Advances in Industrial Engineering, Winter 2021, 55(1): 1-26
DOI: 10.22059/jieng.2021.325262.1773

RESEARCH PAPER

A Comparison of Extended Dijkstra and ACO Algorithm for Shortest Path Problem in Dynamic Networks with Delay Times

Amir Abbas Shojaie[a,*], Seyed Esmail Seyedi Bariran[b]

a. ***School of Industrial Engineering, Islamic Azad University, South Tehran Branch, Tehran, Iran***
b. ***School of Industrial Engineering, Islamic Azad University, South Tehran Branch, Tehran, Iran***

Received: 08 June 2021, Revised: 21 June 2021, Accepted: 22 June 2021

Abstract

Shortest path problem is a typical routing optimization problem that is generally involved with a multi-criteria decision-making process. Therefore, the main objective of this paper is to find the shortest path in discrete-time dynamic networks based on bi-criteria of time and reliability by considering the effect of delay times that varies according to different departure time scenarios. Firstly, the well-known single-criterion Dijkstra's algorithm is extended to fit the conditions of a bi-criteria problem. The solutions obtained from the extended Dijkstra was then compared with a proposed ant colony optimization (ACO) algorithm via a set of multi-objective performance metrics including CPU time, error ratio, spacing and diversity metrics. The analysis was made based on three network scales ranged from small (20-100 nodes) to medium (500-1900 nodes) and large (2000-10000 nodes). The computational results obtained from the analysis suggested that the extended Dijkstra's algorithm has a higher efficiency in medium and large scaled networks. Furthermore, the comparison of the proposed ACO versus Dijkstra's algorithm proved the preference of ACO for networks with larger-scaled (nodes over 5000), while for smaller and medium-scale networks (nodes 20-2000), the extended Dijkstra's algorithm has a dominantly better performance in CPU time as compared to proposed ACO.

Keywords:
Shortest Path Problem;
Bi-Criteria;
Ant Colony;
Dijkstra;
Delay Time;
Dynamic Network

Introduction

The shortest path problem is commonly referred to as one of the most typical network optimization problems that have attracted the attention of many researchers during the previous decade. This attractiveness is mainly due to its practical applications in routing, communications, optical networks, telecommunications, transportation, project timing and resources allocation [1]. In this regard, network flow problems are considered as either

* Corresponding author: (A.A. Shojaie)
Email: Amir@ashojaie.com

static/dynamic or optical/non-optical. In a static network flows problem, time dimension that is a very important component in modeling real cases is usually neglected and the passing time on each arc is assumed as zero. In addition, it is supposed that the values are constant and independent of time. These occur in most optimization cases of real systems such as traffic control, fluid transfer networks, telecommunication networks, and energy transfer networks, systems of refining and distributing of oil, etc. that result in defining the dynamic flow network [2, 3]. Unlike static networks, the status of each node fully dependes on the time in dynamic networks. Therefore, for setting the problem, the state of all nodes should be examined and all optimum paths are selected [4, 5]. In addition to the above classifications, the meta-heuristics algorithms proposed in this study can be alternatively implemented in various optical and non-optical networks applications such as traffic grooming, optical and non-optical routing, wireless optical networks, wavelength conversion, and many other emerging fields such as IOT based optical networks. The shortest path problem can be formulated in non-optical networks as Boolean satisfiability (SAT) problem and further be extended in finding wavelength assignments in optical networks as well [6]. The shortest path planning in optical networks can also be considered as an integer linear programming problem (ILP) that was studied by some Indian researchers using a simplified artificial neural network architecture [7]. Wavelength conversion in shorted path optical networks is another interesting field combing shortest path to optical networks and is frequently addressed in several studies such as [8-10]. shortest path problem techniques can also be effectively applied to many transportation related applications such as traffic grooming as was used in [11, 12]. In terms of optical network design perspectives, it is also important to note that the classical shortest path problems as well as the proposed meta-heuristics have several practical applications. Therefore, it is possible to formulate these algorithms to enhance the performance of both optical and non-optical networks, accordingly.

For the purpose of this study, the focus will be on a type of discrete-time dynamic network in which the departure time is different due to the existence of delay times between the passing times of each two nodes. Therefore, the main objective of this paper is to propose two bi-criteria meta-heuristic algorithms, based on the well-known Dijkstra's algorithm and ACO method for finding those Pareto-optimal solutions that simultaneously secure two purposes of minimizing the time and maximizing the reliability of each path by considering the effect of delay time between the nodes.

The structure of this paper is organized as follows. Section 2 defines the problem and clarifies the specifications of the given network. Section 3 is to provide a comprehensive literature review on Dijkstra and ACO algorithms as relevant to multi-criteria routing optimization problems. In Section 4, the proposed heuristics algorithms are presented based for solving bi-criteria network problems with delay time in node. The implementation procedure and the computational results of 41 random instances (9 small-sized instances, 17 medium-sized and 15 large-sized instances) are discussed in Section 5. In the end, the work concludes with the comparison of the proposed methods based on given performance metrics and directions for future research.

Problem definition

In shortest path problems, the main challenge is to start from a source node and then pass through a series of middle nodes to reach the sink node while optimizing one or more criteria at decision points. The criteria related to these types of networks can be either deterministic/stochastic or discrete/continuous. In dynamic networks, some parameters may change as time goes on and therefore affecting the final solutions resulted from decision-making alternatives. The delay time existing between two nodes along the path is one of the most

frequently occurred variable conditions that is studied in this paper. Furthermore, for solving this type of discrete-time dynamic network, in addition to finding the non-dominated (also called efficient) paths, the primary objective is to obtain the total delay times in each path and in different start times so that it is possible to compute the delay time in subsequent nodes.

The network configured for this problem is denoted as $G(N, A)$ in which "N" and "A" are the total nodes and total directed arcs among these nodes, respectively. Referring to the graphical illustration shown in Fig. 1, it is assumed to move from source node s to sink node t. The symbol "T" is also considered as the total delay time and therefore any path having a delay time more than T is omitted. In addition, "C_{ij}" stands for travel cost from node i to j and "R_{ij}" stands for reliability. The goal in this network is to find all non-dominated paths with minimum cost and maximum reliability from source (s) to sink (t).

Definition 1: A walk in a network $G = (N, A)$ is a continuous route for getting from one node to another by passing through a sequence of arcs. A path is then a walk without any repetition of nodes. In addition, a direct path is a path if the arcs are directed within the given path. In other words, a directed path has no backward arcs. Let $\mathcal{P}$ be the set of all directed paths from node 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 reliability and cost of a path are defined as in Eqs. 1 and 2 as follows:

$$Reliability\ (p) = \coprod_{(x_i, x_{i+1}) \epsilon p} \{R_{x_i, x_{i+1}}\}\ for\ all\ p \epsilon\ \mathcal{P}, \tag{1}$$

$$Cost\ (p) = \sum_{i=1}^{k} C_{x_i, x_{i+1}}\ \epsilon\ p\ \ for\ all\ p \epsilon\ \mathcal{P}, \tag{2}$$

Wherein,
$R_{x_i, x_{i+1}} and\ C_{x_i, x_{i+1}}$, are the reliabilities and costs assigned to each arc $(x_i, x_{i+1}) \epsilon\ p$, respectively.

Definition 2: Let P be the set of all paths from node 1 to node N in a network and let $f_1(p) = cost\ (p),\ f_2(p) = reliability\ (p),\ \forall\ p \epsilon\ P$, a path $p_1 \epsilon\ P$ is said to dominate another path $p_2 \epsilon\ P$, and we consider $p_1 > p_2$, if both the following conditions are met:

1. Path p_1 is no worse than path p_2 in all objectives (cost and reliability).
2. Path p_1 is meaningfully better than path p_2 in at least one objective (cost or reliability) as given in relation 3 below:

3. $$p_1 \geq p_2\ iff \begin{cases} f_1(p_1) < f_2(p_2) \\ f_2(p_1) > f_2(p_2) \end{cases} \tag{3}$$

If any of the above conditions are violated, path p_1 does not dominate path p_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, for $p \epsilon\ P_N$, 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.

In Table 1, information related to the network given in Fig. 1 is elaborated. The second and third columns show cost and reliability between each two arcs respectively. The values are given under columns L1 to L7 show the delay times between the nodes based on seven different start times. Table 2 presents the correlation between four non-dominated paths along with the cost, reliability, and total delay time for each path. Referring to Table 2, the cost and reliability of the first path are 10 and 0.315, respectively. It is assumed that the total delay time in the network depends on the moving start time. If one starts at time L1, the total delay time in this

path is equal to one, if a start at time L2, total delay is two, etc. Therefore, if the total delay time for this example is set as five, then paths 1 and 2 remain unaccepted. In Figs. 2.1-2.4, optimum paths for L7 (as the starting time) are shown by considering the delay between two nodes for each path 1-4 respectively. In addition, the state of each node is shown for each time unit. For instance, as shown in Fig. 2.1., in path 1, moving starts from node *s* to node 1 with one delay time unit, from node 1 to node 2 and from node 2 to node 5, without delay and from node 5 to node *t* with 1 delay unit.

Finding the shortest path for a network with scales shown in Figs. 2.1-2.4 is not very challenging. However, in large-scaled multi-criteria networks with a lot of arcs and nodes, finding the shortest path is much more difficult, time-consuming and the existing single-criterion algorithms such as classical Dijkstra would be inefficient. Therefore, an enhanced Dijkstra's algorithm is proposed based on the original single-criterion Dijkstra presented by Skriver and Andersen [16] for finding the optimum non-dominated path in large-scale networks with populated nodes.

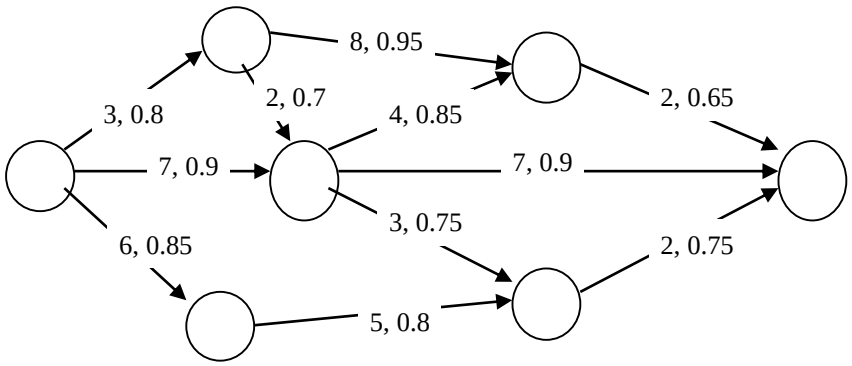


Fig 1. Given Network *G (N, A)* showing the cost and reliability of each arc

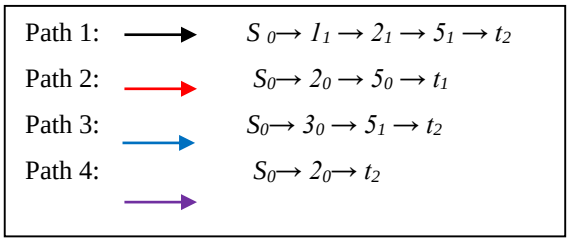
Table 1. Specifications of given network *G (N, A)*

Arc	Cost	Reliability	L1	L2	L3	L4	L5	L6	L7
s – 1	3	0.8	0	0	1	2	1	1	1
s – 2	7	0.9	0	1	2	3	2	1	0
s – 3	6	0.85	1	1	1	1	2	1	0
1- 2	2	0.7	1	1	1	1	1	1	0
1- 4	8	0.95	1	1	2	2	1	1	1
2- 4	4	0.85	1	0	1	1	2	1	0
2 – 5	3	0.75	0	0	1	1	1	0	0
2 – t	7	0.9	1	1	1	2	2	2	2
3 – 5	5	0.8	0	0	1	1	1	1	1
4 – t	2	0.65	0	0	1	2	1	1	0
5 – t	2	0.75	0	1	1	2	2	1	1

Table 2. Non-dominated Paths of given network *G (N, A)*

Non-dominated Paths	Cost	Reliability	Total Delay Time						
			L1	L2	L3	L4	L5	L6	L7

s -1 – 2 – 5 – t	10	0.315	1	2	4	6	5	3	2
s – 2 – 5 – t	12	0.506	0	2	4	6	5	2	1
s – 3 – 5 – t	13	0.51	1	2	3	4	4	3	2
s – 2 – t	14	0.81	1	2	3	5	4	3	2



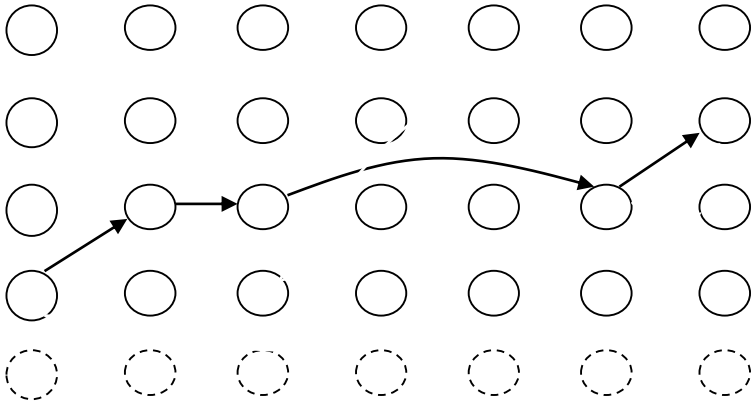
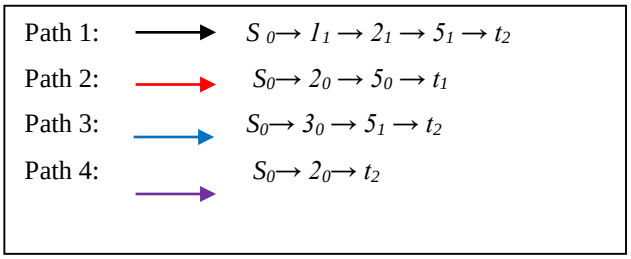


Fig 2.1. Non-dominated paths in start time L7 for path 1



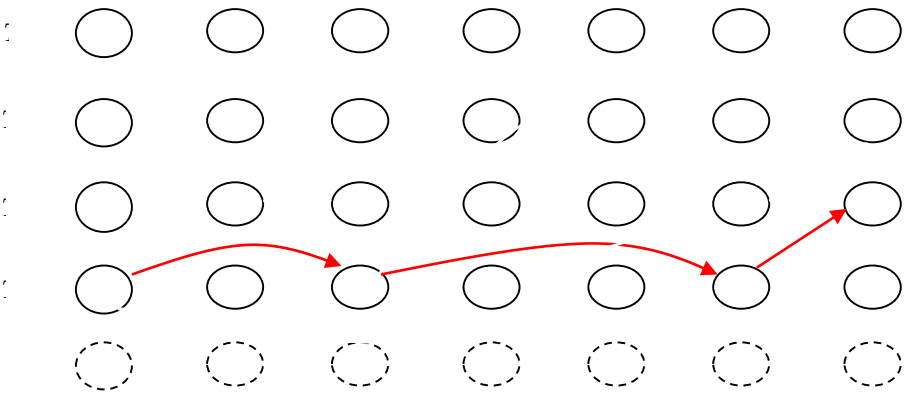


Fig 2.2. Non-dominated paths in start time L7 for path 2

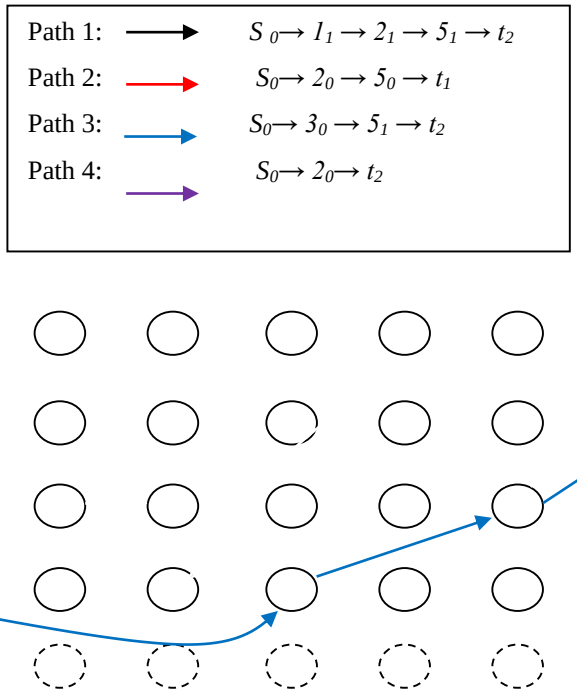


Fig 2.3. Non-dominated paths in start time L7 for path 3

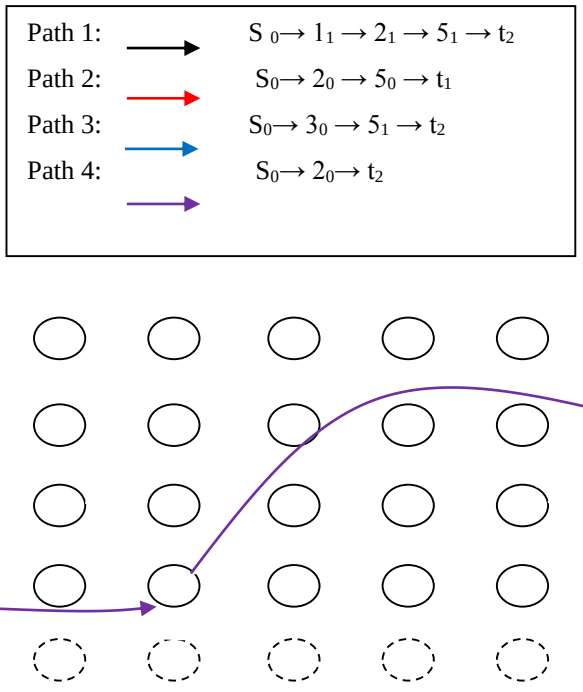


Fig 2.4. Non-dominated paths in start time L7 for path 4

Literature review

Dijkstra and labeling algorithms

Some of the simple classical algorithms focusing on single-objective shortest path problem were presented by Ford [13], Bellman [14] and Ford and Fulkerson [3] were classified as "labeling and label correcting algorithm" and therefore can be modified and used for "multi-objective shortest path problem". For instance, Brumbaugh-Smith et al., [15] presented a label correcting algorithm for bi-objective shortest path problem. In their algorithm, the cost of effective paths from the source to the current node is kept the same for each criterion. As a result, a collection of answers in the sink label, show all non-dominated paths. Skriver et al., [16], improved the previous algorithm by labeling middle nodes and finding minimum cost from middle nodes to sink. This new algorithm improved the performance time significantly [16].

Martinz et al., [1] presented a labeling algorithm for multi-objective shortest path problem (MOSP). In Martin's successive reiterations algorithm, a node that contains the smallest lexicographical vector among all nodes in a temporary collection would be selected for moving from a temporary collection to a permanent one. Since this algorithm selects the smallest lexicographical from other vectors in successive reiterations, the efficiency of the algorithm decreases when there are several effective paths from one node to the other.

However, some other researchers including Warburton, Hassin, Lorenz, Ergun et al., [17-19] tried to solve the MOSP problem in polynomial algorithms by using dynamic programming and approximate algorithms. Hao et al., [20] presented a polynomial model for MOSP that contained two algorithms. Firstly, they found all the paths for which the costs were less than an acceptable limit for each objective independently. Later, they completed their work by finding the algorithm of the K-path and finally selected the best path among the joint paths by multi-objective lattice-order decision-making algorithm.

In another collaborative work, Chen et al., [21] studied the time-dependent reliable shortest path problem (TD-RSPP). They surveyed two variants of TD-RSPP. The first problem was to obtain the earliest arrival time as relevant to the reliable shortest path for a given departure time as the "forward" TD-RSPP. The second problem was to obtain the latest departure time as relevant to the reliable shortest path for a given preferred arrival time as the "backward" TD-RSPP. They proposed two efficient solution algorithms to solve both forward and backward TD-RSPP problems.

In other research, Kwon and Lee [22] studied a robust shortest path problem. They proposed a path enumeration approach using the K-shortest paths search algorithm. They also applied their approach in hazardous materials transportation. In a similar work, Chen and Tang [23] surveyed the problem of finding the K-th shortest path for a time-schedule network. They assumed that each node in the network has a list of given departure times. They developed a new algorithm that builds a map structure at each node in the network. They found out that K-th shortest path without the first K-1 path. Then they applied their algorithm for finding multiple shortest paths in the same network.

As relevant to dynamic networks, Cheung [24] suggested a routing policy that includes a dynamic shortest path in a network with independent, positive and discrete random are costs. His approach follows the classical label-correcting approach to work out the expected path cost. Then he developed stochastic versions of some well-known label-correcting methods. They found out that fast methods for deterministic networks could become very slow for stochastic networks.

In a further research, Murthy and Her surveyed [25] the problem of determining a path between two nodes in a network that minimizes the maximum of r path length values associated with it. They proposed a label-correcting procedure for this problem and extended two pruning techniques that distinguish and eliminate many paths that were not part of the optimal path. Tufekci [26] developed decomposition algorithms for finding the shortest path between a

source node and a sink node of an arbitrary distance network and compared his algorithm with Shier's algorithm [15] and showed that his algorithm is a polynomial of O (n^2).

In a related research conducted by Xuan et al. [42], the performance of two different dynamic programming approaches including Dijkstra's algorithm and Bellman's dynamic programming approach was checked for shortest path problem of transportation road network in different backgrounds. This study can be effectively used to solve the shortest path problem for different needs. The application of Bellman's approach shows that it is computationally Costly due to a lot of duplicate calculations. In comparison, Dijkstra's algorithm can successfully make better the computational output of the backward dynamic programming approach. Pursuant to whether the shortest path from the node to the original node has been found, Dijkstra's algorithm signed the node with permanent and temporal labels. In each step, it simultaneously updates both the permanent and temporal labels to avoid the duplicated calculations in the backward dynamic programming approach. The paper also presented an algorithm using dynamic programming theory to solve the K shortest path problem. The K shortest path algorithm was particularly helpful to find the possible paths for travelers in the real-world. The computational efficiency of the three approaches in large network was prospected.

An extensive review of shortest path problem solving algorithms was presented by Kumawat et al. [43]. The paper provided a detailed overview of Shortest Path Algorithms (SPA) besides problem solving algorithms. The problem of the shortest path was represented as a graph and solved by utilizing different algorithms based on the application. The main goal of SPA was analyzed as shortening the inclusive cost and distance. The application of artificial intelligence was also described in the shortest path algorithms. The time complexity of the different algorithms presented in the various literature was discussed and suggested some future research perspectives. This examination also proved that the performance varies between various algorithms which are used to resolve detailed variations of SPP.

Dynamic multi-objective shortest path problem (MOSP) problem

One of the most comprehensive and pioneer studies in terms of dynamic network flows was conducted by Aronson in 1989 [4]. Referring to his paper, most of the researchers, when faced with a dynamic network flow problem, do not make up methods for exploiting the multi-period structure, but use existing methods and codes, mostly because their interest is the model. If a problem is small and solved infrequently, and an available network code exists, then it may be enough in terms of code development or acquisition cost and computer time. However, if the problem is large and frequently solved, then the execution of a specialized efficient starting procedure and dynamic algorithm should be considered.

The classic shortest path problem considers just one objective function or criterion that is mostly focused on minimizing the total costs or weights of the path, and therefore Dijkstra algorithm or the label-setting algorithm can be applied for solving such problems with positive value [27]. On the other hand, many shortest path problems consist of more than one criterion or objective function. Batta and Chiu [28], Current and Min [29], Current and Marsh [30] are among a few top researchers that studied the applications and characteristics of such problems. In addition, the shortest path problems in real-world problems can be categorized into different types such as static, dynamic or multi-criteria with discrete or continuous, and deterministic or stochastic parameters. In this regard, several research works investigate multi-criteria shortest path problems in static networks.

Ghoseiri et al., [30] proposed an algorithm based on multi-objective ant colony optimization (ACO) to solve the bi-objective shortest path problem. They analyzed the efficiency of the algorithm and checked the quality of solutions by comparing the results of two sets of small and large sized networks 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 were employed to measure distance, uniformity distribution, and extension of the Pareto frontiers. The results on the set of instance problems showed that the proposed ACO algorithm produced good quality non-dominated solutions and time saving in the computation of large-scale bi-objective shortest path problems.

In a novel collaborative research, Liu et al., [31] proposed a simulated annealing algorithm to figure out the multi-criteria shortest path problem as well as the multi-criteria constrained shortest path problem. Their algorithm provided an entirely new searching mechanism in the sense of a "search from a path set to another path set" instead of a 'search-from-a-point' searching mechanism. Their algorithm can be employed for nonlinear objectives. Focusing on the same area, He and Song [32] addressed the optimal path-finding problem in a stochastic time-dependent network where all link travel times were temporally and spatially co-related. In another work, Yu et al., [33] defined the concept of the shortest path based on a scale-free dynamic and stochastic network model. They proposed a temporal ant colony optimization (TACO) algorithm for searching the shortest paths in the network.

In another related work, Bezerra et al., [34] proposed an ACO algorithm, called GRACE, for the Multi-objective Shortest Path Problem. Their approach was compared to the well-known evolutionary algorithm NSGA-II. Furthermore, GRACE was also compared to another ACO algorithm proposed previously for the MSP. The results obtained from a computational experiment with eighteen instances and triple objectives for each instance showed that the proposed approach was capable of producing high quality non-dominated solutions. Another related work refers to Abbasi et al., [35] who offered a two-phased exact algorithm and a Cross-Entropy (CE) algorithm based on bi-criteria to find 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 as compared with the full numeration algorithm.

Guerriero and Pugliese considered a problem that includes cost and time as two main parameters associated with each. They proposed a multi-dimensional labeling algorithm to solve this problem and conducted several computational experiments to evaluate the proposed method. Claudio *et al*., [13] developed an evolutionary algorithm based on Monte Carlo simulation to solve multi-objective optimization problems. Their objectives were focused on simultaneous maximization of shortest path length and minimization of interdiction strategy cost. In addition, they considered the transformation of the first objective into the minimization of the reliability of the most reliable path in interdiction paths.

Reinhardt and Pisinger [36] presented a general framework for dominance tests for MOSP problems involving a number of non-additive criteria such as the probability of reaching the destination, combined distance and probability function, maximum of commissions, number of zones visited, maximum zone distance from departure, zone distance and time and modulo k penalties. In another work, Nasrabadi and Hashemi [37] presented an algorithm to figure out the dynamic flow problem in a discrete-time model in which transit times, transit costs, transit capacities, storage costs and storage capacities were considered as variable parameters that vary with time.

In 2020, Ekmen [41] performed a well theoretical and comprehensive research by analyzing ad comparing four of the most frequently used methods in the evaluation of the shortest path problem including Dijkstra, Bellman-Ford, Johnson's and Floyd-Warshall Algorithm. It was discussed which algorithm is better and more effective in finding the shortest path. The comparison of four different algorithms for positive weighted, non-directional and fully connected graphs on the same graph drawing application is remarkable because there is no study on this subject with the same specifications. According to the results of the study, it was understood that the algorithm that will be used should be selected in line with type of graph and

problem. It is proved that the Dijkstra algorithm absolutely must be used for fully connected, positive weighted and non-directional graphs.

Another novel research was conducted by Hughes et al. [44] focusing on multiple shortest path problems with path de-confliction. Their research was focused on formulating and examining the Multiple Shortest Path Problem with Path De-confliction (MSPP-PD) to balance agent routing efficiency with group vulnerability. Within the general model formulation, multiple agents were routed between respective source and terminus nodes while minimizing both the total distance travelled and a measure of path conflict, where path conflict occurs for any instance of more than one agent traversing an arc and/or node. Within this modeling structure, the paper presented a set of alternative, conceptually-motivated penalty metrics to inhibit path conflict between agents. Subsequent empirical testing over a set of synthetic instances demonstrated the effect of different penalty function metrics on both optimal solutions and the computational effort required to identify them. Finally, the utility of the MSPP-PD model variants both individually and collectively was highly recommended in real-time case studies.

Two proposed algorithms

Extended bi-criteria Dijkstra's algorithm

The classical bi-objective Dijkstra's algorithm that was conceived by computer scientist Edsger Dijkstra in 1956 and published in 1959 is a graph search algorithm that solves the single-source shortest path problem for a graph with a non-negative edge, producing a shortest path tree [3]. This algorithm is commonly used in routing problems or as a sub-routine in other graph algorithms [27]. Due to the following reasons, the performance and functions of a bi-criteria Dijkstra’s algorithm is not comparable to its single-objective model and therefore needs to be independently formulated. Some of the main conceptual differences between the two above-mentioned algorithms follow:

- Single-objective Dijkstra algorithm has only one solution, while bi-objective algorithm may have several answers (non-dominated solutions).
- In single-objective Dijkstra, between every two nodes with a temporary label, one node with minimum index (minimum cost) is selected and is constantly labeled, while in bi-objective Dijkstra, the minimum cost could not be selected due to the existence of non-dominated answers.
- In a bi-objective algorithm, a temporary and a constant label should be kept in exchange for each non-dominated answers.

Therefore, in networks with a large number of nodes, arcs and non-dominated solutions, the performance of single-criterion Dijkstra’s algorithm intensely decreases, while its running time dramatically increases. This is considered as a major drawback of the single-criterion Dijkstra algorithm to be utilized in large-scale networks. This long running time is mainly to the fact that in each step the non-dominated solutions should be calculated several times up to the number of the entering arcs. However, the working process of the proposed bi-objective algorithm (extended Dijkstra) is the same as the single-objective model. The only difference is that for generating the proposed bi-objective algorithm, the process is started by entering nodes rather than focusing on the departing (exiting) nodes.

In each step of the single-objective Dijkstra's algorithm, the selection procedure for each subsequent node is based on the minimum temporary labels of the existing nodes and this process is repeated for all the nodes. However, in extended Dijkstra that is formulated based on the entering nodes, this process is conducted once for each node and therefore yields considerable improvement on decreasing the CPU running time for large-scale networks.

Moreover, in classical bi-objective Dijkstra's algorithm, non-dominated answers should be calculated several times for each node rather than selecting minimum labels. While for extended Dijkstra's algorithm, non-dominated answers are calculated once for each node. The pseudo-code for extended Dijkstra's algorithm is given in Table 3.

Table 3. Pseudo code of the extended Dijkstra's algorithm 1

1: Input: A graph *G= (V, E)*, a set of non–negative edge weights
$\{C_{ij} : (i, j) \in E , R_{ij} : (i, j) \in E \}$, an origin $s \in V$, a destination $t \in V$.
2: Output: shortest paths from s to *t*.
3: Initialization:
M: Two-dimensional Matrix [Cost, Reliability], $N_j = 1$
4: For each child j of i do
$M_j[N_j, 1] = M_i[N_j, 1] + C_{ij}$
$M_j[N_j, 2] = M_i[N_j, 2] + R_{ij}$
$N_j = N_j + 1$
end for
5: find and save non-dominated paths (optimal paths) for *j*
6: if *j=t* then
terminate
else
go to step 4
end if

For networks in which between every two nodes there is a delay time, the moving start time is very important and needs to be carefully selected to avoid unwanted delays. Different starting (departure time) plays a key role in determining the number of the optimum path as well as the total delay time to get to the destination point. Table 4 shows a network with 15 nodes and 28 arcs that have been examined at seven different start times. For each optimum path in this network, the values for cost and reliability are calculated based on different start times (*L1 to L7*) in respective columns.

By considering a permissible level for the delay in this network (10-time units for this case), the number of possible optimum paths become various. In Table 4, each optimum path is indicated by referring to the number of nodes and the relevant delay time in the parenthesis. Three network sizes (small, medium and large) are utilized for comparing the performance of the two algorithms, i.e., the classical bi-objective Dijkstra (based on exiting nodes) and the extended Dijkstra's algorithm (based on entering nodes).

The performance results of the two algorithms (classical vs. extended) are compared in Table 5 and then graphically presented in Fig. 3. Based on the results, for small networks, the two algorithms yielded almost the same performance in CPU time, while for medium and large-scale networks (especially those having 4000 nodes and 150000 arcs), the extended algorithm has a dominantly better performance as compared to the classical bi-objective Dijkstra. Therefore, the extended Dijkstra's algorithms have been proved to have a better performance as compared to the classical bi-objective algorithms.

Complexity Analysis of Dijkstra Algorithm

Let *G (V, E)* be a directed graph in which each vertex has a nonnegative weight. The cost of a path between two vertices in *G* is the sum of the weights of the vertices on that path. It has been shown that for such graphs, the time complexity of Dijkstra's algorithm (E.W. Dijkstra, 1959) implemented with a binary heap, is calculated as *O (|E|+|V| log |V|).*

The complexity of Dijkstra's algorithm is as follows. It takes two *(|V|)* times to construct the initial priority queue of |*V*| vertices. Each of the subsequent priority queue operations takes time two (*log q*) where q is the current size of the queue. Each vertex u is deleted from the queue exactly once after it has obtained its least cost path from the source vertex. After u is deleted from the queue, each neighbor *v* of vertex *u* is tested to see if the path from the source to v through u has a lower cost than the current path from the source to *v*. If a lower cost path is obtained through u, then the path cost for *v* is decreased and the vertex priority changed in the queue. Therefore, the test for improving a path is performed two *(|E|)* times with a worst-case time of two *(log |V|)* to update the vertex priority for each test.

Consequently, the algorithm runs in time *2(|E| log |V|).* We know that for vertex-based cost functions, the test for improving the cost of the path succeed only once per vertex when the neighbor with the least cost path from the source is deleted from the queue. Let *c*(v)* denote the cost of the least cost path from the source vertex *s* to vertex *v* using f. Let u be the immediate predecessor on the least cost path from *s* to *v* using *f*. Then, *c*(v) = f (v) + c*(u).* By the definition of least cost paths, *c*(u) ≤ c*(u¢)* for all *u¢* and refer to *Q3* and *Q3* adjacent to *v*. Since vertices are deleted in path cost order without loss of generality, *u* is deleted before *u¢*. Although *u¢* might be deleted before *v*, the path cost of a vertex *v* will never decrease after the first time when its least path cost neighbor *u* is deleted from the queue. In other words, when a vertex obtains a finite path cost, it obtains its least path cost. Of the two (|*E*|) tests for a lower path cost, only two (|*V*|) tests require an update of vertex priority in the queue. Therefore, the complexity of Dijkstra's algorithm for vertex-based cost functions is two *(|E| + |V| log |V|)* using a binary heap implementation for the priority queue.

Table 4. Non-dominated Paths

Start time	#Non-dominated Paths	Dijkstra's CPU Times(s)		Cost	Reliability	Total of Delay time	Optimum Paths
		Classical	Extended				
L1	7	0.04	0.12	1648	0.581	1	1(0)-3(0)-5(0)-7(0)-8(0)-9(0)-11(1)-12(1)-14(1)-15(1)
				1839	0.607	0	1(0)-3(0)-5(0)-7(0)-8(0)-10(0)-12(0)-14(0)-15(0)
				1913	0.621	1	1(0)-3(0)-5(0)-7(0)-8(0)-9(0)-11(1)-13(1)-15(1)
				2430	0.632	1	1(0)-3(0)-4(0)-6(0)-8(0)-9(0)-11(1)-13(1)-15(1)
				2461	0.651	0	1(0)-2(0)-4(0)-6(0)-8(0)-10(0)-12(0)-14(0)-15(0)
				2535	0.667	1	1(0)-2(0)-4(0)-6(0)-8(0)-9(0)-11(1)-13(1)-15(1)
				2914	0.674	0	1(0)-2(0)-4(0)-6(0)-8(0)-9(0)-10(0)-12(0)-14(0)-15(0)
L2	6	0.04	0.12	1648	0.581	5	1(0)-3(0)-5(0)-7(0)-8(2)-9(2)-11(2)-12(2)-14(3)-15(5)

Start time	#Non-dominated Paths	Dijkstra's CPU Times(s)		Cost	Reliability	Total of Delay time	Optimum Paths
		Classical	Extended				
L3	5	0.04	0.12	1839	0.607	6	1(0)-3(0)-5(0)-7(0)-8(2)-10(2)-12(3)-14(4)-15(6)
				1913	0.621	3	1(0)-3(0)-5(0)-7(0)-8(2)-9(2)-11(2)-13(3)-15(3)
				2430	0.632	2	1(0)-3(0)-4(1)-6(1)-8(1)-9(1)-11(1)-13(2)-15(2)
				2535	0.667	2	1(0)-2(1)-4(1)-6(1)-8(1)-9(1)-11(1)-13(2)-15(2)
				2914	0.674	5	1(0)-2(1)-4(1)-6(1)-8(1)-9(1)-10(1)-12(2)-14(3)-15(5)
				1648	0.581	10	1(0)-3(1)-5(3)-7(5)-8(5)-9(7)-11(7)-12(9)-14(10)-15(10)
				1839	0.607	10	1(0)-3(1)-5(3)-7(5)-8(5)-10(7)-12(9)-14(10)-15(10)
				1913	0.621	8	1(0)-3(1)-5(3)-7(5)-8(5)-9(7)-11(7)-13(7)-15(8)
				2461	0.651	9	1(0)-2(2)-4(2)-6(4)-8(4)-10(6)-12(8)-14(9)-15(9)
				2535	0.667	7	1(0)-2(2)-4(2)-6(4)-8(4)-9(6)-11(6)-13(6)-15(7)
				1648	0.581	8	1(0)-3(0)-5(1)-7(2)-8(2)-9(4)-11(5)-12(6)-14(8)-15(8)
L4	4	0.04	0.12	1839	0.607	4	1(0)-3(0)-5(1)-7(2)-8(2)-10(2)-12(2)-14(4)-15(4)
				2461	0.651	7	1(0)-2(2)-4(2)-6(3)-8(5)-10(5)-12(5)-14(7)-15(7)
				2914	0.674	9	1(0)-2(2)-4(2)-6(3)-8(5)-9(7)-10(7)-12(7)-14(9)-15(9)
				1648	0.581	6	1(0)-3(0)-5(0)-7(0)-8(2)-9(3)-11(4)-12(4)-14(4)-15(6)
				1839	0.607	7	1(0)-3(0)-5(0)-7(0)-8(2)-10(3)-12(5)-14(5)-15(7)
L5	6	0.04	0.12	2430	0.632	4	1(0)-3(0)-4(0)-6(1)-8(1)-9(2)-11(3)-13(4)-15(4)
				2461	0.651	6	1(0)-2(0)-4(0)-6(1)-8(1)-10(2)-12(4)-14(4)-15(6)
				2535	0.667	4	1(0)-2(0)-4(0)-6(1)-8(1)-9(2)-11(3)-13(4)-15(4)
				2914	0.674	8	1(0)-2(0)-4(0)-6(1)-8(1)-9(2)-10(4)-12(6)-14(6)-15(8)
				1648	0.581	8	1(0)-3(2)-5(2)-7(3)-8(3)-9(5)-11(5)-12(6)-14(8)-15(8)
L6	6	0.04	0.12	1839	0.607	6	1(0)-3(2)-5(2)-7(3)-8(3)-10(3)-12(4)-14(6)-15(6)
				1913	0.621	6	1(0)-3(2)-5(2)-7(3)-8(3)-9(5)-11(5)-13(5)-15(6)

Start time	#Non-dominated Paths	Dijkstra's CPU Times(s)		Cost	Reliability	Total of Delay time	Optimum Paths
		Classical	Extended				
L7	7	0.04	0.12	2430	0.632	6	1(0)-3(2)-4(2)-6(3)-8(3)-9(5)-11(5)-13(5)-15(6)
				2461	0.651	6	1(0)-2(1)-4(2)-6(3)-8(3)-10(3)-12(4)-14(6)-15(6)
				2535	0.667	6	1(0)-2(1)-4(2)-6(3)-8(3)-9(5)-11(5)-13(5)-15(6)
				1648	0.581	4	1(0)-3(0)-5(1)-7(1)-8(1)-9(1)-11(2)-12(4)-14(4)-15(4)
				1839	0.607	3	1(0)-3(0)-5(1)-7(1)-8(1)-10(2)-12(3)-14(3)-15(3)
				1913	0.621	4	1(0)-3(0)-5(1)-7(1)-8(1)-9(1)-11(2)-13(4)-15(4)
				2430	0.632	5	1(0)-3(0)-4(1)-6(1)-8(2)-9(2)-11(3)-13(5)-15(5)
				2461	0.651	5	1(0)-2(0)-4(2)-6(2)-8(3)-10(4)-12(5)-14(5)-15(5)
				2535	0.667	6	1(0)-2(0)-4(2)-6(2)-8(3)-9(3)-11(4)-13(6)-15(6)
				2914	0.674	5	1(0)-2(0)-4(2)-6(2)-8(3)-9(3)-10(4)-12(5)-14(5)-15(5)

Table 5. Comparison of CPU Time for classical bi-objective Dijkstra vs. extended Dijkstra

#Problem	# Node	#Arc	Maximum allowable delay time	#Non-dominated solutions	CPU time(s)	
					Classical Bi-objective Dijkstra	Extended Bi-objective Dijkstra
1	25	70	10	6	0.22	0.04
2	50	207	10	15	0.18	0.07
3	100	617	10	26	1.42	0.2
4	250	2005	10	29	19.14	1.12
5	500	4434	10	48	229	2.7
6	750	7641	10	43	1056	7.28
7	1000	11925	10	39	3047	10.29
8	1500	24017	10	42	8038	19
9	2000	38500	10	85	14029	132
10	2250	48195	10	97	19120	167
11	2500	58881	10	104	26291	225
12	2750	70612	10	115	>30000	341
13	3000	83335	15	109	>30000	302
14	3500	112432	15	110	>30000	481
15	3750	128784	15	94	>30000	501
16	4000	146212	20	105	>30000	545
17	4250	164141	20	121	>30000	684
18	4500	183533	20	111	>30000	829

19	4750	204190	20	98	>30000	1016
20	5000	226075	20	121	>30000	1368

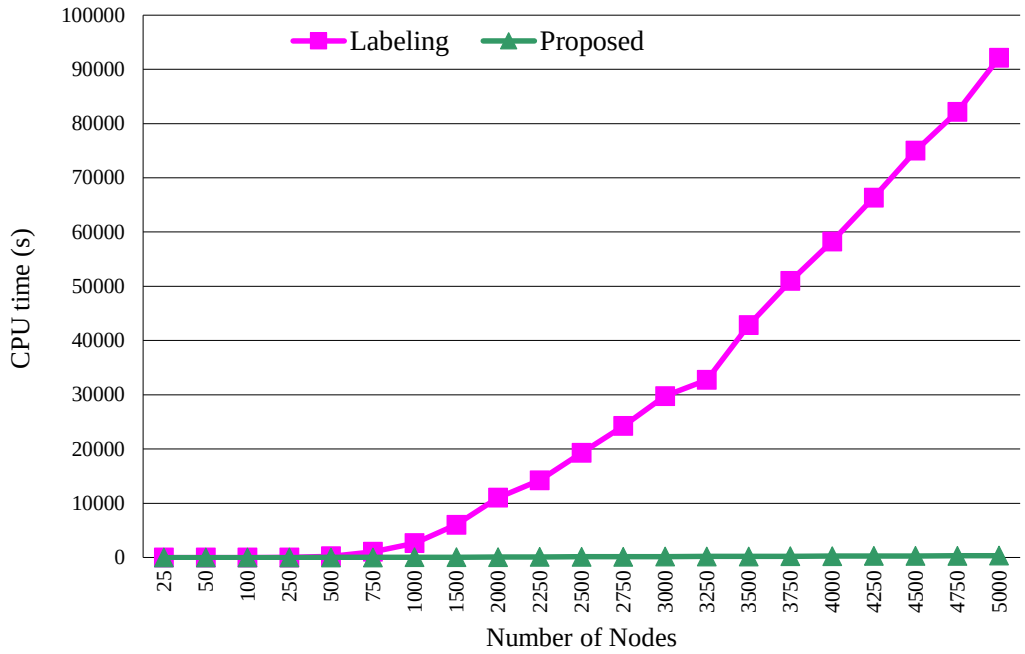


Fig 3. The CPU time of the extended Dijkstra (Proposed) versus the classical bi-objective Dijkstra (Label)

Proposed Ant Colony algorithm

Ant colony optimization (ACO) was presented for setting the travelling salesman problem (TSP) by Dorigo et al., in 1991 as the pioneer in the field. This optimization method has been derived from various ants' behaviors while they were looking for the shortest path among possible paths for finding food resources [38]. Any optimization problem that seeks to find the shortest path is based on the ACO application and may be used for the following purposes [39]:

- Routing inside and between towns
- Routing between substations of power distribution networks with high voltage
- Routing of computer networks

As it was mentioned earlier in the literature review, in recent years many researchers have designed algorithms based on ACO. Since this algorithm has a good speed in creating possible solutions, it has been used as an alternative of exact methods to set the bi-objective problems. In this regard, the bi-objective ant colony optimization can be divided into three categories [39, 40]:

1- Algorithms that apply several colonies for each objective
2- Algorithms that apply several matrixes; the traces of pheromone for each objective
3- Algorithms that apply several creative parameters for each objective

In this paper, the proposed ant colony algorithm is considered to set the problem of the shortest combinational bi-objective path and belongs to the first and third ACO categories explained above. The proposed ACO algorithm is fully presented in this section.

In this problem, the reliability (R) criterion and cost (C) criterion are profit and loss type objectives, respectively. Since the bounds of these two criteria are not similar, Eq. 4 is applied for equalizing their scales and to convert reliability criterion to cost criterion ($\frac{1}{C}$) bounds.

$$R' = \frac{R - Min_R}{Max_R - Min_R} * \frac{Max_C - Min_C}{Max_C * Min_C} + \frac{1}{Max_C} \tag{4}$$

Wherein,
$\boldsymbol{R'}$**:** Equivalent reliability based on cost criterion bounds
$\boldsymbol{R}$ **:** Reliability amount for each arc
$\boldsymbol{Min_R}$ **:** The minimum amount considered for reliability criterion.
$\boldsymbol{Max_R}$: The maximum amount considered for reliability criterion.
$\boldsymbol{Min_C}$ **:** the minimum amount considered for cost.
$\boldsymbol{Max_C}$ **:** the maximum amount considered for cost in this problem.

To start we put an ant at the start of the first node. Then, the ant must select a node out of the selectable nodes. Since the ant has no knowledge of the paths, it should randomly select a path. However, instead of a random selection, the ant can be helped to select better paths by referring to our knowledge about networks. In this problem, a better path is marked as the one that has a lower cost (C) and higher reliability (R) and so the greater R/C rate indicates a better Path to be selected. Therefore, the artificial ants could be helped to select these paths with more probability.

Eq. 5 indicates the probability of selecting the node $\boldsymbol{j}$ for the ant located at node $\boldsymbol{i}$ from existing paths considering the network criteria that is called motion attractiveness $\left(\eta_{\text{ij}} = \frac{R_{ij}}{C_{ij}}\right)$. In this case the ant is able to select almost cleverly instead of a completely random selection.

$$P_{ij}^{x} = \begin{cases} \frac{\frac{R_{ij}}{C_{ij}}}{\sum_{i \epsilon N_i} \frac{R_{il}}{C_{il}}} & if\ j \epsilon N_i \\ 0, & otherwise \end{cases} \tag{5}$$

According to Eq. 6, after moving from node i to node j, each ant leaves some pheromone on this path so that the next ant could be helped by the present pheromone amount on the path and will be able to select the right path based on motion attractiveness.

Wherein τ_{ij} stands for the amount (intensity) of existing pheromone on the arc (i, j) and ρ stands for a constant amount between zero and one

On the other hand, Eq. 7 shows the possibility of selecting the node j for the ant located at node i of existing paths, considering the amount of present pheromone on the path. By natural steps of the algorithm considering the paltry amount of pheromone, the possibilities of selecting the paths are equal.

$$\tau_{ij}^{new} = \tau_{ij}^{old} + \rho . \tau_{ij}^{old} \tag{6}$$

$$P_{ij}^{y} = \begin{cases} \frac{\tau_{ij}}{\sum_{i \epsilon N_i} \tau_{ij}} & if\ j\ \epsilon\ N_i \\ 0, & otherwise \end{cases} \tag{7}$$

When an ant arrives at the final node, some of the remained pheromone on the path will be evaporated according to Eq. 8, where φ stands for evaporation rate of a constant amount between zero and one.

$$\tau_{ij}^{new} = \varphi . \tau_{ij}^{old} \quad (8)$$

Considering the motion attractiveness and the effect of pheromone, by combining Eqs. 5and 7, the ants could be helped to select better paths with more possibility. Eq. 9 below is the final one to select the next node by ants.

$$P_{ij}^{k} = \begin{cases} \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{i \epsilon N_i^k} (\tau_{il})^{\alpha}(\eta_{il})^{\beta}} & if\ j \epsilon\ N_i^k \\ 0, & otherwise \end{cases} \quad (9)$$

Wherein,

P_{ij}^{k} : possibility of selecting the node j at the node i by ant - k^{th}

η_{ij}: attractiveness motion from node i to node j

α**:** is a constant amount for weighting motion attractiveness. The greater this amount is the more the effect of motion attractiveness and randomness become.

β: is a constant amount that causes to intensify the pheromone concentration. The greater this amount is, the more the importance of pheromone concentration and the less the effects of random behavior becomes.

τ_{ij}: the amount (intensity) of pheromone on the path (i, j)

N_i^k: the total selectable arcs for the ant-k^{th} located at the node i.

According to Eq. 10, after a colony of ants arrives at the final node, the optimum non-dominated paths will be updated by pheromone spilling. Wherein Δ indicates the rate of general pheromone spilling and is a constant number determined by considering the rest of the problem criteria. For a better path, the amount of pheromone spilling will be more. This act is reiterated until all considered colonies would find optimum paths. The pseudo-code of the proposed bi-criteria ant colony algorithm is given in Table 6.

$$\tau_{ij}^{new} = \tau_{ij}^{old} + \rho . \tau_{ij}^{old} + \Delta . \frac{R_{ij}}{C_{ij}} \quad (10)$$

Table 6. Pseudo-code of proposed bi-criteria Ant colony algorithm

Input: A graph *G= (V, A)*, a set of non–negative edge weights $\{c_{ij}, R_{ij}: (i,j) \epsilon A\}$,

An origin $s \epsilon V$, a destination $t \epsilon V$.

Output: shortest paths from *s* to *t*.

Initialization:

N:Node, Iteration, Ant, *α, β, ρ,φ*

For *i*=1 to Iteration

For *j*=1 to Ant

Node=1

While Node<*N*

Calculate Heuristic Function

Select Next Node

Local Update Pheromone

End While

Crate Path by Ant *j*

Calculate Evaporation Rate
End for *j*
Select and save Non-Dominate Paths in Iteration *i*
Global Update pheromone
End for *i*
Select Non-Dominate Paths in all Iteration

Computational results and discussion

The algorithms were created in the MATLAB code on a Microsoft Windows 7.0 Professional with 2.3GB RAM and 3GB swap running Digital Intel Core i5 Duo CPU. As shown in Fig. 4, a flowchart for the steps involved in the algorithm is presented. In order to check for the efficiency and validity of the algorithms, some experiments were conducted. Three sets of small, medium and large-sized problems were generated randomly using the suggested algorithm and then results were compared to solutions produced by the proposed multi-objective algorithm based on classical bi-objective Dijkstra.

A set of nine small-sized acyclic network problems, ranging from 20 to 100 nodes with positive integer values ranging from 0.90 to 1.0 and from 100 to 500 were randomly generated with uniform distribution for reliability and cost, respectively. Later, the generated problems were solved and the results were compared to the solutions produced by the proposed multi-objective algorithm based on Dijkstra's algorithm. Table 7 shows specifications of the small-sized instance that ranges from 20 to 100 and summarizes the comparison results based on parameter values including #ants (number of ants in the colony) and #iterations (number of algorithm iterations) that is set to 100. Other ant colony parameters including α, B, ρ, φ and Δ are also equal to 0.5, 0.9, 0.1, 0.98, and 25, accordingly.

In addition to small-scale networks mentioned above, the efficiency of the algorithms was examined by a set of 32 medium-scale and large-scale acyclic networks ranging from 500 to 10000 nodes that were randomly generated. Tables 8 and 9 show the necessary specifications of the instances and summarize the comparison results for medium and large scale instances, respectively. For medium-scale instance problems, 15 acyclic networks ranging from 500-1900 were considered and for large-sized networks that took very long CPU time (all the non-dominated solutions generated from multi-objective Dijkstra's algorithm requires very long process), 17 instances ranging from 2000-10000 nodes were randomly generated and solved as illustrated in Table 9. All parameters values including #ants (number of ants in the colony) and #iterations (number of algorithm iterations) were the same as the ACO model. Other ant colony parameters including α, B, ρ, φ and Δ were also equal to 0.5, 0.9, 0.1, 0.98 and 25, accordingly.

Table 7. Specifications of the small sized instance problems

# **Problem**	#Node	# Arc	#Non-Dominated Solution		CPU Time (s)	
			Extended Dijkstra	Proposed ACO	Extended Dijkstra	Proposed ACO
1	20	71	7	7	0.01	5.83
2	30	157	16	13	0.02	6.67
3	40	267	12	10	0.02	7.04
4	50	396	15	10	0.04	7.60
5	60	544	12	8	0.05	7.82
6	70	677	13	9	0.07	8.20

7	80	837	18	10	0.08	8.36
8	90	984	18	10	0.12	8.75
9	100	1,171	17	8	0.13	9.13

Table 8. Summary of results for the experimental analysis on medium-scaled instance problems

#Problem	#Node	# Arc	#Non-Dominated Solution		CPU Time (*s*)	
			Extended Dijkstra	Proposed ACO	Extended Dijkstra	Proposed ACO
1	500	6,781	46	34	18	42
2	600	8,688	44	32	24	45
3	700	10,645	44	32	32	49
4	800	12,493	42	29	37	53
5	900	14,429	48	33	29	56
6	1000	16,294	43	29	47	61
7	1100	18,272	41	29	56	65
8	1200	20,255	43	32	60	68
9	1300	22,199	38	27	65	72
10	1400	24,144	41	29	68	75
11	1500	26,081	40	27	72	79
12	1600	27,992	48	36	85	83
13	1700	30,065	44	32	87	86
14	1800	31,832	45	32	82	90
15	1900	33,875	43	30	98	93

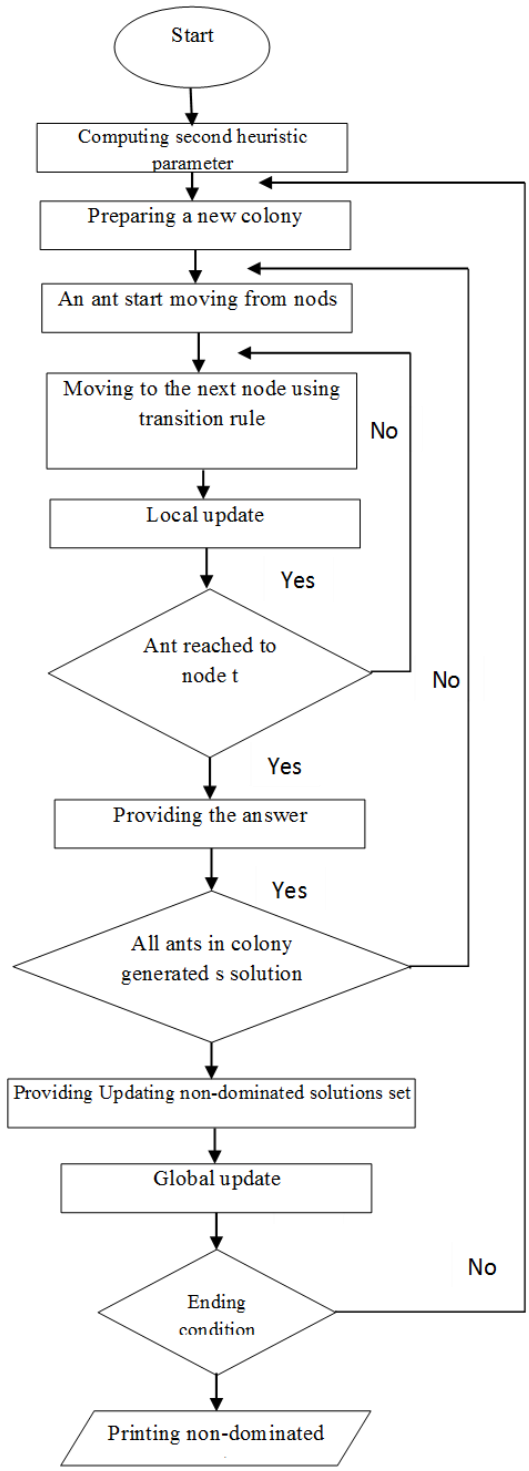


Fig 4. Flowchart of the bi-criteria proposed ACO algorithm

Tables 8 and 9 presents the performance of the bi-criteria ACO algorithm for medium and large-sized problems. However, there are parameters in the ACO algorithm that needs to be elaborated more on how to set them in the ACO algorithm. The number of iterations (colonies) in all runs is equal to 100. The weighting parameters α, β and the updating parameters φ *and* ρ were experimentally set. In the first run of the algorithm, the parameters had respective values of 0.55, 0.95, 0.99 and 0.97 that were changed in the second and third run so that better quality solutions were obtained.

Table 9. Summary of results for the experimental analysis on large-scaled instance problems

Problem	#Node	# Arc	Ant Colony Algorithm Parameters							#Non-Dominated Solution		CPU Time (s)	
			#Iteration	#Ant	α	β	ρ	φ	Δ	Extended Dijkstra	Proposed ACO	Extended Dijkstra	Propos ACO
1	2000	35,862	100	100	0.5	0.9	0.1	0.98	25	41	23	90	81
2	2500	47,181	100	100	0.5	0.9	0.1	0.98	25	46	26	135	98
3	3000	60,833	100	100	0.5	0.9	0.1	0.98	25	47	27	177	116
4	3500	76,650	100	100	0.5	0.9	0.1	0.98	25	32	20	197	141
5	4000	94,752	100	100	0.5	0.9	0.1	0.98	25	52	31	270	174
6	4500	114,792	100	100	0.5	0.9	0.1	0.98	25	32	18	282	207
7	5000	137,607	100	100	0.5	0.9	0.1	0.98	25	46	30	356	226
8	5500	162,929	100	100	0.5	0.9	0.1	0.98	25	44	27	433	252
9	6000	190,553	100	100	0.5	0.9	0.1	0.98	25	43	24	480	289
10	6500	219,978	100	100	0.5	0.9	0.1	0.98	25	57	34	589	328
11	7000	252,191	100	100	0.5	0.9	0.1	0.98	25	44	26	661	371
12	7500	286,768	100	100	0.5	0.9	0.1	0.98	25	67	39	907	423
13	8000	323,563	100	100	0.5	0.9	0.1	0.98	25	44	24	939	476
14	8500	362,764	100	100	0.5	0.9	0.1	0.98	25	57	32	1241	549
15	9000	403,588	100	100	0.5	0.9	0.1	0.98	25	51	30	1303	605
16	9500	447,426	100	100	0.5	0.9	0.1	0.98	25	61	36	1352	642

Since ACO is a stochastic algorithm, ten runs for each set of parameters were made and the set of parameters that yielded the best solutions were $\alpha = 0.5$, $\beta = 0.9$, $\rho = 0.1$, $\varphi = 0.98$, $\Delta = 25$. To study the variability, each problem is repeated 10 times. In Tables 8 and 9, #Nodes and #Arcs denote the number of nodes and arcs in the network, respectively. The number of ants and iterations are considered constant. In addition, the CPU denotes the average CPU time in seconds.

Fig. 5 shows the CPU time of the proposed ACO algorithm versus extended Dijkstra's algorithm. As it is shown, the ACO algorithm is much more suitable for solving large-scaled problems in terms of the required CPU time that proves a 100% preference of the proposed ACO as compared to extended Dijkstra's algorithm. For example, at 6000 nodes, the extended Dijkstra's algorithm takes about 480s, while the proposed ACO algorithm requires the same running time for 8000 nodes.

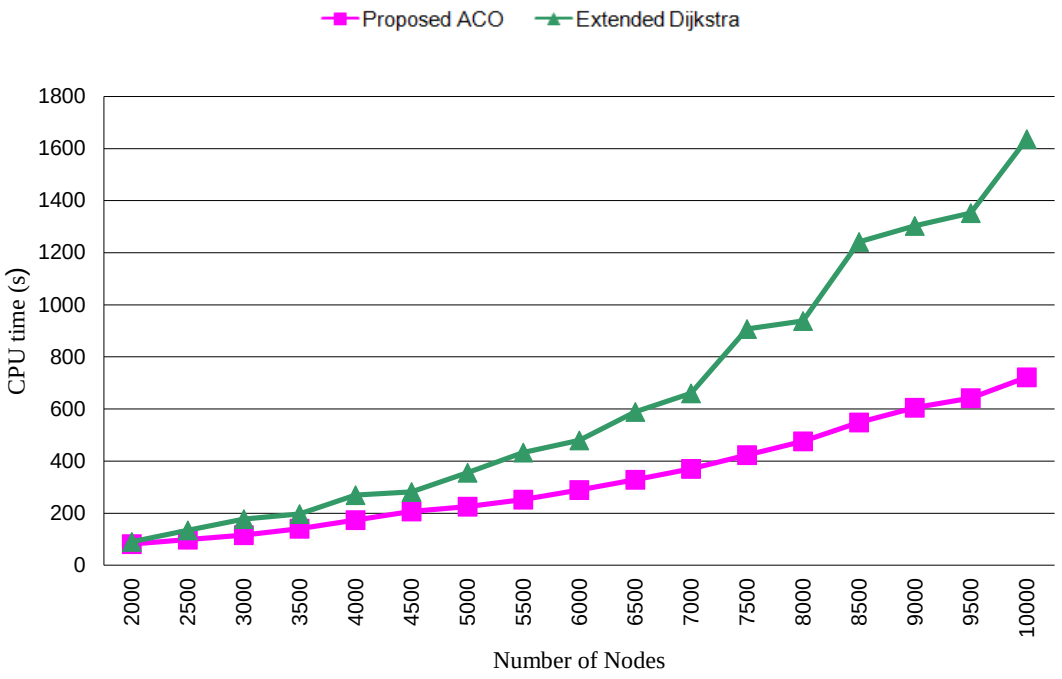


Fig 5. The CPU time of the Proposed ACO algorithm versus Extended Dijkstra algorithm

Furthermore, the comparison of three performance metrics including error ratio (%), spacing metric (Υ) and diversity metric (Delta) is given in Tables 10 and 11 for small-scale and medium to large-scale networks, respectively. From the results, it can be inferred that for small and medium-sized networks, the metric for error ratio presents a satisfactory discriminating performance for most of the instances, while for large-scale networks, the value for error ratio increases as the number of nodes goes up from 2500-5000. And so, the preliminary evaluation doubts the preference of error ratio as a well determining parameter for large-scaled networks. Furthermore, the average value for spacing metric implies that the extended Dijkstra has a better performance as compared to the proposed ACO algorithm for almost all network sizes and therefore this metric can be recommended as a determining metric for studies for future studies as well. On the other hand, for the diversity metric (Δ), it can be concluded that using either of the algorithms, the diversity metric has generated almost the same results for small-scaled networks, while for medium and large scaled networks, the extended Dijkstra has generated more desirable values as compared to the proposed ACO that means a better quality solution. Therefore, there is no guaranty that for all sets of parameters, the ACO algorithm will end with a dominant solution. Therefore, the overall insight is that as far as the currents performance metrics are involved, there seems to be a meaningful correlation (negative and or positive) between the size of the network and the performance of the proposed algorithms that can be taken into consideration for future researches.

Table 10. Summary of performance metrics for small-scaled networks

Spacing Metric(Υ)		Diversity Metric(Δ)		Error ratio (%)	# Non-dominated Solutions		#Nodes
Extended Dijkstra	Proposed ACO	Extended Dijkstra	Proposed ACO		Proposed ACO	Extended Dijkstra	
0.13	0.13	0.69	0.68	0	7	7	20

0.06	0.06	0.48	0.47	19	13	16	30
0.06	0.04	0.36	0.29	17	10	12	40
0.07	0.08	0.61	0.58	33	10	15	50
0.17	0.23	0.84	0.81	33	8	12	60
0.05	0.12	0.47	0.63	31	9	13	70
0.08	0.10	0.71	0.55	44	10	18	80
0.06	0.11	0.56	0.61	44	10	18	90
0.06	0.10	0.60	0.52	53	8	17	100

Table 11. Summary of performance metrics for medium and large scaled networks

Spacing Metric(Υ)		Diversity Metric(Δ)			# Non-dominated Solutions			
Extended Dijkstra	Proposed ACO	Extended Dijkstra	Proposed ACO	Error ratio (%)	Proposed ACO	Extended Dijkstra	#Instance	#Nodes
0.04	0.12	0.50	0.79	38	13	21	1	500
0.05	0.12	0.60	0.89	41	19	32	2	
0.05	0.08	0.71	0.67	49	19	37	3	
0.05	0.06	0.48	0.60	27	19	26	4	
0.05	0.10	0.71	0.89	42	22	38	1	2500
0.06	0.09	0.68	0.72	39	19	31	2	
0.05	0.08	0.74	0.73	39	23	38	3	
0.03	0.06	0.58	0.62	42	22	38	4	
0.05	0.07	0.76	0.76	51	25	51	1	5000
0.04	0.07	0.72	0.78	48	28	54	2	
0.04	0.10	0.65	0.80	54	21	46	3	
0.04	0.11	0.62	0.71	57	20	46	4	

Conclusion

This paper was mainly focused on solving the shortest path problem in discrete-time dynamic networks based on bi-criteria of time and reliability by considering the effect of delay times. Two objectives were determined for the bi-criteria network with delay time in each of the nodes. The first criterion maximizes the reliability of paths and the second deals with obtaining all the feasible shortest paths. The extended Dijkstra's algorithm was presented to solve the problem and the results were compared to classical bi-criteria Dijkstra's algorithm. The computational results showed that for problems of three sizes, the required CPU time of classical bi-objective Dijkstra's algorithm grows intensively when the size of the problem increases, whereas, for the extended algorithm, the running time was increased very smoothly. Later, the proposed ACO algorithm was presented to obtain the non-dominated paths through which 32 generated random instances were solved. To show the efficiency and validity of the proposed ACO algorithm, the new algorithm based on Dijkstra's algorithm was extended to generate the non-dominated paths. The computational results showed that for large-scale networks with a constant number of ants, the CPU time of the ACO algorithm was smaller than extended Dijkstra's algorithm. While, for small and medium-sized problems, the extended Dijkstra had a better performance than the ACO algorithm, considering the running time. From the results in Table 9, it can be inferred that the proposed ACO algorithm dominates the extended Dijkstra for large and too large-scaled problems, while for small and medium-sized networks, the extended Dijkstra yields more

preferable results, at least for the CPU time. As part of future research, it is well recommended to compare the results obtained from the proposed meta-heuristics with other shortest path problems rather than Dijkstra. Furthermore, it is worth considering the effect of interdicted arcs for prospective studies as well as running a sensitivity between the size of the network and the performance of the proposed algorithms based on the current performance metrics. Finally, as an upcoming priority, it is also possible to consider other combinations of parameters such as total travel time and cost or time and reliability, etc. as target bi-objective and investigate the effect of parameters change on the performance of the network.

References

[1] E. Q. V. Martins, "On a multicriteria shortest path problem," *European Journal of Operational Research,* vol. 16, pp. 236-245, 1984.
[2] M. Skutella, "An introduction to network flows over time," in *Research Trends in Combinatorial Optimization*, ed: Springer, 2009, pp. 451-482.
[3] L. Ford and D. R. Fulkerson, *Flows in networks* vol. 1962: Princeton Princeton University Press, 1962.
[4] J. E. Aronson, "A survey of dynamic network flows," *Annals of Operations Research,* vol. 20, pp. 1-66, 1989.
[5] B. Kotnyek, "An annotated overview of dynamic network flows," 2003.
[6] F. A. Aloul, B. Al-Rawi, and M. Aboelaze, "Routing in Optical and Non-Optical Networks using Boolean Satisfiability," *JCM,* vol. 2, pp. 49-56, 2007.
[7] A. Dwivedi, R. Srivastava, P. Kalra, and Y. Singh, "Simplified neural network architecture for shortest path planning in optical network," in *TENCON 2008-2008 IEEE Region 10 Conference*, 2008, pp. 1-4.
[8] T. Erlebach and S. Stefanakos, "Wavelength conversion in shortest-path all-optical networks," *Lecture notes in computer science*, pp. 595-604, 2003.
[9] D. B. A. Teixeira, C. T. Batista, A. J. F. Cardoso, and J. d. S. Araújo, "A Genetic Algorithm Approach for Static Routing and Wavelength Assignment in All-Optical WDM Networks," in *Portuguese Conference on Artificial Intelligence*, 2017, pp. 421-432.
[10] U. Bhanja and D. Mishra, "Quality of service aware fuzzy dynamic routing and wavelength assignment technique in all optical networks," *Photonic Network Communications*, pp. 1-15, 2017.
[11] E. Gajendran, M. Pradeep, and S. B. Prabhu, "Systematic Analysis of Congestion Control in WDM Mesh Networks," *Asian Journal of Applied Science and Technology (AJAST),* vol. 1, p. 1, 2017.
[12] A. K. Pradhan, S. Keshri, K. Das, and T. De, "A heuristic approach based on dynamic multicast traffic grooming in WDM mesh networks," *Journal of Optics,* vol. 46, pp. 51-61, 2017.
[13] L. Ford Jr and D. R. Fulkerson, "Solving the transportation problem," *Management Science,* vol. 3, pp. 24-32, 1956.
[14] R. Bellman, "On a routing problem," DTIC Document1956.
[15] J. Brumbaugh-Smith and D. Shier, "An empirical investigation of some bicriterion shortest path algorithms," *European Journal of Operational Research,* vol. 43, pp. 216-224, 1989.
[16] A. J. Skriver and K. A. Andersen, "A label correcting approach for solving bicriterion shortest-path problems," *Computers & Operations Research,* vol. 27, pp. 507-524, 2000.

[17] A. Warburton, "Approximation of Pareto optima in multiple-objective, shortest-path problems," *Operations research*, vol. 35, pp. 70-79, 1987.
[18] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations research*, vol. 17, pp. 36-42, 1992.
[19] D. H. Lorenz and D. Raz, "A simple efficient approximation scheme for the restricted shortest path problem," *Operations Research Letters*, vol. 28, pp. 213-219, 2001.
[20] G. Hao, D. Zhang, and X. Feng, "Model and algorithm for shortest path of multiple objectives," *Journal of Southwest Jiaotong University*, vol. 42, pp. 641-646, 2007.
[21] B. Y. Chen, W. H. Lam, A. Sumalee, Q. Li, and M. L. Tam, "Reliable shortest path problems in stochastic time-dependent networks," *Journal of Intelligent Transportation Systems*, vol. 18, pp. 177-189, 2014.
[22] C. Kwon, T. Lee, and P. Berglund, "Robust shortest path problems with two uncertain multiplicative cost coefficients," *Naval Research Logistics (NRL)*, vol. 60, pp. 375-394, 2013.
[23] Y. L. Chen and K. Tang, "Finding the Kth shortest path in a time-schedule network," *Naval Research Logistics (NRL)*, vol. 52, pp. 93-102, 2005.
[24] R. K. Cheung, "Iterative methods for dynamic stochastic shortest path problems," *Naval Research Logistics (NRL)*, vol. 45, pp. 769-789, 1998.
[25] I. Murthy and S. S. Her, "Solving min-max shortest-path problems on a network," *Naval Research Logistics (NRL)*, vol. 39, pp. 669-683, 1992.
[26] S. Tufekci, "Decomposition algorithms for finding the shortest path between a source node and a sink node of a network," *Naval Research Logistics Quarterly*, vol. 30, pp. 387-396, 1983.
[27] R. K. Ahuja, "Network flows," TECHNISCHE HOCHSCHULE DARMSTADT, 1993.
[28] R. Batta and S. S. Chiu, "Optimal obnoxious paths on a network: transportation of hazardous materials," *Operations Research*, vol. 36, pp. 84-92, 1988.
[29] J. R. Current, "Multiobjective design of transportation networks," 1981.
[30] J. Current and M. Marsh, "Multiobjective transportation network design and routing problems: Taxonomy and annotation," *European Journal of Operational Research*, vol. 65, pp. 4-19, 1993.
[31] L. Liu, H. Mu, H. Luo, and X. Li, "A simulated annealing for multi-criteria network path problems," *Computers & Operations Research*, vol. 39, pp. 3119-3135, 2012.
[32] H. Huang and S. Gao, "Optimal paths in dynamic networks with dependent random link travel times," *Transportation Research Part B: Methodological*, vol. 46, pp. 579-598, 2012.
[33] Y. M. Nie and X. Wu, "Shortest path problem considering on-time arrival probability," *Transportation Research Part B: Methodological*, vol. 43, pp. 597-613, 2009.
[34] L. C. Bezerra, E. F. Goldbarg, L. S. Buriol, and M. C. Goldbarg, "Grace: A generational randomized ACO for the multi-objective shortest path problem," in *Evolutionary Multi-Criterion Optimization*, 2011, pp. 535-549.
[35] S. Abbasi and S. Ebrahimnejad, "The cross-entropy method for solving bi-criteria network flow problems in discrete-time dynamic networks," *Optimization Methods and Software*, pp. 1-19, 2014.
[36] L. B. Reinhardt and D. Pisinger, "Multi-objective and multi-constrained non-additive shortest path problems," *Computers & Operations Research*, vol. 38, pp. 605-616, 2011.
[37] E. Nasrabadi and S. M. Hashemi, "Minimum cost time-varying network flow problems," *Optimization Methods & Software*, vol. 25, pp. 429-447, 2010.

[38] D. Maniezzo, M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: An Autocatalytic Optimizing Process," 1991.
[39] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of machine learning*, ed: Springer, 2010, pp. 36-39.
[40] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," Computational Intelligence Magazine, IEEE, vol. 1, pp. 28-39, 2006.
[41] Ekmen, Elçin Duygu. "A Study on Performance Evaluation of Optimization Algorithms in the Shortest Path Problem." PhD diss., Ankara Yıldırım Beyazıt Üniversitesi Fen Bilimleri Enstitüsü, 2020.
[42] Li, Xuan, Xiaofei Ye, and Lili Lu. "Dynamic Programming Approaches for Solving Shortest Path Problem in Transportation: Comparison and Application." In Green, Smart and Connected Transportation Systems, pp. 141-160. Springer, Singapore, 2020.
[43] Kumawat, Sunita, Chanchal Dudeja, and Pawan Kumar. "An Extensive Review of Shortest Path Problem Solving Algorithms." In 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 176-184. IEEE, 2021.
[44] Hughes, Michael S., Brian J. Lunday, Jeffrey D. Weir, and Kenneth M. Hopkinson. "The multiple shortest path problem with path de-confliction." European Journal of Operational Research 292, pp.818-829, no. 3 (2021).