Solving Shortest Common Supersequence Problem using Chemical Reaction Optimization

C M Khaled Saifullah
Computer Science and Engineering Discipline
Khulna University
Khulna, Bangladesh
khaledkucse@gmail.com

Md. Rafiqul Islam
Computer Science and Engineering Discipline
Khulna University
Khulna, Bangladesh
dmri1978@gmail.com

Abstract— **The Shortest Common Supersequence problem is an NP-hard optimization problem that has a vast use in real world problems. It is used in data compression and different bio-informatics analysis. Different types of approaches were used to solve this problem. Exact algorithms failed to compute for large instances whereas approximation algorithms lack optimality. In this paper, we propose a meta-heuristic approach named as Chemical Reaction Optimization Algorithm (CRO-SCS) to solve the Shortest Common Supersequence (SCS) Problem. The experimental results demonstrate that our proposed method takes less time to find SCS than dynamic programming and have better performance than other well-known approximation algorithms.**

Keywords— Shortest Common Supersequence; Chemical Reaction Optimization; Meta-heuristic; NP-hard.

I. INTRODUCTION

Shortest Common Supersequence (SCS) problem states as for a given set of strings the task is to find the supersequence of every string that is minimum in length. The term supersequence means the sequence of symbols of a string and sequence of symbols of the computed strings are same (sequence need not to be adjacent). SCS problem has diversified application in probe synthesis during microarray production [1], AI planning [2], query optimization in the database [3] and data compression [4]. Given a finite set of strings L where every string was constructed using an alphabet $\sum$. Now we have to search a string s of minimal length that is a supersequence of all strings in L from the set of supersequence S. Therefore,

$$\text{Objective function} = \min(s \succ L) \text{ where } s \in S \quad (1)$$

An example of Common sequence is depicted in *Fig. 1*. Here, the alphabet $\sum$ contains four symbols, {a, c, g, t}. And, set of strings L has three strings, {S_1= tatcg, S_2= ctagc and S_3= agtgc}. From these three strings, we can generate a common supersequence S = tactagctgc.

Supersequence	t	a	c	t	a	g	c	t	g	c
S_1={t, a, t, c, g}	•	•		•			•		•	
S_2={c, t, a, g, c}			•	•	•	•	•			
S_3={a, g, t, g, c}		•				•		•	•	•

Fig. 1. An example of common supersequence

From *Fig. 1* we can see that, all of the three strings are embedded in supersequnce S. Therefore S is a common supersequence for L. Similar to S, we can generate some other common supersequnce for L. Now, the common supersequence having the least length is the SCS for L.

The SCS problem has shown to be very hard under various formulations and restrictions [4]. Therefore no exact algorithms were able to compute for bigger instances. To find the optimal solution for SCS dynamic programming (DP) algorithm [4] and Branch-and-Bound algorithm [5] have been proposed. Both dynamic programming and branch-and-bound algorithms cannot solve SCS problem unless the number of strings is restricted. Some approximate approaches were implemented for finding solutions of SCS problem where exact algorithms failed to solve. The heuristic approach includes Majority Merge (MM) [6] a variant of MM named as Weighted Majority Merge (WMM) [7], Deposition and Reduction (DR) [8], Reduce and Expand (RE) [9]. MM algorithm shortens the strings that have small size rather than checking the longer ones [7]. But in practical, shorter strings are found embedded in long strings. Worst case approximation ration of DP and RE is $|\sum|$ which is not appealing [10]. Besides, some meta-heuristics approaches like Artificial Bee Colony Algorithm (ABC) [11], Probabilistic Beam Search (PBS) [12], Ant colony optimization (ACO) [13], Genetic Algorithm (GA) [14] and Enhanced Beam search algorithm (IBS_SCS) [10] have also been proposed. Objective function of ABC algorithm does not cover the constraints of SCS whereas, both GA and ACO follow the MM concept which is time consuming. PBS algorithm checks all the candidate

978-1-5090-1269-5/16/$31.00 ©2016 IEEE

solution which is too time consuming [10]. And IBS_SCS algorithm is very much deterministic and does not guaranty optimality though it gives much better result than all other approaches.

In this paper, we propose a meta-heuristic algorithm Chemical Reaction Optimization Algorithm (CRO-SCS) for solving the Shortest Common Supersequence problem.

Experimental results show that CRO-SCS computes SCS in less time than DP and outperforms WMM, RE, DR algorithms both in length of returned supersequence and execution time.

II. Related Work

To find the optimal solution in SCS problem different approaches had been proposed and implemented. These approaches for solving shortest common supersequence are described below.

A. Dynamic Programming

Dynamic Programming algorithm was proposed by V. Timkovskii [4] and the Branch-and-Bound algorithm was proposed by C. Fraser [5]. Dynamic programming algorithm is successful if the strings are small in number. But for a large number of strings DP needs large spaces and branch-and-bound takes a lot of time.

B. Weighted Majority Merge

Failure of implementation of SCS problem for larger instances by exact algorithms leads researchers to work with approximation algorithms. Out of different approximate approaches, one of the earliest heuristic algorithms for SCS problem was Majority Merge (MM). The basic idea of the algorithm was to build supersequence by adding most frequent symbol found at the front of all strings and removing the symbol from the respective strings. But MM missed a fact that the strings could have different lengths [7].It suggests that strings with shorter length can be removed earlier. But it is necessary to shorten the length of the long string rather than removing symbols from the short string. Based on this concept J. Branke et al. [7] proposed Weighted Majority Merge (WMM) where string length was considered to be the weight of the string. WMM showed better performance than MM where the problem has no structure or the structure is deceptive [7].

C. Deposition and Reduction

Deposition and Reduction (DR) Algorithm was proposed by K. Ning et al. [8]. The algorithm includes two processes Deposition and Reduction. In deposition process a small set of SCS templates is generated. Each template is a common supersequence of the given set of strings. To produce templates two algorithms (Look Ahead Sum Height (LA-SH) algorithm and Alphabet algorithm) have been used. The reduction process shortens these templates by removing some characters while preserving the common supersequence property. Finally, the shortest result obtained after this reduction process is selected as the final output of the algorithm. The overall performance of the algorithm fully depends on generating SCS templates. But the algorithm Alphabet that is used to generate the templates has an approximation ratio of $|\sum|$. That means the worst case approximation ratio of DR algorithm is $|\sum|$ which is not satisfactory.

D. Reduce-Expand Algorithm

Reduce-Expand (RE) algorithm has been proposed by Paolo Barone et al. [9]. First of all, RE reduces the sequence into basic sequences. Basic sequences means the sequence will contain different symbols in adjacent position of SCS. Now, expand process tries to add symbols to SCS preserving the characteristics of common supersequence. As similar to DR it has a worst case approximation ration of $|\sum|$.

III. Chemical Reaction Optimization

Chemical Reaction Optimization (CRO) is meta-heuristic based on mimicking the behavior of chemical reaction. In our universe every unstable molecule with higher potential energy wants to get stabilized of low potential energy by reacting with other unstable molecule or the surroundings. A chemical reaction is accomplished by some sub reactions and after every sub reaction a more stable product is generated. On the same time a checking is done whether the product is at optimal point or not. So it is a multi-step optimal point searching. This behavior is very similar to many real world optimization problems. Researchers mimic this natural phenomenon to solve optimization problem. To design the algorithm another important principle of thermodynamics has been considered. Energy cannot be created or destroyed rather it is transformed into one form to another. Researchers use potential energy (PE) and kinetic energy (KE) as the energy of a molecule and central energy buffer as the energy of the surroundings. The potential energy of a reaction is referred as the objective function in an optimization problem and kinetic energy as a numeric value that quantifies how much a molecule can tolerate the worst value. So the acceptance of a change during chemical reaction is done by

$$\sum_{i=1}^{m}(PE_{xi}+KE_{xi}) \geq \sum_{j=1}^{n} PE_{x'j} \quad (2)$$

Where $1 \leq m \leq popsize$

Here m numbers of reactants (x) collide to form n number of products (x') and *popsize* is the number of population. Energy transformation has a big role for chemical reaction. A central energy buffer is used to adjust the energy distribution on the basis of conservation of energy principle. This allows the algorithm to search different region of the search space.

CRO is one of the meta-heuristic methods that is being used to solve optimization problem efficiently. It is a powerful method that mimics the interaction of molecules in chemical reaction to search for the global optimum [15].

Meta-heuristic methods outperformed over heuristic method for solving optimizing problem because of two reasons. Firstly meta-heuristic methods can solve a wide range problem with little or no knowledge of the search space. So they can easily adjust to fit the problem. And it can reach optimal or near optimal solutions in a considerable time though the optimization problem are intractable and NP-Hard and optimal solution cannot be obtained in a polynomial time. CRO has been used to solve different optimization problem recently such as CROG for 0-1 Knapsack problem [16], both Parallel CRO [15] and singular CRO [17] for Quadratic Assignment Problem, Artificial CRO for Multiple Choice Knapsack problem [18], Population Transition Problem in peer live streaming [19], Grid Scheduling problem [20], Artificial neural network training [21], Network coding optimization problem [22] etc.

CRO algorithm is designed for SCS problem because of the fact that, it has both the diversification and intensification properties. So from any sort of generated population it can travel the solution space very efficiently. Through the reaction operators we can easily traverse the solution space and find the near optimal solution very quickly. Besides, being a meta-heuristic approach it can be fitted to SCS problem having no prior information about the problem and its variable population size can easily adjust the system for solving the problem. Moreover, newly designed reform function also ensures the constraints and the quality of solutions of this problem.

A. CRO Algorithm

The CRO algorithm includes three stages such as 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 MolColl. At the end of each iteration, we have to check the termination criteria. The CRO Algorithm is shown in Algorithm1.

Algorithm1 (CRO Algorithm):
Input: Problem specific information (Objective Function, constraints and the dimension of the problem)
1: Initialize population with random solutions and set the parameters.
2: Compute the fitness value of each molecule as PE.
3: Let the central energy buffer be buffer and initialize Buffer=0
4: **while** stopping criteria not met **do**
5: Choose one reaction from the four elementary collisions according to certain rules.
6: Select the molecule(s) for reaction
7: Generate the new molecule(s)
8: **if** the new solution acceptance rules satisfied **then**
9: Substitute new molecule(s) for original one(s)
10: Update the KE for new molecule(s)
11: Update the central energy buffer.
12: **else**
13: Keep the original molecule(s)
14: **end while**
Output: The overall minimum solution and its function value

IV. Design CRO for SCS Problem

A. Population Generation

Initially, the population is generated on the basis of the concept of random selection which has been followed from [10]. Here the frequency of each symbol of alphabet $\sum$ is calculated in Array1. Then in Array2, the frequency is converted to the strings. For example if $\sum$ = {a, c, g, t} and set of strings L = {acctg, cttcg, acact, gtgca} then the structure of Array 1 and Array 2 is depicted in *Fig. 2*.

Array 1

a	c	g	t
4	7	4	5

Array2

1	2	3	4	5	6	7	8	9	10
a	a	a	a	c	c	c	c	c	c

11	12	13	14	15	16	17	18	19	20
c	g	g	g	g	t	t	t	t	t

Fig. 2. Population Generation

Now the symbols from Array 2 are randomly selected to create the population. During selection of symbols frequency of the symbols will be considered. Since the population size is 20 then 20 common supersequences will be created. For example Pop={acctacgatacg, catctgacgtag, …..}.

B. Solution Representation

Each symbol in Alphabet $\sum$ is initialized with a value. The integer string of the value of the corresponding symbol represents a solution. For example let $\sum$ = {a, c, g, t} have a value of {0, 1, 2, 3}. *Fig. 3* represents the solution for the above example.

a	g	t	c	t	a	c

0	2	3	1	3	0	2

Fig. 3. Solution Representation

C. Reaction Operators

We have designed four types of reaction operators in this algorithm which are described as follows

1) On-wall ineffective collision

This is a molecular reaction used for the neighborhood search in solution. Well known evolutionary algorithm mutation operator (*Fig. 4*) is used here. A position *i* in solution *m* will be chosen and a small value is added or subtracted from the value. A small value will be chosen randomly. Reformation function is used to repair the invalid solution after the operation. Reformation function is discussed later.

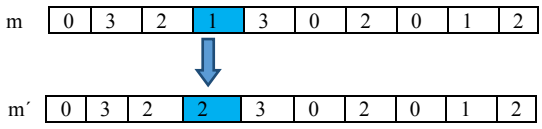

Fig. 4. Mutation Operator

2) Decomposition

The process divides a solution *m* into two solutions. The operator takes a random solution from the pop and creates two solutions m_1' and m_2' dividing from the middle of the solution *m* (*Fig. 5*). Reform function is incorporated to check the validity of new solutions. This operator causes the diversification and makes the algorithm explore in the search space.

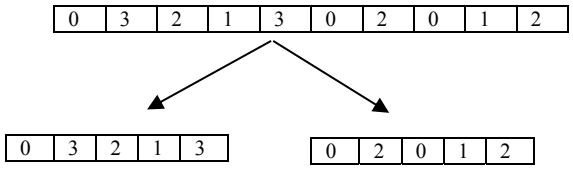

Fig. 5. Decomposition Operator

3) Inter-molecular ineffective collision

The operator takes two solutions m_1 and m_2 randomly from the pop and using two point crossover operators commonly used in the genetic algorithm (*Fig. 6*) produces two new solutions m_1' and m_2'.

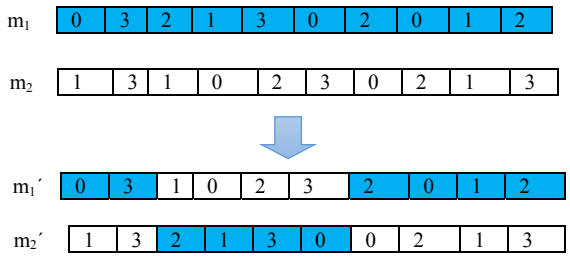

Fig. 6. Two-point crossover operator

The procedure is taking two points x_1 and x_2 randomly and then dividing both the solution into three parts.Now merging 0 to x_1 and x_2 to n part of m_1 and x_1 to x_2 part of m_2 will form m_1' and merging other three parts will form m_2'. Neighbourhood search is implemented using inter-molecular ineffective collision.

4) Synthesis

Synthesis operator combines two molecules into one solution. Probabilistic select operator (*Fig. 7*) is used for synthesis reaction. The process is done by taking one random variable from 0 to 1 and if the value is less than 0.5 then symbol from the m_1 will be chosen and appended to the supersequence. Otherwise, a symbol from m_2 will be chosen. It works as the opposite of the decomposition reaction. Synthesis reaction implements the global search. Massive change in the molecular structure causes exploration throughout the solution space and can avoid getting stuck in local optima.

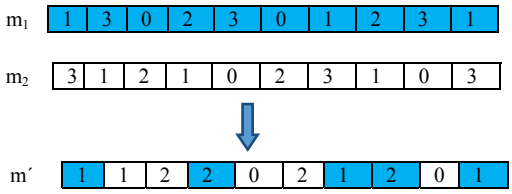

Fig. 7. Synthesis Operator

D. Reform Function

Reform function is used after every reaction to check the constraints are violated or not. For that, the newly formed solution is checked with every string in the set of strings. If the newly formed strings become the common supersequence for all the strings then it will be selected and the previous strings are removed from the pop. Otherwise, the change made by the reaction will be discarded. Besides, if the newly formed supersequence is found valid then the symbols having frequency zero are removed from the supersequences.

E. Termination Condition

After a particular computation time, the algorithm will be terminated and the optimal length of the supersequence from the population will be considered as the output of the algorithm. The computation time is based on the output of the others algorithm to compare the length of the supersequence for algorithms in a particular time

F. Parameter Settings

Chemical Reaction Optimization has six parameters (*KELossRate, InitialKE, MoleColl, Popsize,* α and β). Out of this seven parameter, first four have been taken from [15] where *KELossaRate*=0.8, *InitialKE*=100, *PopSize*=20, *MoleColl*=0.2. Two thresholds values (α and β) are required to check the condition for decomposition reaction and

synthesis reaction. Like [13] we set the value of α as $\alpha \in [10,500]$ and $\beta \in [10,500]$.

V. Experimental Result

The experiment has been done on a personal Toshiba laptop Model no: Satellite A305-S6905 with Intel Core 2 Due T5800 CPU at 2.00 GHz, 3 GB RAM and running on Windows 10 (64 bit). All algorithms have been implemented on Java SE Development Kit 7 platform and Eclipse Ide.

The datasets that have been used for all algorithms are taken from [23] which are also used by K. Ning et al. for implementing DR algorithm [8]. For all cases, ten random instances were used and average outcome results of CRO algorithm are being compared with DR, WMM, RE and Dynamic Programming (DP) algorithm. Due to the exponential behavior of DP algorithm, the experiment could not be conducted for the number of strings more than 500. Moreover, RE and WMM algorithm also take a lot of time for the strings those have lengths over 100. Therefore, the dataset is restricted up to a number of strings 500 and length of string 100. Standard deviation and execution time are also measured as the programs are executed for ten instances and the average values are taken as final outputs. Execution times are measured in seconds.

TABLE I. Comparison of length, Standard Deviation,Execution Time of Supersequence between DP, WMM, RE, DR and CRO-SCS Algorithm

Algorithm	Number of string	Length of string	Length of returned String	Standard Deviation	Execution Time(s)
DP	5	10	20	0	5
WMM	5	10	27.5	1.16	2.4
RE	5	10	20.97	0.73	1.10
DR	5	10	20.3	1.12	0.1
CRO-SCS	5	10	20.3	0.54	0.005
DP	10	10	24	0	19
WMM	10	10	32	2.43	13.5
RE	10	10	26.45	1.25	6.49
DR	10	10	25.7	1.56	0.7
CRO-SCS	10	10	24.9	1.11	0.8
DP	100	10	30	0	38
WMM	100	10	43.7	1.98	21
RE	100	10	35.78	1.44	11.76
DR	100	10	32.1	1.68	8.2
CRO-SCS	100	10	31.9	0.81	1.1
DP	5	100	174	0	66
WMM	5	100	205.8	3.47	32.83
RE	5	100	190.7	6.23	21.72
DR	5	100	194.3	2.87	7.1
CRO-SCS	5	100	186.7	1.75	3.2
DP	10	100	204	0	156
WMM	10	100	295.7	8.46	98
RE	10	100	229.78	7.65	64.37
DR	10	100	227.6	7.1	21.6
CRO-SCS	10	100	222.3	6.47	7.6
DP	100	100	247	0	1837
WMM	100	100	396	6.54	782.7
RE	100	100	287.65	11.76	658.65
DR	100	100	276.6	6.23	37.5
CRO-SCS	100	100	261.6	7.20	13.8
DP	500	100	262	0	3966
WMM	500	100	432	11.7	1378.7
RE	500	100	302.33	17.5	2065.65
DR	500	100	289.6	5.1	1087.3
CRO-SCS	500	100	277.4	6.9	176.5

The experimental results demonstrate that DP gives the optimal results in every case whereas CRO-SCS takes much less time than the DP to find the most near optimal SCS. The difference of length between DP and CRO-SCS increases as the number of strings enlarges. Besides CRO-SCS gives a better result than DR, RE and WMM algorithm in less execution time. Comparison between CRO-SCS, DR, RE, WMM shows that in every case CRO-SCS gives better performance than all other algorithms with less time. Both the quality and the consumption of computational time for CRO-SCS are much better than all other algorithms. For the largest instances, our proposed algorithm takes less than three minutes whereas DR takes over 18 minutes, WMM takes almost 23 minutes, RE consumes 34 and half minutes and DP takes over an hour. That proves the robustness of our proposed algorithm. Moreover, standard deviation suggests that even CRO-SCS has less variation of results than other three approximate algorithms. That demonstrate the our proposed algorithm can return results that will very similar if we execute the algorithm multiple times for same instances and that ensures the reliability of our proposed algorithm.

VI. Conclusion

In this paper, we have reviewed a well-known NP-hard problem Shortest Common Supersequence (SCS) and discussed the different approaches for solving SCS problem. Then we have proposed a meta-heuristic algorithm Chemical Reaction Optimization (CRO-SCS) for solving SCS problem. CRO-SCS algorithm has been designed and parameters are being set according to previous work done by a different author for solving different NP-hard optimization problems by CRO. The combination of local search and global search by the reaction operators ensure the completeness of our algorithm. Besides randomness property in the algorithm governs the searching procedure to a near optimal solution. The experimental results of CRO have been compared with DR, DP, RE and WMM algorithms. DP shows optimal results but it takes very long time whereas CRO-SCS gives near optimal results and takes much less time. Besides, in every case CRO-SCS outperforms DR, RE and WMM both in SCS length and execution time. The robustness and the reliability are ensured by the proposed approaches. Our future target is to

do experiments for more instances and some real datasets which include the DNA and Protein sequences.

REFERENCE

[1] S. Rahmann, "The shortest common supersequence problem in a microarray production setting", *Bioinformatics*, vol. 19, no. 2, pp. ii156-ii161, 2003.

[2] D. Foulser, M. Li and Q. Yang, "Theory and algorithms for plan merging", *Artificial Intelligence*, vol. 57, no. 2-3, pp. 143-181, 1992.

[3] S. Chaudhuri and B. Nicolas, "Method and apparatus for generating statistics on query expressions for optimization.", U.S. Patent No.7,330,848., 2008.

[4] V. Timkovskii, "Complexity of common subsequence and supersequence problems and related problems", *Cybern Syst Anal*, vol. 25, no. 5, pp. 565-580, 1990.

[5] C. Fraser, "Subsequences and supersequences of strings.", *BULLETIN-EUROPEAN ASSOCIATION FOR THEORETICAL COMPUTER SCIENCE*, no. 57, pp. 355-355, 1995.

[6] T. Jiang and M. Li, "On the Approximation of Shortest Common Supersequences and Longest Common Subsequences", *SIAM Journal on Computer.*, vol. 24, no. 5, pp. 1122-1139, 1995

[7] J. Branke, M. Middendorf and F. Schneider, "Improved heuristics and a genetic algorithm for finding short supersequences", *OR Spektrum*, vol. 20, no. 1, pp. 39-45, 1998.

[8] K. Ning and H. Leong, "Towards a better solution to the shortest common supersequence problem: the deposition and reduction algorithm", *BMC Bioinformatics*, vol. 7, no. 4, p. S12, 2006.

[9] P. Barone, P. Bonizzoni, G. Vedova and G. Mauri, "An approximation algorithm for the shortest common supersequence problem: an experimental analysis", in *ACM symposium on Applied computing*, 2001, pp. 56-60.

[10] S. Mousavi, F. Bahri and F. Tabataba, "An enhanced beam search algorithm for the Shortest Common Supersequence Problem", *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 457-467, 2012.

[11] M. Noaman and A. Jaradat, "Solving shortest common supersequence problem using artificial bee colony algorithm.", in *Int J ACM Jordan 2.3*, Jordan, 2011, pp. 180-185.

[12] C. Blum, C. Cotta, A. FernÃ¡ndez and J. Gallardo, "A probabilistic beam search approach to the shortest common supersequence problem.", in *Evolutionary Computation in Combinatorial Optimization*, Berlin Heidelberg, 2007, pp. 36-47.

[13] R. Michel and M. Middendorf, "An ACO algorithm for the shortest common supersequence problem", 1999.

[14] J. Branke and M. Middendorf, "Searching for shortest common supersequences by means of a heuristic-based genetic algorithm", in *University of Vaasa*, 1996.

[15] J. Xu, A. Lam and V. Li, "Parallel chemical reaction optimization for the quadratic assignment problem.", in *World Congress in Computer Science, Computer Engineering, and Applied Computing*, 2010.

[16] T. Truong, K. Li and Y. Xu, "Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem", *Applied Soft Computing*, vol. 13, no. 4, pp. 1774-1780, 2013.

[17] A. Lam and V. Li, "Chemical-Reaction-Inspired Metaheuristic for Optimization", *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 381-399, 2010.

[18] T. Truong, K. Li, Y. Xu, A. Ouyang and X. Tang, "An artificial chemical reaction optimization algorithm for multiple-choice knapsack problem.", in *Proceedings of the International Conference on Artificial Intelligence (ICAI), p. 1.*, 2013.

[19] A. Lam, J. Xu and V. Li, "Chemical reaction optimization for population transition in peer-to-peer live streaming", in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, 2010, pp. 1--8.

[20] J. Xu, A. Lahm and V. Li, "Chemical reaction optimization for the grid scheduling problem", in *Communications (ICC), 2010 IEEE International Conference on*, 2010, pp. 1---5.

[21] J. Yu, A. Lahm and V. Li, "Evolutionary artificial neural network based on chemical reaction optimization", in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, 2011, pp. 2083-2090.

[22] B. Pan, A. Lahm and V. Li, "Network coding optimization based on chemical reaction optimization", in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1---5.

[23] <http://www.biomedcentral.com/content/supplementary/1471-2105-7-S4-S12-S1.zip>[Accessed: Jun- 2015].