An Ant Colony Algorithm for Solving Bi-criteria Network Flow Problems in Dynamic Networks

Sahar Abbasi[1], Mohamad Taghipour[2]

Department of Industrial Engineering, nonprofit institution of higher education, Aba-Abyek, Qazvin, Iran
Department of Industrial Engineering, Robat-Karim Branch, Islamic Azad University, Tehran, Iran

Abstract

Multi-objective shortest path problem is one of the most important problems in network optimization that seeks for the efficient paths satisfying several conflicting objectives between two nodes of a network. The present study tries to focus on the problem of finding the maximum flow along with the shortest path in a dynamic network that this type of the network is presented in. For solving bi-criteria network problems, a two-phased exact algorithm and an ant colony (ACO) algorithm based on bi-criteria are used, where the two-phased exact algorithm is presented by Abbsi et al. and the bi-criteria ant colony algorithm is presented by Ghoseiri et al.First, the two-phased complete enumeration algorithm was used to generate the non-dominated paths. Then, the efficiency and validation of the solutions generated by both the algorithms are compared. The computational results for 33random instances showed that, the CPU time of the ACO algorithm has exponential growth comparing to the two-phased complete enumeration algorithm.

Key Words: Bi-criteria network Flow, Shortest path problem, Ant colony algorithm.

1. Introduction

Considering the maximum flow problem and the shortest paths together in some cases is inevitable since both are pervasive in real applications. But each one is a sub-problem in algorithms for the minimum cost flow problems; in fact, they capture different aspects of the main problem. Minimum cost problems include minimizing the cost arcs, while maximum flow problems maximize traversing flow but not costs. The shortest path problem and the maximum flow problem combine all the basic ingredients of network flow (Ahuja,1993: 166-167). It is a given the fact that in most real-world problems we need to optimize more than one criterion. Thus, a bi-criteria network problem is addressed in this study, taking maximum flow and minimum cost from the source node to the sink into consideration. However, the mentioned criteria are in conflict with each other; in other words, the path with maximum flow does not necessarily have the minimum cost; therefore, an optimal path might not be found. In this condition a set of non-dominated paths will be determined (Abbasi et al, 2014: 1-19) For our study, we used the time-expanded network and two-phased exact algorithm In this research, the flow of traversing through arcs is considered to be independent of time and, also, each arc has limited capacity. Hence, two algorithms are used to figure out the problems of dynamic networks in which K shortest path(s) with maximum flow from a source node to a destination node will be found.

Given a directed network $G = (N, A, T)$ is a network flow with node set N, $| N |=n$, arc set A, $|A| = m$ and a time horizon T, where deadline T is a positive integer. There is a constant u_{ij}, which is interpreted as an upper bound on the rate of flow entering arc$(i,j)\epsilon A$. A non-negative transit time function$\lambda_{ij}(t)$: $\{0,1,\dots T\} \to \{0,1,\dots T\}$is the traverse-time function of arc$(i,j)\epsilon A$, which is the time it takes for a flow to traverse from node$i$ to node j , when given is that the flow starts departing node i at time t. Hence, the flow will reach node j at time $t + \lambda_{ij}(t)$. To ensure that the flow reaches the destination node $n$before the deadline, one might impose $\lambda_{ij}(t)\epsilon 0,1,\dots T - t$.Note that a zero traverse-time is allowed. $c_{ij}(t)$: $\{0,1,\dots T\} \to \mathbb{R}_{\geq 0}$ is the cost function of arc $(i,j)\epsilon A$.which is the cost of a flow traversing from node$i$ to node j. When given is that the flow starts departing node i at time t. We consider the problem that the capacities parameters are fixed for each arc but the transit cost varies with time. Therefore, the source node has no entering arc and destination node has no outgoing arc. Ford and Fulkerson introduced new type of networks as time-expanded networks, which is used in the discrete time model for convert dynamic networks to static networks (Ford and Fulkerson, 1962: 419-433).

Definition1. For a dynamic network $G = (N, A, T)$ the time expanded network $G^T = (N^T, A^T)$is denoted by $G(\theta)$,where $\theta = \{t_0, t_1, \dots, t_p\}$, each node $i\epsilon N$ has $T + 1$ copies in N^T, which are shown $N_0, N_1, \dots, N_p$and each arc$(i,j)\epsilon A$has$T - |\lambda_{ij}| + 1$copies inA^T. Traversing through arc$(i_{q-1}, j_{q'})$ where, $t_{q'} = t_{q-1} + \lambda_{i,j}$ corresponds to leaving node $i$at time $t_{q-1}$and arriving at node $j$at time $t_{q'}$for $q = 1, \dots, p - 1$(Ford and Fulkerson, 1962). The aim of solving this problem is to find all efficient paths from a source node to a sink node in a fixed time horizon T.

Definition2. A path in a network $G = (N, A)$is a continuous way of getting from one node to another by using a sequence of arcs. A directed path is a path without any repetition of nodes. In other words, a directed path has no backward arcs. Let $\mathcal{P}$ be the set of all directed paths from 1 to N in the network G. For any directed path $\{p = (x_1, x_2, \dots, x_k, x_{k+1})\}\epsilon\mathcal{P}$, the flow of a path and cost of a path are defined as follows:

$$flow(p) = \min_{(x_i,x_{i+1})\epsilon p} \{u_{x_i,x_{i+1}}\} \;\; for\ all\ p\epsilon\mathcal{P}$$

$$cost(p) = \sum_{i=1}^{k} C_{x_i,x_{i+1}} \;, (x_i, x_{i+1})\epsilon p \quad for\ all\ p\epsilon\mathcal{P}$$

Where $u_{x_i,x_{i+1}}$ and $C_{x_i,x_{i+1}}$ are the capacities and costs assigned to each arc $(x_i, x_{i+1})\epsilon p$.

Let Pbe the set of all paths from node 1 to node Nin a network and let$f_1(p) = cost(p), f_2(p) = flow(p), \forall p \in P$. a path $p_1 \in P$is said to dominate another path $p_2 \in P$, and we write $p_1 \leq p_2$ if both the following conditions are true:

Path p_1 is no worse than path p_2 in all objectives (cost and flow).

Path p_1 is strictly better than path p_2 in at least one objective (cost or flow).

$$p_1 \leq p_2 \ iff \begin{cases} f_i(p_1) \leq f_i(p_2) \ \forall i \epsilon 1,2 \\ \exists j \epsilon 1,2 \ \ f_j(p_1) \leq f_j(p_2) \end{cases}$$

If any of the above conditions is violated, path p_1 does not dominate pathp_2.

Among a set of all paths P, the set of particular paths known as non-dominated paths,P_N, are those that are not dominated by any member of set P. In other words $p \in P_N$if and only if it is not possible to find a $p^{'} \epsilon P$ and $p^{'} \neq P$ such that cost or flow is improved without obtaining a worse cost or flow, respectively(Abbasi et al, 2014: 1-19).

2. Literature Review

The classic shortest path problem considers just one objective function or criterion, which usually consists minimizing the sum of the costs or weights of the path, and Dijkstra algorithm or label-setting algorithm can be applied for solving such problems with positive value (Bellman, 1958: 87-90). Many shortest path problems consider more than one criterion or objective function. Batta and Chiu (Betta et al, 1988: 84-92), Current and Min (Current et al, 1986:187-201) and (Current et al, 1993:426-438)studied applications of such problems.

The shortest path problems in real-world problems are categorized into different types such as static, dynamic, multi- criteria with parameters of discrete or continuous and deterministic or stochastic.

Several works investigate multicriteria shortest path problems in dynamic networks:Cook et al initially presented an algorithm based on the Bellman's principle of optimality to solve the discrete time dynamic shortest path problems(Cooke and Halsey, 1966:492-498). Nasrabadi et al. presented an algorithm to figure out dynamic flow problem in a discrete time model in which transit times, transit costs, transit capacities, storage costs, and storage capacities vary with time(Nasrabadi, 2010:429-447). Guerriero et al. considered a problem that includes cost and time as two main parameters, associated with each arc; a time window is associated with each node, and a feasible path must satisfy assumed constraints. They proposed the Multi-dimensional labeling algorithms to solve these problems and did many computational tests to evaluate the proposed method (Guerriero and Pugliese, 2011: 395-340). Pugliese et al. focused on the shortest path problem with forbidden paths and defined various algorithms based on dynamic programming (Pugliese andGuerriero, 2011). Their proposed algorithms can be viewed as an extension of the node selection approach proposed by Desrochersin 1988(Desrochersand Soumis, 1988:242-254) Extensive computational results showed that the dynamic programming-based solution is very effective. In recent years, some researches in multi-criteria shortest path problems have been done that are summarized briefly:

Chitra and Subbaraj presented a feasible multi-objective evolutionary algorithm based on the non-dominated sorting genetic algorithm and simulated to solve dynamic shortest path routing problem in computer networks where multiple Pareto optimal solutions can be found in one simulation run(Chitra and Subbaraj,2012:1518–1525). He and Song addressed the optimal path finding problem in a stochastic time-dependent network where all link travel times are temporally and spatially correlated. They designed an exact label-correcting algorithm to find the optimal (He and Song, 2012:579–598) .Abbasiet al. proposed a two-phased exact algorithm and a Cross-Entropy (CE) algorithm based on bi-criteria to finding the maximum flow along with the shortest path in a dynamic network, where the costs change as time functions. The computational results for 53 random instances showed that for large size problems, CPU time has exponential growth compared with that of the complete enumeration algorithm. In the next section, the bi-criteria labeling algorithm will be explained (Abbasi and Ibrahimnejad, 2014: 1-19).

Ant colony optimization (ACO) first introduced by Dorigo et al in 1991 for solving Traveling Salesman Problem.This method of optimization inspired by the behavior of various ants when they are searching for shortest path between possible paths to find the food sources. For solving the problem with ACO algorithm, in most cases, the problem must be defined and represented with a graph (Dorigo et al, 1991:55-90).

In recent years many researchers have developed various ACO algorithms for combinatorial problems such as vehicle routing problem, traveling sales man problem, production scheduling, sequential ordering problem, telecommunication routing, etc. some of these ACO algorithms are proposed by many researchers that they are different in procedures of updating pheromone trails, evaporation and transition rules. Ant System (AS) was first developed by Dorigo et al. and applied to solve classic Traveling Salesman Problem (Dorigo et al, 1991:55-90). Six years later Dorigo and Gambardella introduced another algorithm that performed better than AS and was called ant colony system (ACS). ACS uses different procedures in local and global updating of the pheromone trails as well as transition rule (Dorigo and Gambardella, 1997:53-66). Ghoseiri et al. proposed an algorithm based on multi-objective ant colony optimization (ACO) to solve the bi-objective shortest path problem. to analyze the efficiency of the algorithm and check quality of solutions, the results of two sets of small and large sized are compared with those of label correcting solutions .To compare the Pareto optimal frontiers produced by the suggested ACO algorithm and the label correcting algorithm, some performance measures are employed that consider and compare the distance, uniformity distribution and extension of the Pareto frontiers. The results on the set of instance problems show that the proposed algorithm produces good quality non-dominated solutions and time saving in computation of large-scale bi-objective shortest path problems (Ghoseiri and Nadjari, 2010:1237-1246).

Chaharsooghi et al. addressed the multi-objective resource allocation problem (MORAP) and proposed a modified version of ant colony optimization (ACO) to solve it. They increased the efficiency of algorithm by increasing the learning of ants. Yu et al. defined the concept of shortest path on the basis of a scale-free dynamic and stochastic network model, then they proposed a temporal ant colony optimization (TACO) algorithm for searching the shortest paths in the network (Chaharsooghi and MeimandKermani, 2008:167-1770).

Due to the extensive applications of the bi-criteria dynamic network, this present study has used two new algorithms for figuring out bi-criteria network flow problems in discrete time model. The first algorithm is a complete enumeration of a two-stage algorithm that has two conflicting objective functions; one related to maximization the flow of traversing through the dynamic network and the other one concerned traversing flows through the shortest path. The second is the development of the ACO algorithm based on bi criteria. These two proposed algorithm indicate that in time zero $(t = 0)$ the flows set to start and illustrates the time step visit the next nodes. Ultimately, it is tried to know after how many time steps the flows reach to the sink node. In this way K-shortest path with maximum flow in network is specified (Abbasi and Ibrahimnejad, 2014: 1-19).

This paper is organized as follow.The following part deals with the complete enumeration two-stage algorithm and Ant colony algorithm. Section 4 includes the computational results of 33 random instances. In the end, conclusion is presented in Section 5.

3. Two algorithms review

3.1 The complete enumeration two-stage algorithm

As previously mentioned, a complete enumeration two-stage algorithm is used in this section which can produce most of the non-dominated paths. The cost of the network paths is first calculated based on the sum of the costs on the arcs in each path and is ordered in a vector in ascending form (or non-decreasing) (Paths 'cost vector).Then, the flow value of each path focusing on maximizing the flow of traversing network arcs is calculated and is ordered in the flow vector in descending form (or non-increasing) (Paths' flow vector). Afterwards, this algorithm finds the least value in the cost vector of paths and then searches for its related path position in the flow vector to eliminate the next elements having the lower values of flow in the flow vector and vice versa; it finds the maximum value in the flow vector of paths and determines its related path position in the cost vector to eliminate the next elements having greater value of cost. At the end of first stage, some of the dominated paths may not be removed. For this purpose, the remaining dominated paths are eliminated at the second stage to obtain non-dominated paths of the network according to Figure 1. To better understand the algorithm, the pseudo-code are presented in Table 1. (Abbasi and Ibrahimnejad, 2014: 1-19).

Table 1: Pseudo-code for the complete enumeration two-stage algorithm

The complete enumeration two-stage algorithm

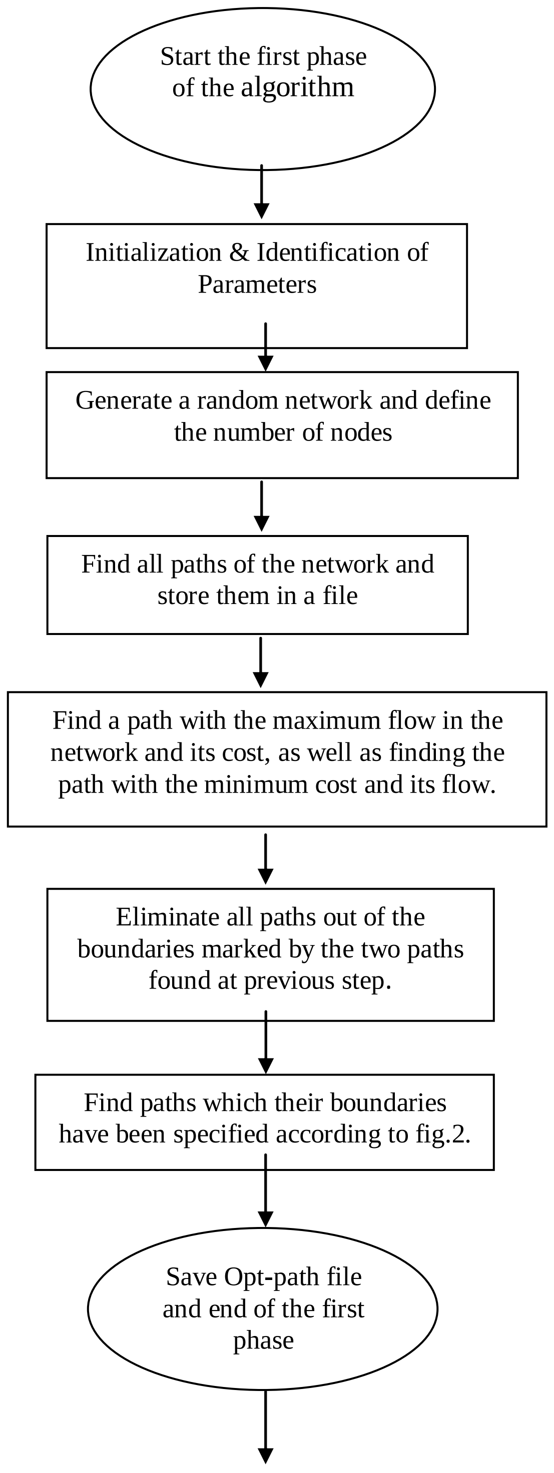
Inputs: Cost vector, Flow vector

Returns: The non-dominated paths(optimal paths)

```
Phase0. Initialization:
    T:Two-dimensional matrix[Cost, Flow]
    N:The total paths
Phase1:
    C = Sort  Matrix T  on  Cost (Ascending)  Then  Flow (Descending)
    F = Sort  Matrix T  on  Flow (Descending)  Then  Cost (Ascending)
    i =1
    while  F[i,1] ≠ C[1,1]  and  F[i,2] ≠ C[1,2]
            i=i+1
    end while
    for  j=i+1   to  N
            delete F[j]
    end for
    N(F) =i=The remaining records in the Matrix F
    i =1
    while  C[i,1] ≠F[1,1]  and  C[i,2] ≠ F[1,2]
            i=i+1
    end while
    for  j=i+1   to  N
            delete C[j]
    end for
    N(C) =i=The remaining records in the Matrix C
Phase2:
    For  i=1 to N(F) -1
```

```
        For  j=i+1 to N(F)
                    If  C[j,2] <= C[ i,2] then
                            Delete C[j]
                    End If
        End For
End For
The remaining records are non-dominated solutions(optimal paths)
For  i=1 to N(C) -1
        For  j=i+1 to N(C)
                    If  F[j,1] >= F[ i,1] then
                            Delete F[j]
                    End If
        End For
End For
The remaining records are non-dominated solutions(optimal paths)
```

As indicated in Table 1, all respondents from Jordan held at least a university degree in a related subject. Not all members of the JSC staff with responsibility for monitoring compliance with IFRS are qualified to do that job. The two respondents from the JSC staff had been in their positions for 8 and 3 years respectively. All academic staff members held a PhD in Financial Accounting at the Jordanian universities and were responsible for preparing and teaching accounting courses which include only an introduction to IFRS. Their experience ranged from 6 to 13 years. The only accountant who agreed to participate had 12 years experience with IFRS. Finally, respondents from the investor group had experience of trading on the ASE for 5 and 9 years respectively.


Start the first phase of the algorithm
Initialization & Identification of Parameters
Generate a random network and define the number of nodes
Find all paths of the network and store them in a file
Find a path with the maximum flow in the network and its cost, as well as finding the path with the minimum cost and its flow.
Eliminate all paths out of the boundaries marked by the two paths found at previous step.
Find paths which their boundaries have been specified according to fig.2.
Save Opt-path file and end of the first phase

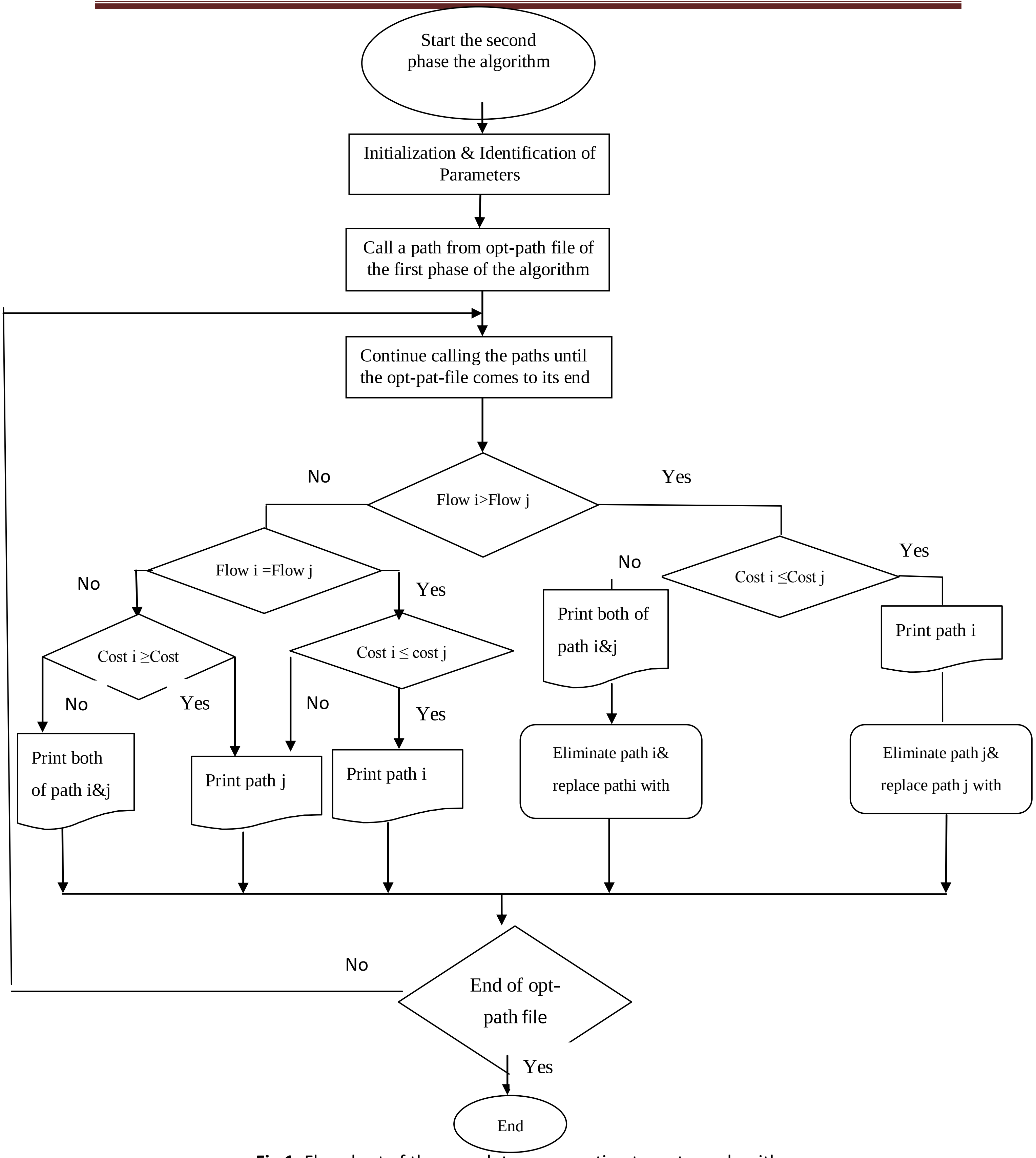

Start the second phase the algorithm
Initialization & Identification of Parameters
Call a path from opt-path file of the first phase of the algorithm
Continue calling the paths until the opt-pat-file comes to its end
No
Yes
Flow i>Flow j
Flow i =Flow j
No
Yes
No
Cost i ≤Cost j
Yes
Cost i ≥Cost
Cost i ≤ cost j
No
Yes
No
Yes
Print both of path i&j
Print path i
Print both of path i&j
Print path j
Print path i
Eliminate path i& replace pathi with
Eliminate path j& replace path j with
No
End of opt-path file
Yes
End

Fig.1: Flowchart of the complete enumeration two-stage algorithm.

3.2. The ant colony optimization algorithm

The suggested ant colony algorithm is composed of some main functions such as pheromone matrix, heuristic parameter, evaporation, local update and global update. These functions are described in this section. The algorithm is designed based on classic ACS proposed by Dorigo and Gambardella with maximum selection in transition rule as well as local and global update. Fig. 2 presents a flowchart for the steps involved in the algorithm. For more details refer to (Ghoseiri andNadjari,2010: 1237-1246).

4. The application of the two algorithms to the problem

The algorithms are created in the MATLAB code and the experiments have been conducted on a Microsoft Windows SEVEN Professional with 2.3GB RAM and 3GB swap, running Digital Intel Core i5 Duo CPU. In order to check for the efficiency and validity of the Ghoseiri's ACO algorithm, two sets of medium- and large-sized test problems are generated randomly using the two-phased exact algorithm, whose parameters of these test problems (such as number of nodes, number of arcs, cost value, flow value) were uniformly distributed within the considered interval. Then results are compared with Ghoseiri's ACO algorithm. The information of problems has been given in Table 2.

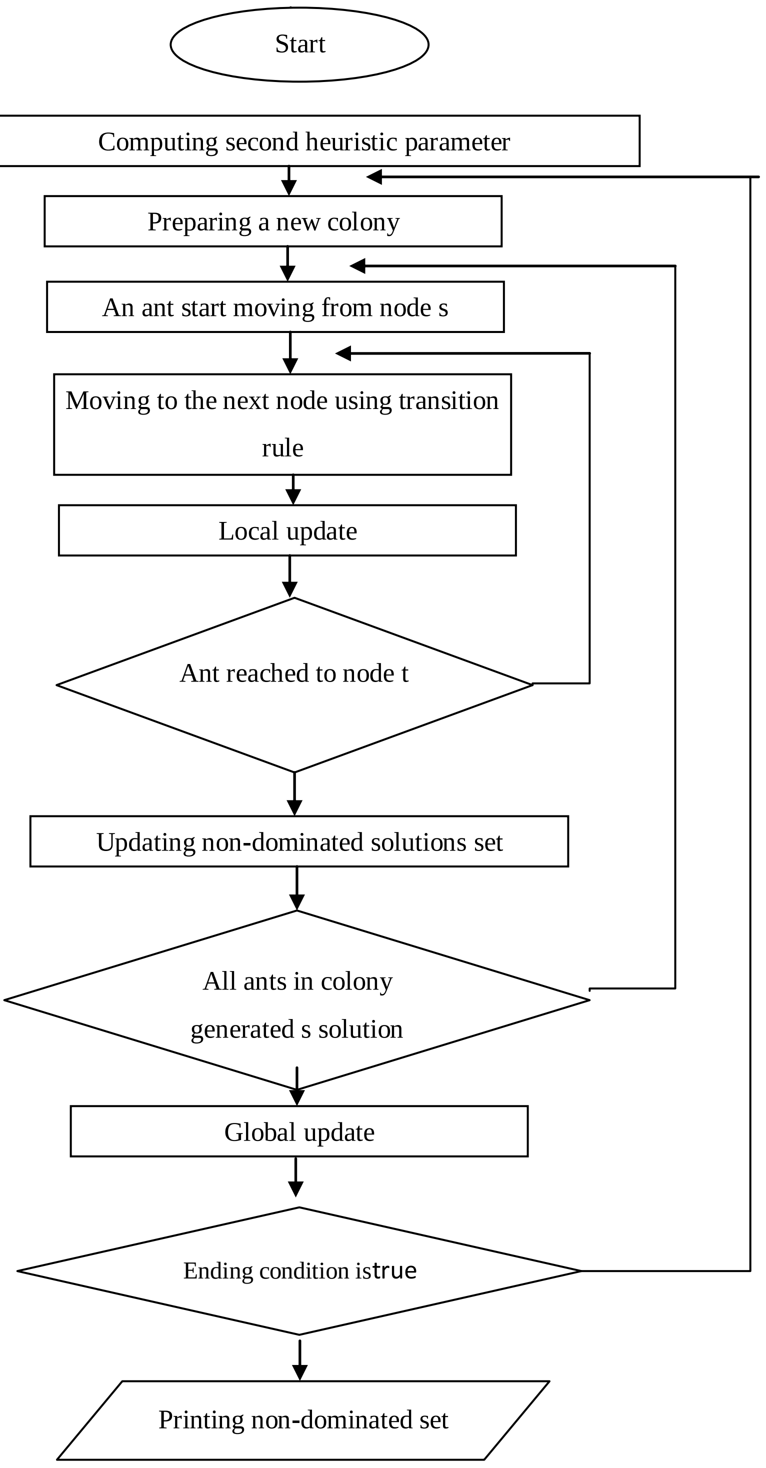

Start
Computing second heuristic parameter
Preparing a new colony
An ant start moving from node s
Moving to the next node using transition rule
Local update
Ant reached to node t
Updating non-dominated solutions set
All ants in colony generated s solution
Global update
Ending condition istrue
Printing non-dominated set

Fig. 2: Flowchart of the ant colony algorithm.

Table 2.Summary of results for the experimental analysis on instance problems.

						Ant Colony parameters										CPU time(s)	
#	#Nodes	#Arcs	T(time horizon)	#Nodes (time-expanded)	#Arcs (time-expanded)	#Iterations	#Ants	q_0	α	β	δ	φ	ρ	a	b	Two phased algorithm	ACO
1	20	51	20	400	1124	100	35	0.95	1	1	0.5	0.99	0.99	24	7	125	20<CPU<100
2	50	343	10	500	3342	100	40	0.95	1	1	0.5	0.99	0.99	24	7	238	20<CPU<100
3	30	125	20	600	2558	100	50	0.95	1	1	0.5	0.99	0.99	24	7	287	20<CPU<100
4	50	335	15	750	4905	100	100	0.95	1	1	0.5	0.99	0.99	24	7	863	20<CPU<100
5	40	210	20	800	4260	100	100	0.95	1	1	0.5	0.99	0.99	24	7	1024	20<CPU<100
6	50	339	20	1000	6586	100	200	0.99	2	2	0.1	0.99	0.99	50	80	1073	100<CPU<800
7	100	1310	10	1000	12427	100	200	0.99	2	2	0.1	0.99	0.99	50	80	3152	100<CPU<800
8	200	5231	5	1000	24339	100	200	0.99	2	2	0.1	0.99	0.99	50	80	7928	100<CPU<800
9	100	1299	15	1500	186546	100	200	0.99	2	2	0.1	0.99	0.99	70	131	8895	100<CPU<800
10	150	2962	10	1500	274776	100	200	0.99	2	2	0.1	0.99	0.99	70	131	9523	100<CPU<800
11	300	11813	5	1500	53835	100	200	0.99	2	2	0.1	0.99	0.99	70	131	12432	100<CPU<800
12	100	1293	20	2000	25167	100	400	0.99	3	4	0.1	0.99	0.99	150	251	13034	100<CPU<800
13	200	5286	10	2000	48205	100	400	0.99	3	4	0.1	0.99	0.99	150	251	13120	100<CPU<800
14	150	2820	15	2250	40886	100	450	0.99	2	4	0.1	0.99	0.99	180	270	14025	100<CPU<800
15	150	3000	20	3000	40296	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
16	200	5223	15	3000	72367	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
17	300	11787	10	3000	107798	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
18	200	5185	20	4000	98717	100	900	0.99	2	4	0.1	0.99	0.99	350	551	>20000	1000<CPU<3000
19	300	11834	10	3000	10793	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
20	300	11750	10	3000	10754	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
21	300	11799	10	3000	10762	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
22	300	11739	10	3000	10768	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
23	300	11811	10	3000	10821	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
2	200	526	15	3000	73285	100	650	0.9	3	4	0.	0.9	0.9	27	38	>2000	1000<CPU<

4		4						9			1	9	9	0	0	0	3000
25	200	5223	15	3000	72198	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
26	200	5188	15	3000	70753	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
27	200	5201	15	3000	72252	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
28	200	5237	15	3000	73346	100	650	0.99	3	4	0.1	0.99	0.99	270	380	>20000	1000<CPU<3000
29	200	5148	20	4000	97654	100	900	0.99	2	4	0.1	0.99	0.99	350	551	>20000	1000<CPU<3000
30	200	5166	20	4000	98852	100	900	0.99	2	4	0.1	0.99	0.99	350	551	>20000	1000<CPU<3000
31	200	5169	20	4000	98293	100	900	0.99	2	4	0.1	0.99	0.99	350	551	>20000	1000<CPU<3000
32	200	5271	20	4000	100706	100	900	0.99	2	4	0.1	0.99	0.99	350	551	>20000	1000<CPU<3000
33	200	5171	20	4000	98078	100	900	0.99	2	4	0.1	0.99	0.99	350	551	>20000	1000<CPU<3000

A set of 33 acyclic network problems ranging from 400 to 4000 nodes are solved and results are compared with solutions produced by the two-phased exact algorithm . Table 2 shows specifications and comparison results and includes parameter values used to solve the instance problems where #Nodes and #Arcs denotes the number of nodes and arcs in origin network, respectively and #Nodes (time-expanded) and #Arcs (time-expanded) denotes the number of nodes and arcs in time-expanded network, respectively. T denotes the time horizon and #ants indicates number of ants in colony and #iterations indicates number of algorithm iterations (colonies). The parameters $q_0, \alpha, \beta, \delta, \phi, \rho, a$ and bdenote the selection parameter in ant colony transition rule, preference weight of pheromone trail, preference weight of the first heuristic parameter, preference weight of the second heuristic parameter, local update evaporation rate, global update evaporation rate, respectively (Ghoseiri andNadjari, 2010: 1237-1246).

Parameter a identifies the number of ants that use only information of the second objective and finally CPU denotes the average CPU time in seconds. The number of iterations (colonies) in all runs considered equal to 100. The number of iterations considered fix in all runs and varied number of ants in colonies regarding the size of problem. Increasing the number of nodes and arcs in network make the algorithm uses more ants to search in larger space. The selection parametersq_0, weighting parameters α, β and δ and also updating parameters ϕ and ρ are experimental values and would be set experimentally. Ghoseiri et al. considered these parameters in the first run of the algorithm respectively set equal to 0.99, 2, 4, 1, 0.9999 and 0.99 and we use these numbers for the first run of the algorithm and in the second and third run they are changed in a way that better solutions are reached (Ghoseiri andNadjari, 2010: 1237-1246).

. Duo to the fact that ACO is a stochastic algorithm, ten runs for each set of parameters are made and the set of parameters yielded the best solutions are reported in Table2.

5. Conclusions

The present study has focused on the bi-criteria dynamic network. First, the bi-criteria dynamic network was converted into a time-phased network as. Then, two objectives were determined for the converted network. The first is maximizing the flow of traversing through the paths and the second deals with obtaining all shortest paths. The above objectives conflict with each other and generally lead to generating the non-dominated paths. In this research, two algorithms were used to obtain the non-dominated paths by which 33 generated random instances were solved. First using a two-stage complete enumeration algorithm the non-dominated paths generated. Then these solutions were compared with ACO algorithm.Computational results show that for problems with greater sizes (more than 800 nodes in time-expanded network), CPU time of the ACO algorithm is much smaller than the complete enumeration algorithm. In other word, the results on the set of instance problems show that the used ACO algorithm time saving in computation of large-scale bi-criteria shortest path problems.

References

R.K. Ahuja, M. Magnanti, J. Orlin," *Network Flows. Theory, Algorithms and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey. (1993)

SaharAbbasi, S.Ebrahimnejad, *The cross-entropy method for solving bi-criteria network flow problems in discrete-time dynamic networks*, Optimization Methods & Software.2014,1-19.

L. R. Ford and D. R. Fulkerson.*Constructing maximal dynamic flows from static flows.*OperationsResearch , 6:419–433, 1958.

L. R. Ford and D. R. Fulkerson.*Flows in Networks*.Princeton University Press, Princeton, NJ, 1962.

R.E. Bellman. *On a routing problem*. Quarterly of Applied Mathematics 16,87–90., 1958.

RBatta, S.S.Chiu. *Optimal obnoxious paths on a network*: Transportation of hazardous materials. Operations Research 36, 84–92.1988.

Current, J.R., Min, H., 1986.*Multiobjective design of transportation networks*: Taxonomy and annotation. European Journal of Operational Research 26, 187– 201.

Current, J.R., Marsh, M., 1993.*Multiobjective transportation network design and routing problems: Taxonomy and annotation*. European Journal of Operational Research 103, 426–438.

L.Cooke, ,E.Halsey,: *The shortest route through a network with time-dependent intermodal transit times*. J. Math. Anal.Appl. 14, 492–498 (1966).

EbrahimNasrabadi , S. Mehdi Hashemi, "*Minimum cost time-varying network flow problems*", Optimization Methods and Software, Volume 25, Issue 3: pages 429-447, 2010.

F. Guerriero & L. Di Puglia Pugliese,*"Multi-dimensional labeling approaches to solve the linear fractional elementary shortest path problem with time windows",* Optimization Methods and Software ,Volume 26, Issue 2: pages 295-340, 2011.

Luigi Di Puglia Pugliese, Francesca Guerriero," *Dynamic programming approaches to solve the shortest path problem with forbidden paths*", Optimization Methods and Software, Version of record first published: 21 Nov 2011.

Desrochers, M. and F. Soumis, 1988. A Reoptimization Algorithm for the Shortest Path Problem with Time Windows, *European Journal of Operational Research,* 35, pp. 242-254.

C. Chitra, P. Subbaraj.,*Anondominated sorting genetic algorithm solution for shortest path routing problem in computer networks*, Expert Systems with Applications 39 , 1518–1525 (2012).

H.He, G.Song*," Optimal paths in dynamic networks with dependent random link travel times",*Transportation Research Part B, 46 , 579–598.2012.

M. Dorigo, V. Maniezzo, A. Colorni, Ant System: *An Autocatalytic Optimizing Process, Technical Report*, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1991.

M. Dorigo, L.M. Gambardella, *A cooperative learning approach to the traveling salesman problem*, IEEE Transaction on Evolutionary Computation 1 (1997).53–66

KeivanGhoseiri, BehnamNadjari: *An ant colony optimization algorithm for the bi-objective shortest path problem*. Appl. Soft Comput. 10(4): 1237-1246 (2010).

S.K. Chaharsooghi, Amir H. MeimandKermani: *An effective ant colony optimization algorithm (ACO) for multi-objective resource allocation problem (MORAP).* Applied Mathematics and Computation.200 (2008),pp.167-177.