Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Grey wolf optimization applied to the maximum flow problem

Raja Masadeh [1,*], Ahmad Sharieh [2], Azzam Sliet [2]

[1]Software Engineering Department, The World Islamic Science and Education University, Amman, Jordan
[2]Computer Science Department, The University of Jordan, Amman, Jordan

ARTICLE INFO

Article history:
Received 21 March 2017
Received in revised form
5 June 2017
Accepted 10 June 2017

Keywords:
Grey wolf optimization
Maximum flow problem
Meta-heuristic
Optimization

ABSTRACT

The problem of getting the maximum flow from source to destination in networks is investigated in this paper. A proposed algorithm is presented in order to solve Maximum Flow problem by using Grey Wolf Optimization (GWO). The GWO is a recently established meta-heuristics for optimization, inspired by grey wolves (Canis Lupus). In addition; in this current research, K-means clustering algorithm is used to group each 12 vertices with each other at one cluster according to GWO constraint. This work is implemented and tested various datasets between 50 vertices and 1000 vertices. The simulation results show rapprochement between experimental and theoretical results.

1. Introduction

Metaheuristic optimization mechanisms are emerging and becoming very popular, the primary categories of these techniques are Single-Solution-Based and Population-Based. In the first category, the search begins with a single elect solution (Kirkpatrick et al., 1983).This unique elect solution is enhanced over several iterations. In the second category, it represents the optimization by starting with a set of random solutions. This population is enhanced over the course of iterations. The main advantage that characterizes the population-based Metaheuristic over the single-based algorithms is its' high exploration power. This power is attained as it works to find a global solution rather than local ones (Munakata and Hashier, 1993). Maximum Flow problem is considered as one of the various well known basic problems of optimization in weighted directed graphs. In addition, it could be applied to numerous applications such as computer science and engineering. Besides, it is solved by many researchers using several methods (Tarjan, 1983).

The issue of Maximum Flow is to define an optimal solution for a graph which is directed and integer weighted; where the weight at each edge (arc) interconnect two vertices (nodes) representing the flow capacity of this arc. According to these constraints, the goal reaches the maximum of total flow from the source to the sink. This scenario illustrates a simplistic version of the Maximum Flow problem.

A directed flow network G= (V, E) is given, where the number of vertices is represented by V and the number of edges is noted by E. The weight at each edge in the graph represents the capacity C_{uv} which is nonnegative c (u, v)>= 0; where u and v belong to V (Cormen et al., 2009). In addition, there are two special vertices; the source which is start vertex and sink which is the target vertex (Cormen et al., 2009). For a vertex u in V let E (u) be the set of all edges generated from vertex u, and Let F = max {C_{uv} by (u, v) in E}.

Thus, the problem is to define an optimal solution for a particular directed network where each edge has a capacity. Under these constraints, the purpose is to get the maximum total flow from a source vertex to a sink vertex (Eppstein, 1998). Representing the flow on the edge (u, v) in E by X_{uv}.a Max Flow Optimization Model can be obtained through Eq. 1 (Cormen et al., 2009).

$$Max\ f(x) = \sum_{(u,v)\in E(s)}^{n} X_{uv} \quad (1)$$

where

$$\sum_{\{v:(u,v)\in E\}} X_{uv} - \sum_{\{v:(v,u)\in E} X_{vu} = 0; For\ all\ u \in V\{s,t\}$$
$$0 \le X_{uv} \le F_{uv};\ For\ all\ (u,v) \in E.$$

In this study, a proposed solution is presented to solve this problem using Grey Wolf Optimization (GWO). The GWO algorithm was introduced by Mirjalili et al. (2014). This algorithm is inspired by species of wolves (the Grey Wolf). The technique imitates the hunting procedures followed by Grey

* Corresponding Author.
Email Address: raja.masadeh@wise.edu.jo (R. Masadeh)
https://doi.org/10.21833/ijaas.2017.07.014
2313-626X/© 2017 The Authors. Published by IASE.

Wolves in nature. Wolves divide leadership hierarchy in their pack into four main types: Alpha, Beta, Delta, and Omega. In order to accomplish the three main steps of haunting are searching for prey, encircling prey and attacking prey.

The remainder of this paper is organized as follows: section II recites related work in details. While section III contains the Grey Wolf Optimization and how it works. Section IV includes the proposed algorithm "MAXFLOW-GWO". The experimental results are presented in section V. Finally, section VI draws the conclusion and future work.

2. Related works

Maximum Flow problem is a heavily studied combinatorial optimization issues by many researchers using various methods (Tarjan, 1983; McHugh, 1990). Ford and Fulkerson (1956) method was the first method that helps to get the Maximum Flow from source to destination by using it augmenting path algorithm. Dinic (1970) and Cormen et al. (2009) show that if each augmenting path in the shortest one, the algorithm will perform O (mn) augmentation steps; where the number of nodes represented by n and number of arcs is noted as m. Edmonds and Karp (1972) presented a new algorithm where its results were closely to Dinic (1970) algorithm. In addition, the shortest path where the length of the arc equals one could be got with the assistant of Breadth First Search (Thomas et al, 2001; Eppstein, 1998). Ever after that, there are many algorithms have been developed. Ahuja et al. (1989) ameliorated the shortest augmenting path algorithm (Thomas et al., 2001). Orlin (2013) presented ameliorated polynomial time algorithm that determined on the sparse network. As well as, he displayed how to solve the max-flow problem in O (mn), where m= O (n), in addition how to solve the max-flow in O (nm + $m^{31/16}$ $\log^2$ n) time, where m= O ($n^{1.06}$). Thus, this is the improvement of the algorithm that presented by King et al. (1994) which solved the Max Flow in O (nm$log^{n}_{m/(n \log n)}$) time.

The Maximum Flow problem considered to be more difficult in applying genetic algorithm than other popular graph problems due to its various rare characteristics (Munakata and Hashier, 1993). Thus, Munakata and Hashier (1993) applied the Genetic algorithm (GA) to discover the Maximum Flow from the source to the sink in a weighted directed graph. In this approach, each solution is performed by a flow matrix. Two characteristics mainly exist in the fitness function - balancing vertices and the saturation rate of the flow. The initial population is chosen randomly at the beginning, the next generation will be better after using a genetic algorithm. Thus, optimal or near optimal solutions are found after a specific number of iterations. Barham et al. (2016) presented Chemical Reaction Optimization algorithm (CRO) for Max Flow problem. The CRO is presented by Lam and Li (2010), which is a meta-heuristic algorithm that designed for solving combinatorial optimization problems. The MaxFlow-CRO algorithm is proposed to find the best Maximum Flow that could be transported from the source to the sink in a flow network with no constraints for the capacity and violation where the flow in every arc rests within the upper bound value of the capacity.

3. Grey wolf optimization

The inspiration for Grey Wolf Optimizer (GWO) is a Species of Wolves called the Grey Wolf (Canis lupus), by imitating its hunting methods and hierarchical pack distribution which are referred to as Alpha, Beta, Delta, and Omega. These are used to imitate the series of commands as shown in Fig. 1 (Mirjalili et al., 2014). As seen, sovereignty reclines from top to bottom. The first level is *Alpha* which is the leader, which is not necessary to be the most robust wolf but the superior to other wolves in managing the pack. Thus it is responsible for the decision making. The second level is *Beta* which helps Alpha in decision making. Thus, it represents as the mentor to Alpha and an educator to the pack. The third level is *Delta* controls Omega. This category could be Scouts, sentinels, elders, hunters and caretakers. Finally, the fourth level is Omega that acts as the scapegoat and gives up to all dominant wolves. The Hunting behavior of Grey Wolves is split into three procedures: chasing, encircling and attacking the victim as described in (Mirjalili et al., 2014).

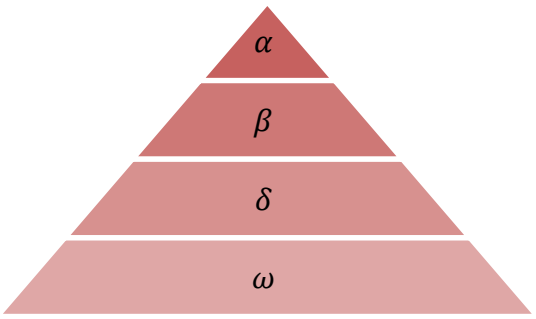

Fig. 1: Hierarchy of grey wolf

3.1. Chasing phase

The Algorithm considers that Alpha (α) as the best solution, Beta (β) as the second best solution and Delta (δ) as the third best solution. However, Omega represents the rest candidate solutions. Thus the hunting is led by the dominant wolves (α, β, and δ). In other words, Grey Wolves could recognize the position of the prey through an iteration process and surround it.

3.2. Encircling phase

In this phase, Grey Wolves encircle the victim through the hunt (optimization). In addition, it is mathematically modeled by Eqs. 2 and 3.

$$\vec{D} = |\vec{C}.\vec{X}_p(t) - \vec{X}(t)| \quad (2)$$
$$\vec{X}(t+1) = |\vec{X}_p(t) - \vec{A}.\vec{D}| \quad (3)$$

Such that D illustrates the distance between the location of the prey (X_p) and the location of the wolf (X) and the existing iteration number is cited by t. A and C are coefficient vectors as in Eqs. 4 and 5.

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \quad (4)$$
$$\vec{C} = 2\vec{r}_2 \quad (5)$$

where, r_1 and r_2 are chosen randomly in [0, 1] and a is decreased from 2 to 0 linearly over the iteration.

3.3. Hunting phase

The hunt generally is led be the leader (α). However, sometimes Beta and Delta contribute in hunting. In another hand, there is no idea about the position of the prey that represents the optimum. Therefore, the algorithm assumes that Alpha, Beta, and Delta have preferable knowledge about the position of prey. Thus, the algorithm saves the first three best solutions then update the locations of the rest wolves (Omega) depending on the position of the dominant wolves (best search agent) according to the Eqs. 6, 7 and 8.

$$\vec{D}_\alpha = |\vec{C}_1.\vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_2.\vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_3.\vec{X}_\delta - \vec{X}| \quad (6)$$
$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_\alpha.(\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_\beta.(\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_\delta.(\vec{D}_\delta) \quad (7)$$
$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (8)$$

3.4. Attacking phase (Exploitation)

In this case; when the prey stops proceeding, the hunting (optimization) ends by assaulting it. It proceeds by reducing the value of a from 2 to 0 linearly. Thus this reduces the value of A which is a random value in [$-a$, a]. When A<1, candidate solutions tend to converge towards the prey. As well as that the algorithm prone to the local stagnation.

3.5. Search for prey phase (Exploration)

To avert the local stagnation, random values A is greater than 1 or less than -1, are used to force the Grey Wolves far from the victim. This emphasizes exploration and searches globally. When A>1 compels the Grey Wolves to space from the victim to discover fitter prey. There is another component that affects this phase which is C and belongs to [0, 2].This component represents random weights of the prey in determining the distance. When C>1, it emphasizes its influence. However, when C<1, it reduces its effect.

4. Algorithm "MAXFLOW-GWO"

The Grey Wolf Optimization algorithm (MAXFLOW-GWO) is developed to solve the maximum flow problem (MFP). It is theoretically analyzed; and it is implemented and tested on datasets with different sizes. The run time performance of the MAXFLOW-GWO algorithm is compared with run time of the Ford-Fulkerson algorithm on the same datasets. By applying Grey Wolf Optimization to get the optimal solution for Maximum Flow problem, Figs. 2-6 show the proposed pseudo-code for "MaxFlow-GWO" algorithm.

1. Initialize the Grey Wolf population X_i (i=1, 2, 3, ……, n)
2. Initialize A, C and a.
3. Choose the wolf (source) and prey (sink) randomly.
4. Calculate the numbers of clusters by equation 9.
5. Calculate the numbers of centroids by equation 10.
6. Calculate the fitness for each wolf X_i.
7. Invoke Cluster Function.
8. Store the shortest path between source and destination.
9. Set the Capacity C (i, j) randomly for each edge (i, j) between Wolf (i) and Wolf (j).
10. Invoke Max-Flow Function
11. Return the best solution.

Fig. 2: Pseudo-code for "MaxFlow-GWO" Algorithm

4.1. Initialization stage

As presented in Fig. 2 initialize the Grey Wolf population, which is chosen at random. From these wolves select one of them randomly to be the prey and choose another one to be the source.

4.2. Fitness function

The goal of fitness function is to measure the minimum distance between each wolf and all centroids to assign to the nearest one and join to that cluster to be a member of its pack. As presented in Fig. 3.

1. For each Wolf (i)
2. Calculate the distance between each wolf (i) and all centroids by the equation 11.
3. If Wolf (i) is not assigned
4. Assign Wolf (i) to its closest centroid.
5. End For (if i= max number of wolves)

Fig. 3: Fitness function

4.3. Clustering

According to Mirjalili et al. (2014), let the group size between 5 and 12 wolves. Thus, K-means clustering algorithm is used to group each pack at least 5 wolves and at most 12 wolves. At the beginning, the number of clusters is calculated by Eq. 9. Moreover, the number of centroids is measured by Eq. 10. Thereafter, centroids are chosen randomly. Each wolf assigned to its nearest centroid by using Eq. 11. The double notation in the equation denotes that it is a function of Euclidean distance (Fong et al., 2014). Such that X_i is the wolf (i) and Cen_j is the centroid in the cluster j.

$$Number\ of\ Clusters = Ceilling\ (\frac{Number\ of\ wolves}{12}) \quad (9)$$

$$Number\ of\ centroids = Number\ of\ Clusters \quad (10)$$

$$Distance = min\ \left|\left|X_i - Cen_j\right|\right|. \quad (11)$$

As shown in Fig. 4, each cluster k finds the first three best solutions randomly at the beginning. In other words, it finds α, β, and δ. During the three iterations, the fittest search agents update their positions around the prey relying on the positions α, β and δ by using Eqs. 2-8. At the end, each cluster has three best solution which is locally optimal, then compare these local optimal with others local optimal in other clusters. The minimum local optimal presents the first global optimal. Then, the second local optimal illustrates the second global optimal and so on.

1. For each cluster K
2. Find the first three search agents α, β and δ.
3. While (iteration <= max_iteration)
4. Update the position of the current search agents
5. By the equations 6, 7 and 8.
6. Calculate the fitness of all search agents.
7. Update a, A and C.
8. Update α, β, and δ.
9. Iteration= iteration +1.
10. End while.
11. Return α for each cluster K.

Fig. 4: Cluster function

4.4. Shortest path function

After all, clusters are constructed, three factors are determined: α, β, and δ. The factor α represents the nearest wolf to the prey in the cluster, the next closest wolf to the prey in the cluster is represented as β, and δ is the third nearest wolf to the prey in the same cluster. The lowest values of α, β, and δ are presented the minimum local optimal for each cluster. Then, the entire local optimal are compared, to find the first global optimal that is considered as the first step in the shortest path. The second global optimal is considered as the second step in the shortest path. And so on until the source is reached. In other words, not all clusters are included in the shortest path, depending on the distance between two wolves, as illustrated in Fig. 5.

1. Compare all local_optimal clusters
2. Set Global_Optimal = nearest cluster to the destination
3. Path [1]= Global_Optimal
4. For cluster[i]
5. Global_Optimal= Min (all Local_Optimal -
6. Global_Optimal)
7. Path [j: 2 to source]= j_Global_Optimal
8. End for

Fig. 5: Shortest path between source and destination

4.5. Max-Flow function

After applying the previous functions, all wolves are grouped, each 12 in one cluster, and got all clusters done, the Max Flow function is applied. Numbers of edges between all wolves are found depending on the shortest path. In other words, there exist many paths from the wolf (source) to the prey (sink). In this function, Ford- Fulkerson algorithm (King et al., 1994) is used, which depends on the augmenting paths that found in the residual graph, which is used at each iteration. The flow on any edge could be raised or reduced. Thus, reducing the flow value on some edge may be substantial for with a view to being able to transfer more flow. Iterate the augmenting flow until the residual network graph has not any more augmenting paths.

1. Select all paths from the wolf (source) to the prey (destination).
2. For each edge (i, j) in the graph G.
3. Flow of edge= 0.
4. While there is a path (p) from the wolf (source) to the prey in the remaining graph.
5. Remaining_Capacity(p)= min (remaining_capacity (i, j) for this edge in the path (p).
6. Flow= flow+ remaining_capacity (p)
7. For each edge (i, j) in the path (p)
8. If (i, j) is a forward edge
9. Flow (i, j) = Flow (i, j) + remaining_capacity (p).
10. Else
11. Flow (i, j) = Flow (i, j) - remaining_capacity (p).
12. Return the maximum flow

Fig. 6: Max-Flow function

For the initial Maximum Flow of the population O (N E F), where N indicates the number of wolves (vertices), E is the number of edges between wolves and F represents the Maximum Flow in the graph. Below is the complexity calculation for each function and method:

- The fitness function is O (n); Cluster function is O (n), the shortest path between source and sink is O (n); where n is the number of nodes.
- Maximum Flow function: while loop is repeated F_m times at most, where F_m represents the maximum flow. Moreover, the cost of finding the augmenting path is O (E).In case the graph is complete, the F_m =E. Thus the run time complexity of Maximum Flow function is O (E *F_m) = O (E^2).
- Total time complexity of "MaxFlow-GWO" Algorithm is O (n +E^2)

5. Experimental results

In order to evaluate the performance of the MaxFlow-GWO, MATLAB program based simulation program was developed by using a dataset of different network sizes, which indicates the number of vertices in the graph. Moreover, the dataset sizes were taken between 50 and 1000 vertices according to previous studies (Barham et al., 2016). In this study, each scenario was repeated 10 experiments based on previous studies (Barham et al., 2016), and not depending on one experiment, to generate accuracy by using these specifications: Intel (R) core (TM) i7-47000MQ CPU with 2.40 GHz, 16 GB RAM and Windows7, 64-bit operating system. Table 1 presents the average run time for different datasets which were calculated in seconds. It is clear from Fig. 7 that the time complexity is quadratic polynomial which means it increases with increasing the

number of vertices in the graph. In addition to that, it is fairly a good enough performance. Depending on Fig. 7 and Fig. 8, it is obvious that the experimental and theoretical results are very approximate. Ford-Fulkerson algorithm is chosen to compare with because it is the general algorithm that solved maximum flow problem (Goldberg and Tarjan, 1988; Chintan et al., 2010) and to validate the MAXFLOW-GWO algorithm. Moreover, the same specifications that were used for the proposed algorithm as shown in Table 2. In addition to that, it was repeated 10 experiments for each data size. Thus, Fig. 9 shows the comparison between MAXFLOW-GWO algorithm and Ford-Fulkerson algorithm in term of average running time for experimental results. It is obvious from Fig. 9 that the proposed algorithm accomplished better performance, especially for networks with large sizes.

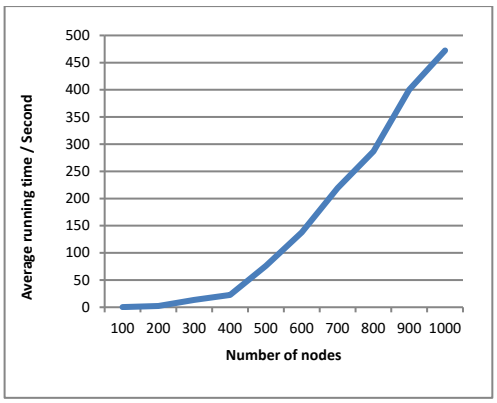

Fig. 7: Average running time for experimental results for MAXFLOW-GWO

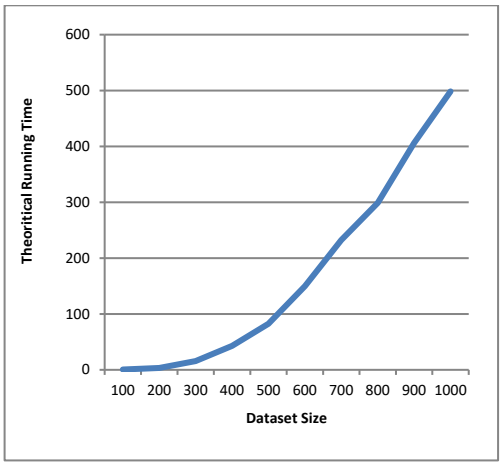

Fig. 8: Theoretical runtime

6. Conclusion

This paper proposes a novel solution to Maximum Flow problem using the Grey Wolf Optimization algorithm. The GWO is used to get the optimal solution for Maximum Flow problem. The maximum flow problem was reviewed. The proposed MaxFlow-GWO algorithm was introduced and explained how it can be used to solve the maximum flow problem. The run time complexity of the algorithm is presented. The theoretical run time complexity is estimated to be O (N E F), where N indicates the number of wolves (vertices), E is the number of edges between wolves and F represents the Maximum Flow in the graph. The time complexity of MaxFlow-GWO algorithm is proven theoretically to be O (n+ E^2). Experimentally, the run time complexity is obtained as a quadratic polynomial, on the tested dataset, which indicates it increases proportionally to the number of vertices in the graph. Thus, theoretical and experimental results converge well.

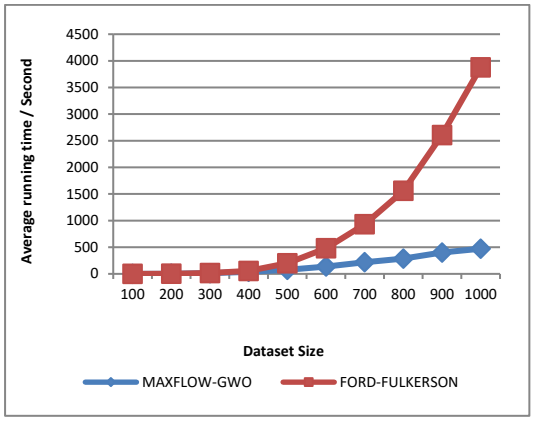

Fig. 9: The experimental results for Ford-Fulkerson Algorithm and MAXFLOW-GWO

Table 1: Run time of MaxFlow-GWO for various datasets

Number of nodes	Average run times/ Seconds	Network Size	Average run times/ Seconds
50	0.115	550	98.236
100	0.328	600	137.827
150	0.936	650	163.578
200	2.356	700	219.327
250	5.866	750	253.184
300	13.603	800	286.781
350	22.706	850	362.921
400	38.236	900	399.954
450	51.625	950	432.627
500	76.365	1000	472.345

Table 2: Run time of Ford-Fulkerson for various datasets

Number of nodes	Average run times/ Seconds	Network Size	Average run times/ Seconds
50	0.1225	550	312.3281
100	0.3419	600	479.9688
150	0.9788	650	700.4531
200	2.4656	700	930.7500
250	6.8125	750	1175.7812
300	15.2344	800	1559.79687
350	27.0781	850	2017.125
400	54.2656	900	2606.609378
450	114.5000	950	3210.9375
500	198.6094	1000	3878.0625

However, since institutional communication professions have been continually exploring effective measurement metrics for their communication initiatives, focusing on how communication practices can be effectively linked to improved performance (Altamony et al., 2012; Alkalha et al., 2012; Masa'deh et al., 2015; Shannak et al., 2010), further research is required. As a future direction, MaxFlow-GWO can be developed to obtain better performance by implementing it on parallel computer. Moreover, a comparison between this proposed algorithm and other Metaheuristics which are utilized to solve the Maximum Flow problem could be presented.

References

Ahuja RK, Magnanti TL, Orlin JB (1989). Network flows. In: Nemhauser GL, Rinnooy Kan AHG, and Todd MJ (Eds.), Optimization handbooks in Operations Research and Management Science, 1: 211-369, Amsterdam, Netherlands.

Alkalha Z, Al-Zu'bi Z, Al-Dmour H, Alshurideh M, and Masa'deh R (2012). Investigating the effects of human resource policies on organizational performance: An empirical study on commercial banks operating in Jordan. European Journal of Economics, Finance and Administrative Sciences, 51(1): 44-64.

Altamony H, Masa'deh R, Alshurideh M, and Obeidat B (2012). Information systems for competitive advantage: Implementation of an organisational strategic management process. In the 18th IBIMA Conference on Innovation and Sustainable Economic Competitive Advantage: From Regional Development to World Economic, Istanbul, Turkey.

Barham R, Sharieh A, and Sliet A (2016). Chemical reaction optimization for max flow problem. International Journal of Advanced Computer Science and Applications, 7(8): 189-196.

Chintan J, Garg DG, and Goel SG (2010). An approach to efficient network flow algorithm for solving maximum flow problem. .Ph.D. Dissertation, Thapar University, Patiala, India.

Cormen TH, Leiserson CE, Rivest RL and Stein C (2009). Introduction to algorithms. MIT Press, Cambridge, USA.

Dinic EA (1970). Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Math Doklady, 11: 1277-1280.

Edmonds J and Karp RM (1972). Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM), 19(2): 248-264.

Eppstein D (1998). Finding the k shortest paths. SIAM Journal on Computing, 28(2): 652-673.

Fong S, Deb S, Yang XS, and Zhuang Y (2014). Towards enhancement of performance of K-means clustering using nature-inspired optimization algorithms. The Scientific World Journal, 2014: Article ID 564829, 16 pages. https://doi.org/10.1155/2014/564829

Ford LR and Fulkerson DR (1956). Maximal flow through a network. Canadian Journal of Mathematics, 8(3): 399-404.

Goldberg AV and Tarjan RE (1988). A new approach to the maximum-flow problem. Journal of the ACM (JACM), 35(4): 921-940.

King V, Rao S, and Tarjan R (1994). A faster deterministic maximum flow algorithm. Journal of Algorithms, 17: 447-474.

Kirkpatrick S, Gelatt CD, and Vecchi MP (1983). Optimization by simulated annealing. Science, 220(4598): 671-680.

Lam AY and Li VO (2010). Chemical-reaction-inspired metaheuristic for optimization. IEEE Transactions on Evolutionary Computation, 14(3): 381-399.

Masa'deh R, Tayeh M, Al-Jarrah IM, and Tarhini A (2015). Accounting vs. market-based measures of firm performance related to information technology investments. International Review of Social Sciences and Humanities, 9 (1): 129-145.

McHugh JA (1990), Algorithmic graph theory. Prentice-Hall, Upper Saddle River, USA.

Mirjalili S, Mirjalili SM, and Lewis A (2014). Grey wolf optimizer. Advances in Engineering Software, 69: 46-61.

Munakata T and Hashier DJ (1993). A genetic algorithm applied to the maximum flow problem. In the 5th International Conference on Genetic Algorithms, Urbana-Champaign, Urbana, USA: 488-493. Available online at: http://grail.cba.csuohio.edu/~munakata/publs/pdf/ICGA93.pdf

Orlin JB (2013). Max flows in O (nm) time, or better. In the 45th annual ACM symposium on Theory of Computing, ACM, Palo Alto, USA: 765-774. https://doi.org/10.1145/2488608.2488705

Shannak R, Masa'deh R, Obeidat B, and Almajali D (2010). Information technology investments: A literature review. In the 14th IBIMA Conference on Global Business Transformation Through Innovation and Knowledge Management: An Academic Perspective, Istanbul-Turkey: 1356-1368.

Tarjan RE (1983). Data structures and network algorithms. Society for Industrial and Applied Mathematics, Philadelphia, USA.

Thomas H, Cormen Leiserson CE, Rivest RL, and Stein C (2001). Introduction to algorithms. MIT press, Cambridge, USA.