A Comparison between Chemical Reaction Optimization and Genetic Algorithms for Max Flow Problem

Mohammad Y. Khanafseh
King Abdulla II School for Information and Technology
University of Jordan
Amman-Jordan

Ahmad Sharieh
King Abdulla II School for Information and Technology
University of Jordan
Amman-Jordan

Ola M. Surakhi
King Abdulla II School for Information and Technology
University of Jordan
Amman-Jordan

Azzam Sleit
King Abdulla II School for Information and Technology
University of Jordan
Amman-Jordan

***Abstract*—This paper presents a comparison between the performance of Chemical Reaction Optimization algorithm and Genetic algorithm in solving maximum flow problem with the performance of Ford-Fulkerson algorithm in that. The algorithms have been implemented sequentially using JAVA programming language, and executed to find maximum flow problem using different network size. Ford-Fulkerson algorithm which is based on the idea of finding augmenting path is the most popular algorithm used to find maximum flow value but its time complexity is high. The main aim of this study is to determine which algorithm will give results closer to the Ford-Fulkerson results in less time and with the same degree of accuracy. The results showed that both algorithms can solve Max Flow problem with accuracy results close to Ford Fulkerson results, with a better performance achieved when using the genetic algorithm in term of time and accuracy**.

Keywords—Chemical reaction optimization; Ford-Fulkerson algorithm; genetic algorithm; maximum flow problem

I. Introduction

A flow network is a weighted directed graph where each edge has a capacity and receives a flow [17]. The amount of flow on an edge cannot exceed the capacity of the edge. A flow must satisfy the restriction that the amount of flow into a node equals the amount of flow out of it, except when it is a source or sink. The maximum flow problem is to determine an optimal solution for the directed graph by finding the maximum flow from the source to the sink node [17].

Flow network can represent many real-life situations like a traffic in a road system, fluids in pipes, currents in an electrical circuit, or anything similar in which something travels through a network of nodes [15]. Due to its importance in many areas of applications such as computer science, engineering and operations research, the maximum flow problem has been extensively studied by many researchers using a variety of methods [14], [15]. They include: a classic approach [8], maximal flow problem in layered network [3], the shortest augmenting path algorithm [10], and more [2], [4]-[6], [9], [11]-[13].

In this study, Chemical Reaction Optimization (CRO) algorithm and Genetic algorithm (GA) will be implemented and tested on the maximum flow problem. The goal is to determine which algorithm could give a better performance on finding a solution to the maximum flow problem near to the Ford-Fulkerson (FF) solution with less running time duration and same accuracy.

The rest of paper is organized as follows: Section 1 introduces the maximum flow problem. Section 2 presents some related works. Sections 3 and 4 explains the review the CRO and Genetic algorithms, respectively for solving the maximum flow problem. Section 5 shows the experimental and comparison results and Section 6 presents the conclusion and future works.

II. Related Works

A. Maximum Flow Problem

The flow network is a directed graph with two special vertices; the source and the sink [17]. Each edge in the graph connect two verticies and has a capacity and receives a flow that should be less than or equal to its capacity. In the operation research, a directed graph is called a network, the vertices are called nodes and the edges are called arcs [17].

A network is a directed graph $G = (V, E)$, with two special kinds of vertices are distinguished: a source S and a sink T, and every edge e = (u,v) ∈ E has a non-negative, real-valued capacity c(u,v). A flow network is an integer valued function f defined on the edges of G and satisfying that $0 \leq f(u,v) \leq c(u,v)$, for every Edge (u,v) in E.

For each edge (u,v) in E, the flow f(u,v) is a real valued function that must satisfy the following three properties for all nodes u and v:

1) Capacity constraints: $f(u,v) \leq c(u,v)$. The flow along an edge cannot exceed its capacity.

2) Skew symmetry: $f(u,v) = -f(v,u)$. The flow from u to v must be the opposite of the net flow from v to u.

3) Flow conservation: $\sum_{v\in V} f(s, v) = 0$,

unless u = s or u = t. The flow to a node is zero, except for the source, which “produces” flow, and the sink, which “consumes” flow.

To achieve flow conservation, the flow into the node should be equal to the flow going out from the node. Also, the total amount of flow going from source s equals total amount of flow going into the sink t. The value of the flow is given by (1):

$$|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t) \qquad (1)$$

An example of the flow network with a source node s, sink node t and four additional nodes is shown in Fig. 1. The flow and the capacity is denoted by f/c. The network upholds skew symmetry and capacity constraints. The total amount of flow from s is 5, which is also the incoming flow to t.

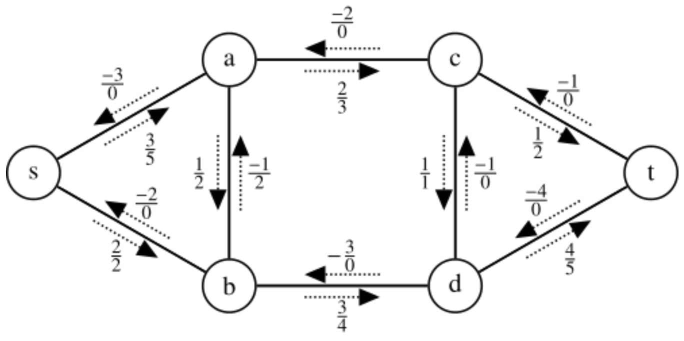

Fig. 1. A flow network with the flow and capacity.

The maximum flow problem involves finding a maximum flow through a single-source, single-sink flow network.

B. Ford-Fulkersom Algorithm

The Ford-Fulkerson method [1] (named for L. R. Ford, Jr. and D. R. Fulkerson) is the most popular algorithm used to computes the maximum flow in a flow network. The main idea of the algorithm is to find an augmenting path from the source to the sink with available capacity on all edges in the path to send flow along it. While there exist an augmenting path, you send a flow along it.

The Ford-Fulkerson algorithm has two main steps as shown in Fig. 2. The first is a labeling process that searches for a flow augmenting path i.e., a path from the source s to the sink t where the flow is less than the capacity along all forward arcs and the flow > 0 along all backward arcs. If this step finds a flow augmenting path, the second step changes the flow accordingly. Otherwise, no augmenting path exists then you get the maximum flow.

The runtime of Ford–Fulkerson is bounded by O(Ef), where E is the number of edges in the graph and f is the maximum flow in the graph. We run a loop as long as there exists an augmenting path, each iteration of the loop takes O(E) time to find an augmenting path, and increases the flow by at least 1 and un upper bound f, so the time complexity of the algorithm might not be a polynomial.

To decrease the computational time and get a better performance, many researches gave different algorithms.

```
Ford-Fulkerson algorithm:
initialize flow to 0
path = find Augmenting Path(G, s, t)
while path exists:
augment flow along path
G_f = create Residual Graph()
path = find Augmenting Path(G_f, s, t)
return flow
end algorithm
```

Fig. 2. Ford-Fulkerson algorithm.

Because of the importance of the maximum flow problem in many applications such as computer science, engineering researches, it has been extensively studied by many researchers using a variety of methods and techniques. A recent research in [15] applied to solve maximum flow problem using Chemical Reaction Optimization algorithm. The results showed a better performance with a complexity of O(I E2), for I iterations and E edges. Genetic algorithm was also used to solve maximum flow problem in [13]. The algorithm was implemented sequentially, and the fitness function is defined to reflect two characteristics: balancing vertices and the saturation rate of the flow. The performance of the algorithm depends on the population size and the number of generations needed to find the solution. In order to reduce running time of the algorithm, a parallel implementation was proposed in [18], the results showed a good enhancement in terms of the running time and system performance.

III. Chemical Reaction Optimization

Chemical reaction optimization (CRO), proposed in [6], is a chemical-reaction-inspired general-propose meta-heuristic established for optimization and inspired by the nature of chemical reactions.

CRO refers to multi-agent algorithm which consists of different molecules, where each molecule has different attributes. Some of these attributes are important to CRO operations like molecular structure, kinetic energy (KE) and potential energy (PE) which refers to flow at the graph.

There are four main elementary reactions in CRO operation that take place at the CRO iteration, and are employed to manipulate the solution and distribute the energy through the molecule.

The molecule, here, can be described as container where these molecules are interacting with each other through this container with different forms as follows:

On-wall effective collision which refers to situation when different molecules collide with the wall of the container that contains different molecules. This collide converts the structure of molecule when collision happens new structure like this $\omega \rightarrow \omega'$.

- Decomposition interaction, this refers to situation when a molecule was collided with the wall of the container and then a molecule was divided into two parts $\omega \rightarrow \omega 1 + \omega 2$.
- Inter-molecular ineffective collision: this situation of collision between molecules happens when two molecules collide with each other and they bounce away, like this example when there are $\omega 1$ and $\omega 2$ where both collide with each other, then two new molecules $\omega' 1$ and $\omega' 2$ were produced from those two molecules which interact or collide with each other. This can be presented as: $\omega 1+ \omega 2 \rightarrow \omega' 1 + \omega' 2$.
- Synthesis: This makes an opposite of decomposition. Through this kind of interaction between molecules, two molecules hit with each other to produce new molecule. It can be implemented as $\omega 1 + \omega 2 \rightarrow \omega'$.

The CRO can be implemented to solve Maxflow problem. This needs to explore search space and to generate number of solutions and molecules to achieve optimal solution. Different solutions will be happening due to reaction between different selected molecules. Some of these solutions are near to desired solution and others were far away from it. After a selected number of iterations, the best solution will be taken from the list of these generated solutions.

In this paper, the CRO was applied to generate a possible solution for the Maxflow problem.

A. *Cro-Maxflow Algorithm*

The CRO-Maxflow implementation has three main phases, initialization, iteration and final phase.

- The initialization phase. In this phase, we define the graph as a source, sink node and a number of graph nodes. The nodes on graph are connected by edges where each edge has a weight value or capacity. From the source to the sink node, there are different flows that can be found, these flows refer to parent size and number of generated parent which depends on the value of parent size that had been specified through this step.

The first population will make the reaction with each other or with the wall of the container to generate other molecule or populations.

Some other basic CRO parameters like KE and molecule random number used as stopping criteria beside the use of the number of iteration that had been defined. The Maxflow value can be found in CRO using objective function, which can be computed using shortest augmenting path from source to sink. This value determines the Maxflow value which will be improved by the number of iterations. The objective function was used here as potential energy, other values were defined in the initialization step, such as α which refers to decomposition threshold and β which refers to synthesis threshold.

- The iteration step, the goal of this step is to improve solution or objective function value. Most of the heuristic algorithms depend on the number of iteration to get a better solution. Through iteration step, potential energy or objective function was calculated for each iteration until reaching to iteration number, which was specified at previous step. Other collision happens based on the value of β which refers to the value generated randomly. This value is compared with molecule value. If β value is greater than molecule value, then one parent will be selected. Parent selection is important to know what kind of collision will happen when one parent is selected and this will give the ability for the decomposition reaction or on-wall-effective collision to occur; otherwise the other type of collision will occur.
- After selecting different molecule and calculating Potential energy for different iteration and the number of iterations reach max, the last step will start that refers to selection step. Through this step molecules with best value or largest value for Potential energy will be selected, this value present Maxflow result for the graph.

The pseudo code for the CRO-Maxflow algortithm is shown in Fig. 3 [15].

```
CRO-Maxflow algorithm:
Initialization phase
Set flow_network_size, C[i][j]: maximum capacity
ParentSize, iterationNumber,s: source node, t:
Sink node
HIT= 0
B =parentSize/2
A =parentSize/2
KE = parentSize/ 1.5
Generate molecule ϵ [0,1]
parentGeneratins (C[i][j], parentSize)
for (int i=1 to iterationNumber ) //iteration phase
      Generate b ϵ [0,1]
      If b> Molecule   then
      Randomly select one parent
      If (HIT > α) then
          Decomposition()
      Else
         OnWallIneffectiveCollision ()
      End if

      Else
      Randomly select two molecules
      If (KE <=β && parentSize >=2) then
      Synthesis ()
      Else if (parentSize >=2)
          IntermolecularIneffectiveCollision ()
       End if
       End if
    HIT ++
    KE - -
    Check for any new maximum solution
    End for-loop //final stage
    Return best solution found
    End algorithm
```

Fig. 3. Pseudo-code for CRO-Maxflow algorithm.

IV. Genetic Algorithm

A genetic algorithm (GA) is a method for solving complex optimization problems based on a natural selection process that mimics biological evolution. It can be used to design computer algorithms, to schedule tasks, and to solve other optimization problems.

Population can be viewed as binary bit strings. The initial values of this population are usually randomly generated and evaluated. The relation between the combination of ones or zeros in the population is found by an evaluation function that return a 'fitness' value for some bit string [7].

The three main operations of the genetic algorithm are: Reproduction (or Selection), Crossover and Mutation.

- **Reproduction:** use a fitness value to selects the best individuals and discards the bad ones from the population. The best individuals are those having more chances to survive in the next generation.
- **Crossover**: includes two steps. First, pairs of bit strings will be mated randomly to become the parents of two new bit strings. The second part consists of choosing a place (crossover site) in the bit string and exchanges all characters of the parents after that point. The process tries to artificially reproduce the mating process where the DNA of two parents determines the DNA for the newly born.
- **Mutation**: changes a 0 for a 1 and vice versa for the bits that can't be changed by the previous operations due to its absence from the generation, either by a random chance or because it has been discarded.
- Repeat the above steps until reaching the termination condition. The pseudo code of the Genetic algorithm is shown in Fig. 4 [16].

```
1.  Genetic algorithm:
2.  Initialize population
3.  Evaluate population
4.  While (!stopCondition) do
5.      Select the best-fit individuals for reproduction
6.      Breed new individuals through crossover and mutation
    operations
7.      Evaluate the individuals fitness of new individuals
8.      Replace least-fit population with new individuals
9.  End algorithm
```

Fig. 4. Pseudo-code of Genetic algorithm [16].

A sequential implementation for Genetic algorithm has been applied to solve max flow optimization problems [14]. The number of iterations have been determined according to previous GA applications to achieve optimal or near optimal solutions.

This study aims to apply a sequential version of genetic algorithm on the max flow problem again. In order to get better results and performance, one of the possible solutions here is to reduce the number of generations required to determine the solutions.

A. *Genetic-Maxflow ALGORITHM*

Maxflow problem consist of graph with number of nodes and edges between these nodes. Each edge has specific weight or capacity which is saved in matrix called C[i,j]. Based on this value, different optimistic path was selected for each iteration and solution will be build based on this paths between source and sink. These solutions in the Genetic algorithm refer to population and will be saved in population matrix.

For a graph, G, with n vertices and m edges; G is represented by the flow capacity matrix, $C = [c_{ij}]$, i, j = 1, n. Each solution is represented by a flow matrix $F = [f_{ij}]$, i, j = 1, n. The initial flow was generated randomly.

The first step of Genetic-Maxflow refers to initialization step. In this step, number of iteration, population size and mutation ratio values must be defined. After this step, the selection step must be implemented to select best solution from different solutions. This is implemented based on the fitness function to find path between source and sink. Based on the value of Fitness Function, it will select some ratio for all matrix. Next the crossover will be implemented between different solutions. The first population or solution have best Maxflow result, then generate a new population in new matrix Pop1 with good results for Maxflow. After that step, mutation will be implemented to change some of solution to different one from parent solution which depends on doing a crossover process. This mutation based on specific ration. This ration must be small between 0.01 and 0.025 of all population to change population. New solutions for Maxflow problem was generated in new Matrix Pop2. This solution for one iteration. Genetic algorithm is heuristic algorithm where number of iterations was important to achieve enhancement of solution at each iteration. These steps were repeated at each iteration in the proposed solution and best population will be selected based on population size which specified at the beginning of the algorithm.

V. Experimental Results and Comparisons

In this study, a sequential implementation for FF, CRO and GA is applied to find maximum flow problem with a different network size. The algorithms FF, CRO and GA were tested using Intel core I7-3632QM CPU2.20GHz, 8GB of RAM and windows 7 64 bits. The application programs were written using java and executed on Net-Beans IDE 8.1. The implementations were done over different network size started from 50 nodes until 6030 nodes, with different number of parents and different number of iterations, in order to achieve the best solution which is near to Ford-Fulkerson one, as we will see from results. Each experiment was repeated 10 times, and the average results were calculated.

The first comparison was made between the Ford Fulkerson algorithm and the CRO algorithm on the Maxflow problem based on the time needs to calculate Maxflow and the accuracy level of the proposed solution for a graph when using different number of nodes, and different number of iterations. Each experiment was executed for 10 times and the average result was calculated.

Different results for CRO-Maxflow were conducted. These results were optimized until reaching to accuracy level near to the Ford-Fulkerson with less time than the Ford-Fulkerson algorithm running time and using different network size. A part of comparison between the program results with optimal Ford Fulkerson results are shown in Table 1. Runtime in Table 1 is in milliseconds. Table 1 shows the results for the implementation of CRO-Maxflow and Ford Fulkerson Maxflow.

The number of nodes plays an important role in Ford-Fulkerson solution. The CRO algorithm took less time to find maxflow than Ford-Fulkerson for large number of nodes. The four main steps for CRO were repeated based on the number of iterations needed to reach to a solution near the Ford-Fulkerson according to the accuracy when finding Maxflow. The number of iterations have some limitations when increasing it to be more than some specific value it will consumes the memory efficiency.

TABLE. I. RESULTS FOR IMPLEMENTING CRO-MAXFLOW AND FF MAXFLOW, WITH NUMBER OF NODES= 50 TO 6030

Size	CRO Time	FF Time	FF result	CRO result	Quality
3550	8.3	14.3	14.2	9.7	0.683099
3602	14.3	15.1	13.4	13.4	1
3654	9.2	15.2	14.7	13.2	0.897959
3706	6.3	15.7	37.2	33.2	0.892473
3758	13.9	16.3	22.5	22.3	0.991111
3810	16.5	16.5	24.3	19.7	0.8107
3862	11.5	17	22.8	21.8	0.95614
3914	17.1	17.5	17.1	14.7	0.859649
3966	15.4	18.1	22.8	16	0.701754
4018	10.7	18.3	17.4	16.7	0.95977
4070	8.2	19	19.1	15.4	0.806283
4122	19	20.1	30.6	27.5	0.898693
4226	22.6	20.1	22.9	22.5	0.982533
4278	15.1	20.8	22.2	19.4	0.873874
4330	20.3	21.3	16.5	15.3	0.927273
4486	13.5	22.9	23.2	14.4	0.62069
4590	20.6	24.1	13.6	12.6	0.926471
4746	25.5	25.6	20.9	14.3	0.684211
5162	11.7	29.9	16.4	12.4	0.756098
5214	26.4	30.6	11.6	6.7	0.577586
5422	20.2	33.4	22.2	18.8	0.846847
5474	13	34.3	20.3	20.3	1
5942	31.2	39.8	19.6	18.6	0.94898
5990	18.8	40.6	18.8	18.4	0.978723
6098	25.5	41.7	21.3	19	0.892019
6306	46	49	29	28.5	0.982759

From the result of 1000 nodes and 10 iterations, we can say that the CRO-Maxflow gives less time for execution than Optimal of Ford-Fulkerson Maxflow with accuracy rate near to 90% to Ford-Fulkerson results. This gives a good enhancement when use a heuristic algorithm to solve Maxflow problem with less time and same level of accuracy, which we implemented through the proposed solution. To compare results for Maxflow problem using CRO and Ford-Fulkerson solution, Fig. 5 presents the relation between time needed for CRO and Ford-Fulkerson Solution at same data sets.

Fig. 5 shows that run time needed to solve Maxflow using CRO is less than that needed to solve the same problem on the same dataset and using same machine and environment for optimal Ford Fulkerson algorithm.

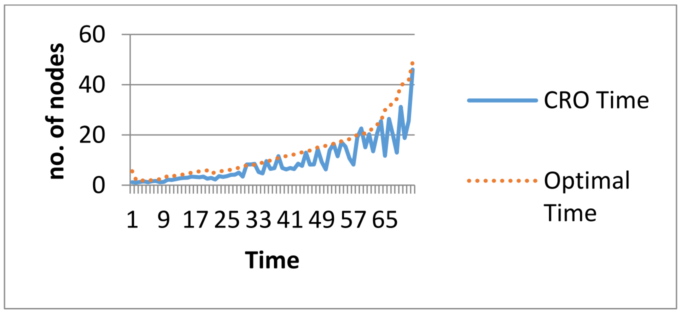

Fig. 5. Execution time of CRO and FF algorithm to solve max flow problem.

Fig. 6 shows the accuracy of CRO-Maxflow problem and Ford-Fulkerson Maxflow. This results for different number of nodes, started from 50 nodes to 1000 nodes. The proposed solution to solve Maxflow has accuracy near to Ford-Fulkerson accuracy, which is a good achievement.

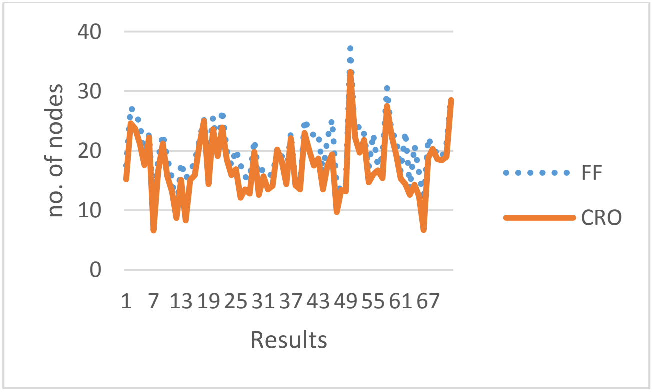

Fig. 6. Accuracy results for calculating Maxflow using CRO and FF.

The second comparison was between Genetic-Maxflow and FF Maxflow.

In [14], the authors applied GA to find the maximum flow from the source to sink in a weighted directed graph. The experiment was run for various graphs with different number of vertices. The results showed that Genetic algorithm found an optimal or near optimal solution for the maximum flow problem, with a reasonable number of iterations compared to other previous GA applications [14].

TABLE. II. RESULTS FOR IMPLEMENTING GENETIC-MAXFLOW AND FF MAXFLOW, FOR DIFFERENT NUMBER OF NODES STARTED FROM 50 NOD UNTIL 1986

Size	**GA Time**	**FF Time**	**FF result**	**GA result**
4590	12.9	23.5	13.6	12.6
4642	12.8	23.8	20.2	18.7
4694	13.5	24.4	15.7	15.6
4746	12.9	24.7	20.9	20.7
4798	13.8	25.4	19.3	17.5
4850	14.8	26.9	24.6	20.6
4902	15.3	26.8	17.1	14
4954	14.7	27.2	14.4	13.1
5006	14.3	27.7	27.4	27.4
5058	14.7	28.2	17.9	16.8
5110	14.6	28.4	17.1	16.3
5162	14.5	29.4	16.4	15.1
5214	14.8	29.9	11.6	8.1
5266	14.7	30.3	19.1	18.9
5318	14.1	30.8	26.1	25.5
5370	16.1	31.7	19.3	14.8
5422	16.9	32.6	22.2	20.5
5474	17.2	33.1	20.3	19.9
5526	17.9	33.8	20.9	17.8
5578	17.2	34.2	28.1	26
5630	17.6	34.7	22	22
5682	17.5	35.3	15.7	15.3
5734	17.7	36.2	16.6	14
5786	18.3	37.9	12.7	11.7
5838	18.1	37.7	14.6	13.3
5890	19.7	38	12.7	10.7
5942	18.3	38.9	19.6	19.6
5994	18.9	39.4	22	12.4
6046	20.5	39.8	21	20
6098	19.5	40.7	21.3	19.7
6150	21.7	41.6	20.3	18.2
6202	20.2	41.7	25.1	20.9
6254	21	42.6	19.9	18.4
6306	21.4	43.5	29	26.9
6358	20.9	44	19.7	19.3

We implemented the GA to solve Maxflow problem and compare results. The implementation of the three algorithms FF, CRO and GA were tested on the same data and the same environment. Both objective functions which we used to calculate Maxflow in CRO, are the same as Fitness functions which we used in GA.

When implementing GA on Maxflow problem, the population is selected first then we start with genetic steps from selection of best population of all populations to cross over process. Through this process, we did cross over for two selected solutions from selection process then we implemented the mutation step in order to make changes for the generated population. After the cross over step, the mutation step was checked to make sure it does not exceed the range from 0.01- 0.025 for all populations. The process was done randomly.

These steps of GA were repeated based on the number of iterations, which specified to reach solutions near to FF one in accuracy level to find Maxflow. But generating number has some limitation. If you increase this number more than specific value, it will affect the memory space.

The GA was implemented at the same environment that used to implement CRO-Maxflow. We compared results of GA with results of optimal FF for the same experiment. Each experiment with specific size of network was repeated 10 times and average result for this repeated time was calculated with specific number of iterations similar to process which is done in CRO-Maxflow experiment. Table 2 shows part of the results for the time needed for Genetic-Maxflow less than time which needs to solve Maxflow problem using FF solution, at level of accuracy near to FF level.

The results show that GA reach to accuracy near to FF with time less than time needs to solve maxflow by FF algorithm as shown in Fig. 7 and 8, respectively.

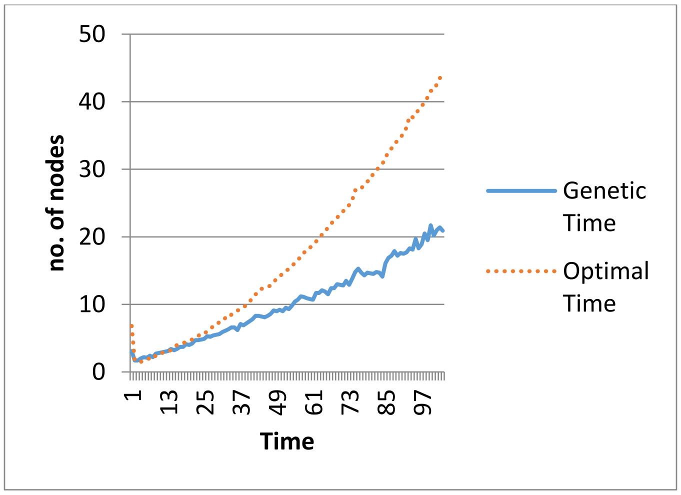

Fig. 7. Relation between time needs for Maxflow Problem using GA and FF, at same node.

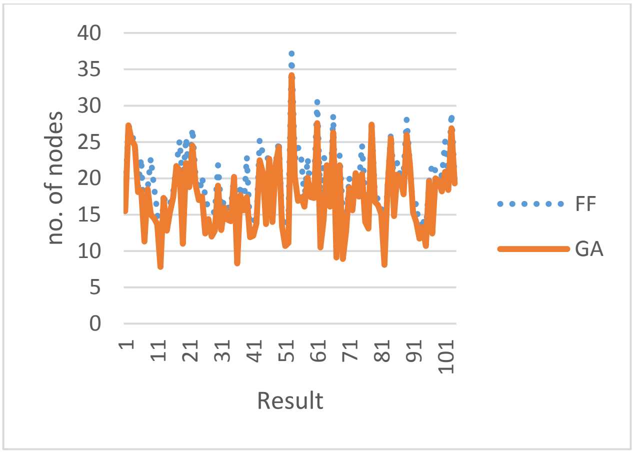

Fig. 8. Accuracy results for calculate Maxflow using GA and FF.

The results from using CRO-Maxflow were compared with the Genetic-Maxflow as shown in Table 3 and Fig. 9.

TABLE. III. CRO-MAXFLOW VS GENETIC-MAXFLOW

Size	CRO Time	FF Time	GA Time	CRO Result	GA Result
3550	8.3	14.3	14.7	9.7	16.8
3602	14.3	15.1	14.6	13.4	16.3
3654	9.2	15.2	14.5	13.2	15.1
3706	6.3	15.7	14.8	33.2	8.1
3758	13.9	16.3	14.7	22.3	18.9
3810	16.5	16.5	14.1	19.7	25.5
3862	11.5	17	16.1	21.8	14.8
3914	17.1	17.5	16.9	14.7	20.5
3966	15.4	18.1	17.2	16	19.9
4018	10.7	18.3	17.9	16.7	17.8
4070	8.2	19	17.2	15.4	26
4122	19	20.1	17.6	27.5	22
4226	22.6	20.1	17.5	22.5	15.3
4278	15.1	20.8	17.7	19.4	14
4330	20.3	21.3	18.3	15.3	11.7
4486	13.5	22.9	18.1	14.4	13.3
4590	20.6	24.1	19.7	12.6	10.7
4746	25.5	25.6	18.3	14.3	19.6
5162	11.7	29.9	18.9	12.4	12.4
5214	26.4	30.6	20.5	6.7	20
5422	20.2	33.4	19.5	18.8	19.7
5474	13	34.3	21.7	20.3	18.2
5942	31.2	39.8	20.2	18.6	20.9
5990	18.8	40.6	21	18.4	18.4
6098	25.5	41.7	21.4	19	26.9
6306	46	49	20.9	28.5	19.3

The results show that GA took less time with same level of accuracy as CRO algorithm for the same network size and same number of iterations. As the performance of the GA depends on doing the cross over and mutation steps by each iteration. While CRO algorithm depends on different number of collisions that can be happened between different molecules which compared with molecule value to determine number of molecules that will be selected on interaction or collision process. This will need more time to achieve it. Through the experiment of implementation, both algorithms were implemented using java programming language. Based on the results for each step, calculate the time needs for each step for both experiments GA and CRO to decide which step consume most of the time. According to that optimize that step which spent most of execution time in CRO and GA to achieve same level of enhancement on both algorithms.

The copy matrix step consumes most of the time, some optimization steps was done to enhance time results like allocation and de allocation for matrix when one matrix for a graph was deleted, new matrix for a new graph was build and this consumes time and memory through the implementation, because of that we reuse the matrix by using a java object pool feature that allow to use the same matrix with replacing the nodes for the old matrix with a new matrix results.

Other technical enhancement which has been done for both the CRO and GA deals with the connected edges. As the matrix presents a graph with nodes and edges, we worked with the submatrix that contains ones instead of dealing with the whole matrix with its connected and disconnected edges.

Both algorithms, CRO and GA gave better execution time than FF algorithm for large network size. When this number becomes very large, the CRO and GA keep on the same level of efficiency in terms of accuracy and velocity.

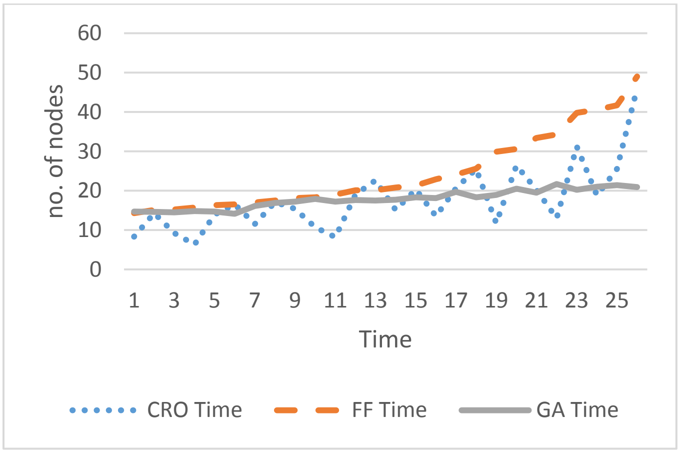

Fig. 9. Relation between time needs for Maxflow problem using CRO, GA and FF, at same node.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, CRO and Genetic algorithms were implemented sequentially on Intel core I7-3632QM CPU2.20GHz, 8GB of RAM and windows 7 64 bits to solve the Maxflow problem. The application programs were written using java and executed on Net-Beans IDE 8.1. The implementations were done over different network size started from 50 nodes until 6030 nodes, with different number of parents and different number of iterations, in order to achieve the best solution which is near to Ford-Fulkerson one.

The results show that GA and CRO both can solve Max Flow problem with accuracy result near to FF results, with better performance achieved when using the genetic algorithm in term of time and accuracy.

For future work, we need to implement both Genetic and CRO on parallel to solve max flow problem by using a super computer to test the amount of enhancement on time with large number of network size.

References

[1] FORD.L.R. AND D. R. FULKERSON 1956. Maximal Flow Through a Network. Can. J. Math. 8,399-404

[2] Edmonds, Jack; Karp, Richard M. *(1972). "Theoretical improvements in algorithmic efficiency for network flow problems". Journal of the ACM.* Association for Computing Machinery. *19 (2): 248–264.* doi:10.1145/321694.321699.

[3] Dinic, E. A. (1970). "Algorithm for solution of a problem of maximum flow in a network with power estimation". Soviet Math. Doklady. Doklady. **11**: 1277–1280.

[4] KARZANOVA.V. 1974. Determining the Maximal Row in a Network by the Method of Preflows. Soviet Math.Dokl.15,434-437.

[5] Baumann N, Skutella M (2006) Solving evacuation problems efficiently–earliest arrival flows with multiple sources. In: 47th annual IEEE symposium on foundations of computer science (FOCS'06), pp 399–410

[6] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," IEEE Trans. Evol. Comput., vol. 14, no. 3, pp. 381– 399, 2010

[7] D. Arjona, "A hybrid artificial neural network/genetic algorithm approach to on-line switching operations for the optimization of electrical power systems", appears in "Energy Conversion Engineering Conference", pp 2286 – 2290, vol.4, Aug 1996.

[8] L.R. Ford, Jr. and D.R. Fulkerson, Flows in Networks, Princeton, NJ: Princeton University Press, 1962

[9] Goldberg, A.V., Tarjan, R.E., "A new approach to the maximum flow problem". Proc. 18th ACM STOC, 1986, pp. 136-146

[10] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin., Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.

[11] G. Mazzoni, S. Pallottino and M.G. Scutella, "The Maximum Flow Problem: A Max-Preflow Approach," European Journal of Operations Research, vol. 53, pp. 257-278, 1991

[12] 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.

[13] 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.

[14] Munakata, T. and Hashier, D.J. "A genetic algorithm applied to the maximum flow problem", Proc. 5thInt. Conf. Genetic Algorithms, 1993, pp. 488-493

[15] R.Barham, A.Sharieh, A.Sliet. "Chemical Reaction Optimization for Max Flow Problem", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 8, 2016

[16] MahmoodA.Rashid, M.A.Hakim Newton,Md. Tamjidul Hoque, Abdul. Sattar," Mixing Energy Models in Genetic Algorithms for On-Lattice Protein Structure Prediction", Hindawi Publishing Corporation Bio Med Research International, Volume2013,ArticleID924137,15pages

[17] Zhipeng Jiang, Xiaodong Hu, and Suixiang Gao, "A Parallel Ford-Fulkerson Algorithm For Maximum Flow Problem", The Allen Institute for Artificial Intelligence, SemanticScholar.

[18] Ola M.Surakhi, Mohammad Qatawneh, Hussein A.. "A Parallel Genetic Algorithm for Maximum Flow Problem", International Journal of Advanced Computer Science and Applications, 2017.