

Two strings (S1 and S2) are generated as follows:

$$S_i = \text{rand}\ [A, G, C, T],\ i=1, 2, 3,……, n.$$

Here, rand [A, G, C, T] generates a character from {A, G, C, T} uniformly. The length of the string is n.

In CRO parameters affects its performance. For first testing we set the parameters as initialize PopSize = 10, CollisionRate = 0.2, buffer = 0, KE = 1000, lossRate = 0.1, DecompositionThreshold (α) = 15000 and SysthesisThreshold (β) =10 etc. The decomposition condition and synthesis condition respectively effect by the value of Decomposition Threshold and Systhesis Threshold. We perform 50 runs for each of the chosen values. Among the chosen values the sets of 50 runs are compared and the longest length's sequence is selected. During the experiment, we see that 50 runs for each of the chosen values is not sufficient for longest String which length is bigger than 1000 or more. Then we need to increase the number of runs such as, 100 or more. We showed the performance of the algorithm in the table.

The test environment is set up on a personal computer with Core i5 CPU at 2.50 GHz CPU, 4G RAM, running on Windows8. Each LCS algorithm was run several times in order to obtain accurate CPU time measurements. System.currentTimeMillis() method was used for obtained the time measurement. It returns current CPU time in milliseconds. All of the algorithms are developed using Java programming language and the process of coding, Eclipse IDE is used**.**

TABLE I. SIMULATION RESULTS OF CRO

Case 1: Length of string 1 =180, Length of string 2 = 28 , Length of output sequence = 28				
Number of iteration	***Best case***	***Worst case***	***Average***	***Time (ms)***
50	75	57	66	1.09
100	78	50	60	1.24
150	80	48	58	1.30

TABLE II. SIMULATION RESULTS OF CRO

Case 2: Length of string 1 =270, Length of string 2 = 56 , Length of output sequence = 53				
Number of iteration	***Best case***	***Worst case***	***Average***	***Time (ms)***
50	70	58	64	6.92
100	80	50	59	6.94
150	82	40	63	7.00

TABLE III. SIMULATION RESULTS OF CRO

Case 3: Length of string 1 =360, Length of string 2 = 84 , Length of output sequence = 78				
Number of iteration	***Best case***	***Worst case***	***Average***	***Time (ms)***
50	55	53	54	13.60
100	67	48	50	13.80
150	70	30	60	14.00

TABLE IV. SIMULATION RESULTS OF CRO AND OTHER ALGORITHMS

Algorithm	***Length of string1***	***Length of string2***	***Length of output sequence***	***Time(ms)***
LCS	128	28	28	10
FLCS			28	7
CRO			28	**4.32**
LCS	270	56	53	18
FLCS			53	12
CRO			51	**6.96**
LCS	360	84	78	25
FLCS			78	21
CRO			78	**13.98**

VI. CONCLUSIONS

In this paper we have proposed an algorithm to solve longest common subsequence problem based on the concept of chemical reaction optimization. Our main target is to reduce the time complexity of LCS problem and generate an optimal solution. We have compared proposed CRO based method with Dynamic LCS [2] and Fast Dynamic LCS [3] and found that our proposed algorithm has superior performance when compared for all proposed test instances. It reduces the execution time. Here we have used two strings of different lengths. In the future, we will design a CRO algorithm to find LCS for multiple strings.

REFERENCES

[1] Tabataba, Farzaneh Sadat, and Sayyed Rasoul Mousavi. "A hyper-heuristic for the Longest Common Subsequence problem." *Computational biology and chemistry* 36 (2012): 42-54.

[2] Cormen, Thomas H. *Introduction to algorithms*. MIT press, 2009.

[3] Truong, Tung Khac, Kenli Li, and Yuming Xu. "Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem." *Applied Soft Computing*13.4 (2013): 1774-1780.

[4] Ossman, Mohammed, and Lamiaa Fathi Hussein. "Fast Longest Common Subsequences for Bioinformatics Dynamic Programming." *population* 5 (2012): 7.

[5] Xu, Jin, Albert YS Lam, and Victor OK Li. "Parallel chemical reaction optimization for the quadratic assignment problem." *World Congress in Computer Science, Computer Engineering, and Applied Computing*. 2010.

[6] Chaudhuri, Arindam. "A Dynamic Algorithm for the Longest Common Subsequence Problem using Ant Colony Optimization Technique." *arXiv preprint arXiv:1307.1905* (2013).

[7] Lam, Albert, Jialing Xu, and Victor OK Li. "Chemical reaction optimization for population transition in peer-to-peer live streaming." *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010.

[8] Pan, Bo, Albert Lam, and Victor OK Li. "Network coding optimization based on chemical reaction optimization." *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011.

[9] Lam, Albert, Victor OK Li, and James JQ Yu. "Real-coded chemical reaction optimization." *Evolutionary Computation, IEEE Transactions on* 16.3 (2012): 339-353.

[10] Lam, Albert, and Victor OK Li. "Chemical reaction optimization for cognitive radio spectrum allocation." *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010.

[11] Xu, Jin, Albert Lam, and Victor OK Li. "Chemical reaction optimization for the grid scheduling problem." *Communications (ICC), 2010 IEEE International Conference on*. IEEE, 2010.

[12] Xu, Jin, Albert Lam, and Victor OK Li. "Chemical reaction optimization for task scheduling in grid computing." *Parallel and Distributed Systems, IEEE Transactions on* 22.10 (2011): 1624-1631.

[13] Xu, Jin, Albert YS Lam, and Victor OK Li. "Stock portfolio selection using chemical reaction optimization." *Proceedings of the international conference on operations research and financial engineering. Paris, France.* 2011.

[14] Yu, James JQ, Albert Lam, and Victor OK Li. "Evolutionary artificial neural network based on chemical reaction optimization." *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011.

[15] Lam, Albert, and Victor OK Li. "Chemical-reaction-inspired metaheuristic for optimization." *Evolutionary Computation, IEEE Transactions on* 14.3 (2010): 381-399.