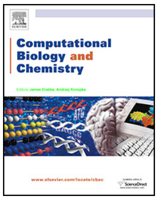
Computational Biology and Chemistry 64 (2016) 82–93

Contents lists available at ScienceDirect

Computational Biology and Chemistry

journal homepage: www.elsevier.com/locate/compbiolchem



Chemical reaction optimization for solving shortest common supersequence problem

C.M. Khaled Saifullah, Md. Rafiqul Islam*

Computer Science and Engineering Discipline, Khulna University, Khulna 9208, Bangladesh

ARTICLE INFO

Article history:
Received 8 March 2016
Received in revised form 13 April 2016
Accepted 19 May 2016
Available online 31 May 2016

Keywords:
Chemical reaction optimization
Shortest common supersequence
Meta-heuristics
NP-hard problem
Algorithm

ABSTRACT

Shortest common supersequence (SCS) is a classical NP-hard problem, where a string to be constructed that is the supersequence of a given string set. The SCS problem has an enormous application of data compression, query optimization in the database and different bioinformatics activities. Due to NP-hardness, the exact algorithms fail to compute SCS for larger instances. Many heuristics and meta-heuristics approaches were proposed to solve this problem. In this paper, we propose a meta-heuristics approach based on chemical reaction optimization, CRO_SCS that is designed inspired by the nature of the chemical reactions. For different optimization problems like 0-1 knapsack, quadratic assignment, global numeric optimization problems CRO algorithm shows very good performance. We have redesigned the reaction operators and a new reform function to solve the SCS problem. The outcomes of the proposed CRO_SCS algorithm are compared with those of the enhanced beam search (IBS_SCS), deposition and reduction (DR), ant colony optimization (ACO) and artificial bee colony (ABC) algorithms. The length of supersequence, execution time and standard deviation of all related algorithms show that CRO_SCS gives better results on the average than all other algorithms.

1. Introduction

The shortest common supersequence (SCS) problem states as for a given alphabet and a set of strings, the task is to find a string of minimum length that is the supersequence of every string. Supersequence of a given string means the sequence of characters of the supersequence is exactly same as that of the string (sequence need not to be adjacent).

Let $\sum$ be an alphabet. A string is a set of zero or more characters from $\sum$. Given a set, $S = \{s_1, s_2, \ldots, s_n\}$ of n strings, where $s_i \neq s_j$ if $i \neq j$. A string x is called a supersequence of each string s_i where s_i can be embedded in x and $s_i \in S$, and $i = 1, 2, \ldots, n$. It implies that s_i is a subsequence of x. For example, given $\sum = \{A, C, G, T\}$ and a set $S_1 = \{CAG, ACT, AAG\}$, a supersequence of all sequences in S is CACTAG. For a set of strings, there can be more than one supersequence. The shortest common supersequence (SCS) is a supersequence which is the shortest in length of all the supersequences. For the above example, the shortest common supersequence is ACTAG. So, the Shortest Common Supersequence (SCS) problem can be stated, we have a set S of n sequences (strings) over the alphabet $\sum$, as to get a sequence x of the shortest length such that each s_i is a subsequence of x. If $m = |\sum|$ and n is the number of strings, the complexity of the SCS problem is defined as $O(m^n)$. Let we have a set of strings $S_2 = \{CTG, CAG, CTA, AGA\}$ if W is the set of supersequences of each sequence in S_2 then $W = \{CTAGA, CATGA, CTGAGA, AGCATGA\}$. Now we have to find a sequence from W that is shortest in length. In this example the shortest common supersequence is CTAGA and the length of this supersequence is five. Fig. 1 shows a supersequence of four given sequences.

Now we create an objective function and constraints for the SCS problem. Let $S = \{s_1, s_2, \ldots, s_n\}$ is a set of n strings, $W = \{w_1, w_2, \ldots, w_k\}$ is a set of supersequences of each string in S and $L = \{l_1, l_2, \ldots, l_k\}$ is a set of lengths of the supersequences, where l_i is a positive integer and it represents the length of the supersequence w_i, where $w_i \in W$. According to the SCS problem we have to seek and find a supersequence w_m where l_m is the minimum in L. Thus the shortest common supersequence problem may be formulated as follows:

$$F(l) = \min\{l_m\} \tag{1}$$

Subject to $w_m \in W$, $s_m \in S$ and $w_m \succ s_m$ for $m = 1, 2, \ldots, k$. Where $\succ$ means supersequence, $F(l)$ is an objective function and it takes the minimal length in L.

The SCS problem has great application in data compression (Timkovskii, 1989), AI planning (Foulser et al., 1992), query

---

* Corresponding author.
E-mail address: dmri1978@yahoo.com (Md. Rafiqul Islam).

http://dx.doi.org/10.1016/j.compbiolchem.2016.05.004
1476-9271/

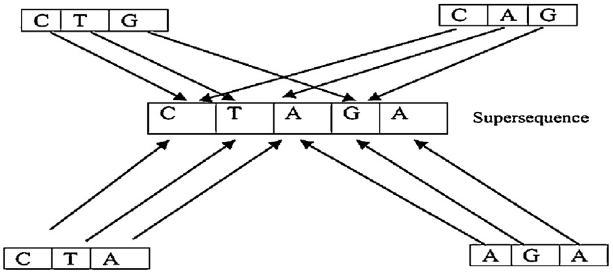


Fig. 1. An example of a common supersequence.

optimization in the database (Chaudhuri and Bruno, 2008), probe synthesis during microarray production (Rahmann, 2003), multiple sequence alignment (Sim and Park, 2003).

The SCS problem gets the attention of the researchers because of its resource utilization nature. It gives an output that has least resource with maximum profit. For any sort of representation of information, if the information is arranged with the help of the SCS then it takes less time to compute different operations (Mousavi et al., 2012). It gives the least resources to other optimization problems (Michel and Middendorf, 1999), and minimization causes cost effective, less time consuming in the industrial world (Mousavi et al., 2012).

Researchers proposed exact algorithms like dynamic programming (DP) (Jiang and Li, 1995) and branch-and-bound (Fraser and Irving, 1995) to solve the SCS problem. These algorithms perform optimally in polynomial time when the instances are small or restricted. However for large instances, DP takes lots of memory and branch-and-bound takes exponential time (Barone et al., 2001). Due to the impossibility of execution of large instances of exact algorithms, the researchers proposed approximation algorithms. The approximation algorithms may not give the optimal solution, but within polynomial time, it provides good or near optimal solutions. Among the approximation algorithms, heuristic approaches include Alphabet algorithm (Barone et al., 2001), reduce and expand (Barone et al., 2001), deposition and reduction (DR) (Ning and Leong, 2006), majority merge (Branke et al., 1998).

Other approaches are meta-heuristics algorithms such as ant system and ant colony optimization (Michel and Middendorf, 1999), artificial bee colony (Noaman and Jaradat, 2011), hybridization of the mimetic algorithm and beam search (Hybrid MA_BS) (Gallardo et al., 2007), probabilistic beam search (Blum et al., 2007), enhanced beam search (Mousavi et al., 2012), etc. The deposition and reduction outperforms the majority merge, the tournament and greedy, the reduce and expand algorithms, whereas the probabilistic beam search outperforms the majority merge, the weighted majority merge and the hybrid MA-BS algorithms. The enhanced beam search algorithm is the most recent approach that outperformed the DR, the probabilistic beam search and the hybrid MA-BS algorithms (Mousavi et al., 2012). Ant colony optimization algorithm uses majority merge concept for building SCS which is not efficient. Besides it takes lot of computational time (Ning and Leong, 2006). Objective function of artificial bee colony (ABC) algorithm derived in Noaman and Jaradat (2011) shows that it finds the SCS that has the most similarity of sequences with set of strings than all other candidate SCS. From the implementation we have seen that, for small instances it gives optimal or near optimal solutions. But for large instances, it may return a string which is not SCS at all. Both Deposition and Reduction (DR) and Reduce and Expand (RE) algorithms have approximation ratio of which is not appealing (Mousavi et al., 2012). Since DR uses Alphabet algorithm for building candidate SCS and Alphabet algorithm has approximation ratio of. So, DR has approximation ratio of. Besides, Enhanced Beam Search (IBS_SCS) gives very much deterministic property being a meta-heuristic approach. For an instance it gives same result irrespective of number of runs. Because of this deterministic property IBS_SCS lacks optimal property. It gives same near optimal results for all sorts of instances for all run. Besides, before iteration IBS_SCS algorithm uses dynamic programming method to calculate heuristic values. For small instances, it works well. But for the large instances, it takes much computational time.

In this paper, we have proposed an algorithm based on the nature of the chemical reaction, chemical reaction optimization (CRO) algorithm to solve the SCS problem. Every unstable molecule in this universe wants to reach a stable state with low energy by reacting either with another molecule or with the surface. The CRO algorithm mimics that molecular interaction during a chemical reaction. The CRO algorithm has both local search and global search properties. This properties help any sort of population to find global best result. The high flexibility of designing reaction operators, population generation causes CRO algorithm fit for any optimization problem irrespective of problem information or complexity. However, recent studies prove that it performs better than other meta-heuristics approaches in solving different types of optimization problems. The CRO with Greedy strategy (CROG) (Truong et al., 2013a) outperforms ant colony optimization, genetic algorithm and quantum-inspired evolutionary algorithm to solve 0-1 knapsack problem. The artificial CRO outperforms the genetic algorithm to solve multiple choice 0–1 knapsack problem (Truong et al., 2013b). The CRO algorithm outperforms simulated annealing and tabu search for quadratic assignment problem (Lam and Li, 2010). The parallel version of CRO shows a better result than singular CRO for the same problem (Xu et al., 2010).

Here we have designed CRO_SCS algorithm using the basic four reaction operators that search the global optimum point in the search space. Reaction operators are used to spread out the initial population throughout the solution space using both global and local searches. Besides a new reform function has been designed that checks the validation of new supersequence which makes the algorithm fit for the problem. Symbol reduction procedure in reform function helps the algorithm to return shortest result. And defining amount of repairs by threshold value in Reform function makes the algorithm time efficient approach. The results of CRO_SCS are compared with those of the deposition and reduction (DR), the enhanced beam search algorithm (IBS_SCS), the artificial bee colony (ABC) and the ant colony optimization (ACO) algorithms.

2. Related works

To find the optimal solution of the SCS problem different approaches were proposed. Since the problem is NP-hard, so a slight mislead in approach can give the worst result. The different approaches proposed by researchers for solving the shortest common supersequence problem are described below.

2.1. *Ant colony optimization*

Ant colony optimization (ACO) algorithm is proposed by Michel and Middendorf (1999). Here the probabilistic version of the majority merge algorithm is used to find the supersequence. Every ant chooses a symbol to add in the supersequence either by pseudo-random-proportional-action method with a threshold value or using a probability distribution function. The state vector is updated after adding the symbol in the supersequence. The state vector is a numerical array where the present state of every string remains. After generating the valid supersequences, the authors rank the quality of the supersequences based on the lengths.