

Chemical Reaction Optimization for Max Flow Problem

Reham Barham

Department of Computer Science
King Abdulla II School for
Information and Technology
The University of Jordan
Amman, Jordan

Ahmad Sharieh

Department of Computer Science
King Abdulla II School for
Information and Technology
The University of Jordan
Amman, Jordan

Azzam Sliet

Department of Computer Science
King Abdulla II School for
Information and Technology
The University of Jordan
Amman, Jordan

Abstract—This study presents an algorithm for MaxFlow problem using "Chemical Reaction Optimization algorithm (CRO)". CRO is a recently established meta-heuristics algorithm for optimization, inspired by the nature of chemical reactions. The main concern is to find the best maximum flow value at which the flow can be shipped from the source node to the sink node in a flow network without violating any capacity constraints in which the flow of each edge remains within the upper bound value of the capacity. The proposed MaxFlow-CRO algorithm is presented, analyzed asymptotically and experimental test is conducted. Asymptotic runtime is derived theoretically. The algorithm is implemented using JAVA programming language. Results show a good performance with a complexity of $O(I E^2)$, for I iterations and E edges. The number of iterations I in the algorithm, is an important factor that will affect the results obtained. As number of iterations is increased, best possible max-Flow value is obtained.

Keywords—Chemical reaction optimization(CRO); Decomposition; Heuristic; Max Flow problem; Molecule; Optimization; Reactions; Synthesis

I. INTRODUCTION

Imagine a material traveling without obstruction through a system from a source, where the material is produced, to a destination, where it is consumed. The flow of the material at any point in the system is the rate at which the material moves [1]. We can interpret the weighted directed graph as a flow networks. Flow networks can represent many real life situations like petrol flowing through conduits, and parts through assembly lines [1]. The maximum flow problem becomes one of the most well known problems for combinatorial optimization in the weighted directed graph [2]. This problem can be applied in many areas of applications such as networks, engineering, and transportations [2].

Related to its important, many researches solved this problem using different methods and techniques [1]. In the maximum flow problem, the goal is to get the greatest rate at which we can ship the material from the produced point to the consumed one without violating any capacity constraint in which the flow of each edge remains within the upper bound value of the capacity [1].

Toshinori et al. in [2] define the max flow problem as: "The problem is to determine an optimal solution for a given directed, integer weighted graph. The weight at each edge

represents the flow capacity of the edge. Under these constraints, we want to maximize the total flow from the source to the sink". In [3] they define it as " In deterministic networks, the maximum-flow problem asks to send as much flow (information or goods) from a source to a destination, without exceeding the capacity of any of the used links. Solving maximum-flow problems is for instance important to avoid congestion and improve network utilization in computer networks or data centers or to improve fault tolerance".

In this study, a potential solution to the maximum flow problem using a recent algorithm which is called "Chemical Reaction Optimization (CRO)" is investigated. This algorithm, as it is reported in [4]: "CRO is a recently established meta-heuristics for optimization, inspired by the nature of chemical reactions. A chemical reaction is a natural process of transforming the unstable substances to the stable ones. The molecules interact with each other through a sequence of elementary reactions. At the end, they are converted to those with minimum energy to support their existence. This property is embedded in CRO to solve optimization problem".

Furthermore, CRO is a technique which loosely couples chemical reactions with optimization. It does not attempt to capture every detail of chemical reactions [4]. In general, the principles of chemical reactions are governed by the first two laws of thermodynamics. The first law (conservation of energy) says that energy cannot be created or destroyed; energy can transform from one form to another and transfer from one entity to another. The second law says that the entropy of a system tends to increase, where entropy is the measure of the degree of disorder [4]. Potential energy is the energy stored in a molecule with respect to its molecular configuration [4].

The rest of this study is organized as follow. Section II will present the literature review. Section III will illustrate the maximum flow problem. Section IV will show how chemical reaction optimization works. Section V presents the proposed algorithm. Section VI shows the analysis of the algorithm. An illustrated example is presented in section VII. The experimental results are presented in section VIII. Section IX is the conclusion and future work.

II. LITERATURE REVIEW

The maximum flow problem has been widely studied by many researchers using several methods. The first pseudo-polynomial algorithm for the maximum flow problem is the augmenting path algorithm of Ford and Fulkerson (1956) [5, 6]. Dinic [7] and Edmonds and Karp [8] independently obtained polynomial versions of the augmenting path algorithm. Edmonds and Karp (1972) and Dinic (1970) independently proved that if each augmenting path is shortest one, the algorithm will perform $O(nm)$ augmentation steps, where n is the number of vertices, and m is the number of edges in the graph. The shortest path (length of each edge is equal to one) can be found with the help of breadth-first search (BFS) algorithm. Since then, several more-efficient algorithms have been developed. Ahuja and Orlin improved the shortest augmenting path algorithm in 1987 [12]. The push and re-label method is introduced by Goldberg [9] and Goldberg and Tarjan [10], along with some of its more efficient variants. It maintains a preflow and updates it through push operations. It introduces the relabel operation to perform fine-grain updates of the vertex distances. Orlin [14] presents improved polynomial time algorithms for the max flow problem defined on a network with n nodes and m arcs, and shows how to solve the max flow problem in $O(nm)$ time, improving upon the best previous algorithm due to King, Rao, and Tarjan [15] who solved the max flow problem in $O(nm \log m / (n \log n))$ time.

Genetic algorithm (GA), which is considered as evolutionary algorithm, has been also applied to solve max flow optimization problems such as in [2]. In [2], each solution is represented by a flow matrix. The fitness function is defined to reflect two characteristics: balancing vertices and the saturation rate of the flow. Starting with a population of randomized solutions, better and better solutions are sought through the genetic algorithm. Optimal or near optimal solutions are determined with a reasonable number of iterations compared to other previous GA applications.

CRO is a recently proposed general-purpose meta-heuristic, which has been developed intensely in the past few years [11]. The CRO was proposed by Lam et al. in 2010, and was originally designed for solving combinatorial optimization problems. They solved some classical problems, e.g., quadratic assignment problem and channel assignment problem. It could also provide solutions to some applications solved by other evolutionary algorithms such as genetic algorithm. As in [13], they use genetic algorithm to deal with optimization of parameters of constructive cost model. Therefore, CRO can be used for these types of applications. In this study, we proposed to use chemical reaction optimization algorithm to solve the maximum flow problem.

III. MAXIMUM FLOW PROBLEM

Suppose there is a directed network $G = (V, E)$ defined by a set V of nodes (or vertices) and a set E of arcs (or edges). Each arc (i, j) in E has an associated nonnegative capacity u_{ij} where i, j are nodes in V . Also, there are two distinguished special nodes in G : a *source* (or start) node s and a *sink* (or a target) node t [1]. For each i in V , denote by $E(i)$ all the arcs emanating from node i . Let $U = \max \{u_{ij} \mid (i, j) \in E\}$. Denote the number of vertices by n and the number of edges by m [1].

The goal is to find the maximum flow from the source node s to the sink node t that satisfies the arc capacities and mass balance constraints at all nodes [2]. Representing the flow on arc (i, j) in E by x_{ij} , an optimization model for the maximum flow problem can be obtained as in (1) [1]:

$$\begin{aligned} \text{Maximize } f(x) &= \sum_{(i,j) \in E(s)} x_{ij} \text{ subject to } \dots (1) \\ \sum_{\{j:(i,j) \in E\}} x_{ij} - \sum_{\{j:(j,i) \in E\}} x_{ji} &= 0 \quad \forall i \in V \setminus \{s, t\} \\ 0 \leq x_{ij} &\leq u_{ij} \quad \forall (i, j) \in E \end{aligned}$$

IV. CHEMICAL REACTION OPTIMIZATION (CRO)

The CRO algorithm as described in [4]: "is a multi-agent algorithm and the manipulated agents are molecules. Each molecule has several attributes, some of which are essential to the basic operations of CRO. The essential attributes include: (a) the molecular structure (ω); (b) the potential energy (PE); and (c) the kinetic energy (KE). Other attributes depend on the algorithm operators and they are utilized to construct different CRO variants for particular problems provided that their implementations satisfy the characteristics of the elementary reactions. Molecular structure ω captures a solution of the problem. It is not required to be in any specific format: it can be a number, a vector, or even a matrix. Potential energy PE is defined as the objective function value of the corresponding solution represented by ω . If f denotes the objective function, then $PE(\omega) = f(\omega)$. Kinetic energy KE is a non-negative number and it quantifies the tolerance of the system accepting a worse solution than the existing one".

There are four types of elementary reactions in CRO, each of which takes place in each iteration of CRO. They are employed to manipulate solutions (i.e. explore the solution space) and to redistribute energy among the molecules [4]. Assume that molecules are in a container, one of the following four reactions will be possible to occur in each of CRO iteration.

A. On-wall ineffective collision

An on-wall ineffective collision represents the situation when a molecule collides with a wall of the container and then bounces away remaining in one single unit but with new structure due to collision with wall. In this collision, ω produces ω' , i.e., $\omega \rightarrow \omega'$ [4].

B. Decomposition

Decomposition refers to the situation when a molecule hits a wall of the container and then breaks into several parts. Assume that ω produces ω_1 and ω_2 , i.e., $\omega \rightarrow \omega_1 + \omega_2$ [4].

C. Inter-molecular ineffective collision

Inter-molecular ineffective collision takes place when two molecules collide with each other and then bounce away. Assume that ω_1 and ω_2 , after collision with each other, produce ω_1' and ω_2' , i.e., $\omega_1 + \omega_2 \rightarrow \omega_1' + \omega_2'$. This reaction could be similar to on-wall ineffective collision but with two molecules to collide rather than one [4].

D. Synthesis

Synthesis does the opposite of decomposition. A synthesis happens when multiple (assume two) molecules hit against each other and fuse together, i.e., $\omega_1 + \omega_2 \rightarrow \omega'$ [4].

In order to find the maximum flow using CRO, there is a need to explore the search space. To achieve that, number of solutions, which are called Molecules in CRO, need to be generated. Each of these solutions or 'molecules' has its own potential energy which is the objective function value for that solution, and a kinetic energy which helps in making decision; if the generated solution is better than its parent or not. Now imagine that these molecules are in a container and due to their high kinetic energy, they move inside the container in arbitrary directions. This situation, leads to perform collisions between these molecules in different forms. These collisions are known as chemical reactions. Chemical reactions take different shapes and conditions [4].

CRO mimics these reactions and use them as operators to help moving the molecules from local optimum into global optimum as possible [4]. CRO, as mentioned before, has four types of reactions. These types differ from each other based on the nature of the reaction and the numbers of molecules participate in the reaction [4]. If one molecule is selected, and an ineffective collision is occurred, then it is on-wall ineffective reaction. If an effective reaction occurs, then the decomposition reaction will happen. In the other hand, if two molecules are selected, then there are possibilities of occurrence of synthesis reaction which is effective reaction, or inter-molecular reaction which is ineffective reaction [4].

Through these reactions, some of good solutions will be generated and some of bad ones will be destroyed. After particular number of iterations or stop criteria is met, the best solution among available solutions in the container will be selected and considered.

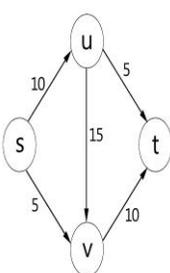
V. ALGORITHM: "MAXFLOW-CRO"

Now let us apply CRO to find the possible solution for Maximum Flow Problem. Figures 1 through 6, present the pseudo-code for the proposed "MaxFlow_CRO" algorithm. TABLE 1 shows the main attributes and their meaning related to the proposed algorithm "MaxFlow_CRO" for the molecules, in comparison with chemical meaning in CRO.

A. MaxFlow-CRO initialization stage

As in [4], CRO has three main stages: the initialization, the iterations, and the final stage. The MaxFlow-CRO algorithm, as shown in Figure1: lines1, 11 and 31, presents these three stages. First, the initialization stage can be shown in Figure1 as in lines 1-10. From Section III, maximum flow problem has a graph of nodes and edges; each edge is a directed and weighted edge. These weights represent the capacity for those edges. This graph can be represented as a capacity matrix $C[i][j]$, and used as a basis to generate number of flow matrices, Figure1:lines2 and 10, which are called parents. There are number of generated parents based on the value setting for parentSize variable (Figure1: line3). S is the source node, and t is the sink. Iteration number variable is important here, it is used as a stop criterion (Figure1:line3).

TABLE I. PROFILE FOR CRO-MAXFLOW MOLECULE

Profile: Flow network	Chemical meaning	CRO-MaxFlow meaning
	Molecular structure	Candidate solution: flow matrix
	Potential energy	Value of maximum flow for such candidate solution
	Kinetic energy	Measure of tolerance of having worse solution
	Number of hits	Current total number of iterations
	Minimum structure	Current optimal max flow solution

After that, a number of parents based on the specified value of the parentSize variable (Figure1: line10), will be generated. The generated parents are represented by a flow matrix $F[i][j]$. Parent, solution and molecule are used interchangeably. These parents represent the first generation in the population. Generating parents had been a challenge because in order to consider them as max flow solutions, they should obey the constraints as mentioned in formula (1). Max flow problem has two common constraints. The first one says the generated flow must not exceed the identified capacity for any edge. The second says that for each node in the graph, the incoming flow must equal to the outgoing flow. The first constraint could be controlled by specifying the upper range of the generated flow based on the capacity on an edge. Generating the flow randomly will make second constraint hard to satisfy. So, MaxFlow-CRO use a function called objectiveFunction() to repair the flow and compute the current max flow value. Therefore, a feasible solution meets the constraints can be obtained. The details of parent generating function can be shown in Figure 2.

The objective function (Figure2: line5) can be computed using the known shortest augmenting path algorithm as in [8]. The max flow value, computed by the objective function, represents the current maximum flow value, and as the iterations and reactions proceed, this value will be improved gradually. Note that, objective function and PE are used interchangeably.

In Figure1, lines 6 and 7, α is a decomposition threshold, β is a synthesis threshold, and both of them are initially assigned by dividing parentSize variable value by 2. The reason from computing α and β like that, to make sure that synthesis and intermolecular reactions will be possibly happened after number of on-wall and decomposition reactions are triggered. The goal is to degrade the chance of synthesis reaction occurrence. Both of decomposition and on-wall reactions help to save or increase the number of parents, but synthesis reactions tend to decrease number of parents.

B. MaxFlow-CRO Iteration stage

The goal from CRO reactions is to increase the ability to improve the resulted value of max flow. As iterations operate,

the max flow value will be improved gradually until reach the best max flow. Iteration stage is shown in Figure 1: lines 11-31. After generating the required parents, PE for each of them will be computed. Then one or two molecules will be picked randomly based on the value of the variable "b". The variable b is generated randomly between 0 and 1 (Figure1: line12). If the value of b is larger than the value of the variable "molecule", which is a predefined variable used as a threshold for the variable b, then one molecule will be selected; but if not, then two molecules will be selected. Also, the value of the molecule variable will be generated randomly between 0 and 1 (Figure1:line9).

If one molecule is selected, there is a possibility to on-wall ineffective or decomposition reactions to happen. So, to be more accurate, CRO as in [4] puts a criteria, which is $HIT > \alpha$ (Figure1: line15), where HIT is a variable counts the number of times the specific molecule participate in a reaction and initialized to 0 (Figure1: line5 and 29). If this criterion has been met, then decomposition reaction will be occurred (Figure1: line16). Else, the on-wall ineffective collision will be occurred (Figure1: line18).

The same will happen if two molecules are selected but here the reactions are different. In this case, the possibility is for synthesis and inter-molecular reactions to be happened. There are some conditions that must be satisfied, which are $KE \leq \beta$ [4] (where KE is kinetic energy) and the number of parents is not less than 2. This is to be sure that synthesis and inter-molecular reactions can be occurred correctly (Figure1: line23). If the criterion in line 23 has been met, then synthesis reaction will occur (Figure1: line24). Else, the inter-molecule ineffective reaction will occur (Figure1: line26). Each time the reaction happens, the KE will be decremented by one (Figure1: line30).

In the final stage and after finishing the specific number of iterations, which was the stop criterion, the best molecule which has the largest maximum flow value obtained will be selected and its maximum flow value will be returned (Figure1: lines 31-33).

C. Reactions

When on-wall ineffective function is called (Figure3), the chosen molecule will be changed by regenerating randomly the flow of its first half, (Figure3: lines1-3). Then, the objective function will be computed and the max flow value is compared with that for the original molecule. If it is greater than the original, then it will be confirmed and the original solution will be destroyed; else the generated molecule will be destroyed.

The same will be happened when intermolecular ineffective collision is called (Figure4), but in this case, there are two molecules instead of one. Apply the same procedure used for on-wall function for each molecule separately and test the resulted molecules if they have max flow greater than the original molecules or not. Figure4: lines 1-13 are for the first molecule. Lines 14-26 are for the second one. In synthesis function (Figure6), the molecules with large max flow value will be returned and the other ones will be destroyed (Figure6: lines 1-7).

Algorithm: MaxFlow-CRO

```
1 //initialization phase
2 Set flow_network_size, C[i][j]: maximum capacity
3 parentSize, iterationNumber, s: source node, t:
4 sink node
5 HIT= 0
6  $\beta = \text{parentSize}/2$ 
7  $\alpha = \text{parentSize}/2$ 
8  $KE = \text{parentSize}/1.5$ 
9 Generate molecule  $\in [0, 1]$ 
10 parentGenerating(C[i][j] , parentSize )
11 for (int i=1 to iterationNumber) // Iteration phase
12   Generate b  $\in [0, 1]$ 
13   if b > Molecule then
14     Randomly select one parent
15     if (HIT >  $\alpha$ ) then
16       Decomposition( )
17     else
18       OnWallIneffectiveCollision( )
19   end if
20
21 else
22   Randomly select two molecules
23   if ( $KE \leq \beta$  && parentSize  $\geq 2$  ) then
24     Synthesis( )
25   else if (parentSize  $\geq 2$ )
26     IntermolecularIneffectiveCollision( )
27   end if
28 end if
29 HIT++
30 KE--
31 Check for any new maximum solution
32 end for-loop // final stage
33 return the best solution found
```

Fig. 1. Pseudo-code for "MaxFlow-CRO" algorithm

Function: parentGenerating()
Input: C[i][j]: maximum capacity, parentSize
Output: F_i: flow matrices

```
1 for ( int i=1 to popSizs )
2   for (int j=1 to no. of edges in the graph)
3     randomly generate Fi[i][j]
4     // where  $0 \leq F_i[i][j] \leq C[i][j]$ 
5     PEi = objectiveFunction(Fi)
6     return Fi, PEi
7   end for-loop
8 end for-loop
```

Fig. 2. Pseudo-code for parent initialization function

Function: onWallIneffectiveCollision()
Input: F_i //one molecule
Output: F_i' // new molecule

```
1 for (j=1 to flow_network_size/2)
2   F1'=generate new flow randomly
3 end for-loop
4 //Compute objective function for the new molecule
5 PE' = objectiveFunction(F1')
6 //confirm the new solution or dismiss
7 if (PE'> PE ) then
8   destroy Fi
9   return F1'
10 else
11   dismiss F1'
12 End if
```

Fig. 3. Pseudo-code for on-wall ineffective collision function

Function: IntermolecularIneffectiveCollision()
Input: F₁, F₂ // two molecules
Output: F₁', F₂' //two new molecules

```
1 // for F1
2 for (j=1 to flow_network_size/2)
3   F1'= generate new flow randomly
4 end for-loop
5 PE1' = objectiveFunction(F1')
6 //Compute PE' objective function for the
7   new molecule
8 if (PE1'> PE1 ) then // solution confirmed
9   destroy F1
10  return F1'
11 else
12  dismiss F1'
13 end if
14 // for F2
15 for (j=1 to flow_network_size/2)
16   F2'= generate new flow randomly
17 end for-loop
18 PE2' = objectiveFunction(F2')
19 //Compute PE' objective function for the
20   new molecule
21 if (PE2'> PE2 ) then // solution confirmed
22   destroy F2
23   return F2'
24 else
25   dismiss F2'
26 end if
```

Fig. 4. Pseudo-code for inter-molecule ineffective collision function

Figure5 shows the decomposition function. In this function one molecule is used to produce two molecules. First, two copies, for the original molecule, are produced (Figure 5: lines 2 and 15). For the first copy, take the first half of flows and randomly regenerate them (Figure5: lines 3-5), then compute the objective function (Figure5: lines7) and compare it with that for the original (Figure5: lines 8-13). In order to confirm the new molecule, it should have max flow greater than that for the original. The same occurs for the second copy but regenerate the second half of flows in order to produce a molecule different from the first one (Figure5: lines 15-18). Then compute the objective function and compare the result

with that for original to confirm or destroy the second new molecule (Figure5: lines19-27).

Function: Decomposition()
Input: F_i // one molecule
Output: F₁', F₂' // two new molecules

```
1 // generate the first solution F1'
2 Copy the original solution Fi into F1
3 for (i=1 to network_flow_size/2)
4   F1'= Randomly generate new flow
5 end for-loop
6 // objective function for the new molecule
7 PE1'=objectiveFunction(F1')
8 if (PE1'> PEi ) then // solution confirmed
9   destroy Fi
10  return F1'
11 else
12  dismiss F1'
13 end if
14 // generate the second solution F2'
15 Copy the original solution Fi into F2
16 for (i= network_flow_size/2 to network_flow_size)
17   F2'= Randomly generate new flow
18 end for-loop
19 // objective function for the new molecule
20 PE2'=objectiveFunction(F2')
21 //confirm the new solution or dismiss
22 if (PE2'> PEi ) then // solution confirmed
23   destroy Fi
24   return F2'
25 else
26   dismiss F2'
27 end if
```

Fig. 5. Pseudo-code for Decomposition function

VI. ANALYSIS OF MAXFLOW-CRO ALGORITHM

A. Time complexity

For initial max flow of the first parent generation, the run time complexity can be approximated by $O(N E f)$, where N is the number of parent nodes, E is the number of edges in flow network and f is the maximum flow in the flow network graph. The Random Number between 0-1 is approximated by $O(1)$; which can be ignored since it has a constant value.

Starting the optimization algorithm: the run time complexity can be approximated by $O(I X)$, where I is the number of iterations and X is the complexity for each method (depends on random selection). Below is the complexity calculation for each reaction, so we will be able to find the worst case.

- 1) Decomposition function: $O(E^2)$, where E is number of edges in the graph.
- 2) On-Wall ineffective collision function: $O(E^2)$, where E is number of edges in the graph.
- 3) Synthesis function: $O(C)$, where C: constant, synthesis function contains If and assignment statements, see Figure6.

4) Inter-Molecular function: $O(E^2+E+E)=O(E^2)$, where E is number of edges in the graph.

The worst case is when randomization leads to $O(E^2)$, so replace X with $O(E^2)$. Therefore the time complexity for the optimization algorithm is $O(I E^2)$.

Max flow calculation for the results: This is similar to the initial $O(R E f)$, where R is the remaining items in the input array, E is the number of edges in the graph, f is the maximum flow in the graph. Thus, the final Time complexity will be: $O(N E f) + O(I E^2) = O(I E^2)$.

B. Space complexity:

The initial input needs $O(N E^2)$, where E is the number of edges in the graph, and N is the number of nodes. For the four methods in the optimization algorithm:

- 1) Intermolecular function: $O(2E^2) = O(E^2)$.
- 2) On-Wall ineffective collision function: $O(E^2)$.
- 3) Synthesis function: $O(1)$, the size of the used graph already calculated in input so no need to include it again.
- 4) Decomposition function: $O(2E^2)$. Additional graph for swapping is needed, so it is $O(E^2)$. The worst case will be $O(2E^2) + O(E^2) = O(E^2)$.

Function: synthesis()

Input: F_1, F_2, PE_1, PE_2 // two molecules

Output: F_3 : solution has greater value of flow

- ```

1 If (PE1>PE2) then
2 F3 = F1
3 destroy F2
4 else
5 F3 = F2
6 destroy F1
7 return F3

```

Fig. 6. Pseudo-code for Synthesis function

Therefore, the final space complexity will be:  $O(N E^2) + O(E^2) = O(N E^2)$ .

## VII. EXAMPLE FOR MAXFLOW-CRO ALGORITHM

To understand how this study uses CRO to optimize maximum flow problem, refer to TABLE1 again. It shows a profile for a directed graph of four nodes and five edges, each edge has its own directed capacity, which will be used in this example and called molecule. The goal from this capacity matrix is to use it as a basis for producing flow matrices within their edges capacity. This example represents the capacity and the possible flow on the edge as flow/ capacity, as shown in Figures 7 through 10.

### A. On-wall ineffective reaction

On-wall ineffective reaction uses total-half change method [4]. First, take the molecule that picked randomly from the population of parents, as one shown in Figure 7. Then, regenerate the first half part of its flows. After that, objective function will be applied to compute the new flow. This max

flow value will be compared with the original max flow value. If it is larger than before, the new value will be confirmed; else it will be dismissed and the parent molecule will be remained. Figure7 shows that the new molecule has max flow value better than that for original one, so it will be confirmed.

### B. Decomposition reaction

In decomposition reaction, the molecule will be copied into other two molecules. For each of these resulted molecules, half values will be selected and regenerated randomly. For the first copy, select the first half, while second half is selected for the second copy. So, a new two molecules will be produced. After that, objective function will be applied, as shown in Figure8. This operation is called total-half change operator [4].

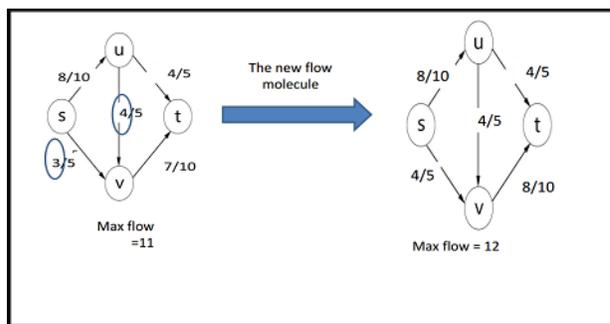


Fig. 7. On-wall ineffective collision reaction example

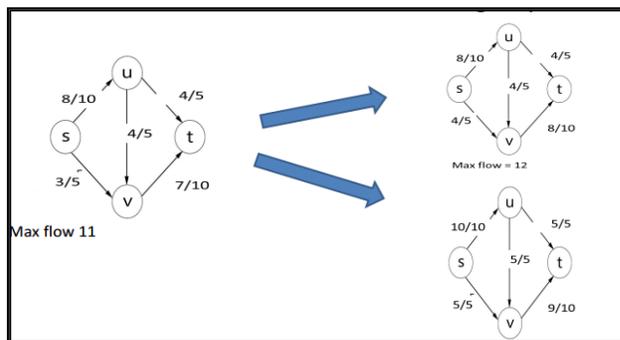


Fig. 8. Decomposition reaction example

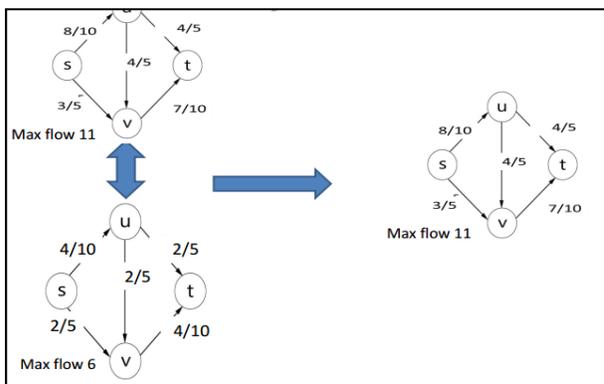


Fig. 9. Synthesis reaction example

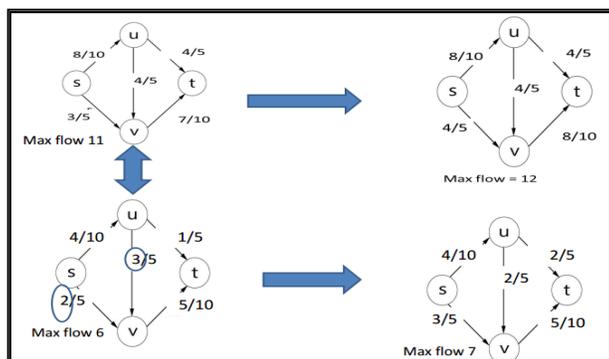


Fig. 10. Inter-molecular reaction example

C. Synthesis reaction

In the synthesis reaction, simply select the molecule which has greater flow value than the other molecule, as shown in Figure 9.

D. Inter-molecular reaction

Inter-molecular reaction is the same as on-wall reaction, but the procedure will be applied on two molecules rather than one, as shown in Figure 10.

VIII. RESULTS

The MaxFlow-CRO program was implemented and executed using a dataset of different flow\_network\_size: number of nodes in the graph. These dataset sizes are ranging from 50 to 1000 nodes as shown in TABLE2. Datasets with further sizes were unable to be tested on the test PC due to memory limitations. The algorithm was tested using Intel-core™ i5-2450M CPU with 2.50 GHz, 4 GB RAM (2.5 GB usable), and windows7 32-bit operating system. The application program is written using Java and executed in Net-Beans IDE 7.1.2, using ten structured classes. Each dataset is experimented 10 times, runtime in seconds is recorded, and an average runtime is calculated. Figure11 shows the experimental runtimes depicted from TABLE2. Figure11 represents how execution time behaves with increasing size of the graph. From this Figure, we can conclude that the algorithm has a quadratic polynomial time complexity, which is possibly a good performance. The experiments show that, as initial parent size and number of iterations increase, better results for max flow will be obtained. Runtimes in TABLE2 are recorded when the highest possible value for max flow is reached by the application program.

TABLE II. RUN TIMES (IN SECONDS) FOR DIFFERENT NETWORK SIZES

| Network Size | Average runtimes(sec) | Network Size | Average runtimes(sec) |
|--------------|-----------------------|--------------|-----------------------|
| 50           | 0.109                 | 550          | 114.198               |
| 100          | 0.625                 | 600          | 149.497               |
| 150          | 2.429                 | 650          | 178.372               |
| 200          | 4.449                 | 700          | 230.718               |
| 250          | 11.278                | 750          | 278.584               |
| 300          | 18.694                | 800          | 301.045               |
| 350          | 24.45                 | 850          | 396.915               |
| 400          | 43.233                | 900          | 425.62                |
| 450          | 54.37                 | 950          | 459.185               |
| 500          | 80.142                | 1000         | 506.103               |

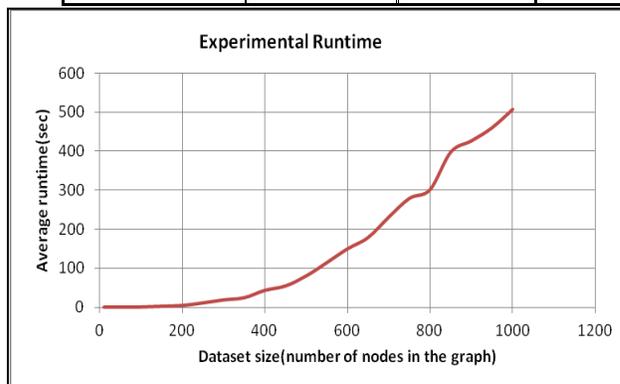


Fig. 11. Runtime graph for experimental results

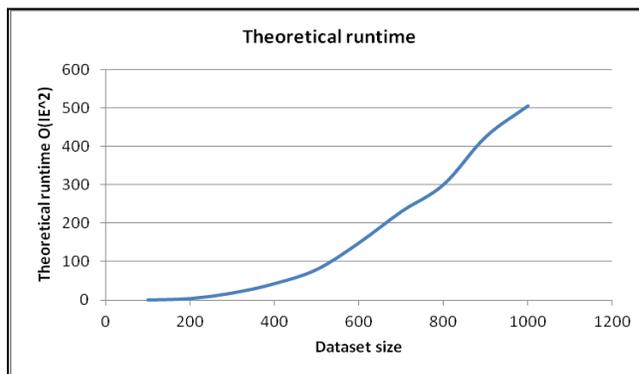


Fig. 12. Theoretical runtime graph for  $O(IE^2)$  complexity

Figure 12 shows the chart for the asymptotic notation ( $IE^2$ ), such that  $I = 75$  and  $E = 100$ . It is clear from both Figure11 and Figure12 that experimental and theoretical results converge. Many terms are removed from the asymptotic notation of the runtime complexity when calculated theoretically, and that explains the slight difference in shape between the two Figures.

## IX. CONCLUSION AND FUTURE WORK

This study proposes a potential solution to maximum flow problem through using chemical reaction optimization algorithm. The proposed MaxFlow-CRO algorithm is presented, analyzed asymptotically and experimental test is conducted. Asymptotically, the algorithm runtime is  $O(I E^2)$ . Asymptotic runtime is proved theoretically. The experiments show that, as initial value of parent\_size variable and numbers of iterations are increased, better results for max flow will be obtained. Initial parent\_size variable didn't affect the run time, because initialization operations, in general, are out of runtimes consideration. In other hand, iterations number is an important factor that can affect the value of max flow and run time duration. As a future work, the algorithm could be improved to reach the possible highest max-flow value using less number of iterations by implementing the algorithm on supercomputer to evaluate its performance in parallel. In addition, we can conduct a comparison between this proposed algorithm and other heuristic, meta-heuristic or evolutionary algorithms used to solve maximum flow problem in terms of their performance.

### REFERENCES

- [1] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, 3rd ed., The MIT Press, 2009.
- [2] Munakata, T. and Hashier, D.J. "A genetic algorithm applied to the maximum flow problem", Proc. 5th Int. Conf. Genetic Algorithms, 1993, pp. 488-493
- [3] Kuipers, S. Yang, and S. Trajanovski, "Constrained Maximum Flow in Stochastic Networks", IEEE 22nd International Conference on Network Protocols, 2014, pp. 397-408.
- [4] A.Y.S. Lam, V.O.K. Li, "Chemical reaction optimization: a tutorial", Memetic Computing 4, 2012, pp. 3-17.
- [5] Ford jr., L.R., Fulkerson, D.R., Flows in networks. Princeton University Press, Princeton, 1962
- [6] Ford jr., L.R., Fulkerson, D.R., "Maximal flow through a network". Can. J. Math. 8(3), 1956, pp. 399-404.
- [7] Dinic, E.A., "Algorithm for solution of a problem of maximum flow in networks with power estimation". Sov. Math. Doklady. 11(8), 1970, pp. 1277-1280.
- [8] Edmonds, J., Karp, R.M., "Theoretical improvements in algorithmic efficiency for network flow problems". J. Assoc. Comput. Machinery 19(2), 1972, pp. 248-264.
- [9] Goldberg, A.V., "A new max-flow algorithm". Technical Report MIT/LCS/ TM-291, Laboratory for Computer Science, M.I.T, 1985.
- [10] Goldberg, A.V., Tarjan, R.E., "A new approach to the maximum flow problem". Proc. 18th ACM STOC, 1986, pp. 136-146.
- [11] Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization", IEEE Trans Evol Comput vol. 14, no. 3, 2010, pp. 381- 399.
- [12] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin., Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.
- [13] Sheta, A. F. "Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects". Journal of Computer Science 2 (2), 2006, pp. 118-123.
- [14] J. B. Orlin. Max flows in  $o(nm)$  time, or better. In STOC'13: Proceedings of the 45th Annual ACM Symposium on the Theory of Computing, 2013, pp.765-774.
- [15] V. King, S. Rao, and R. Tarjan, "A faster deterministic maximum flow algorithm", In Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, 1992, pp. 157-164.