Iranian Journal of Fuzzy Systems

Volume 21, Number 4, (2024), pp. 123-139

Original paper

University of Sistan and Baluchestan

Fuzzy maximum capacity path problem and its application to optimal routing control

J. Tayyebi [1], A. Deaconu [2], E. Hosseinzadeh [3] and A. M. Golmohammadi [4]

[1] *Department of Industrial Engineering, Faculty of Industrial and Computer Engineering, Birjand University of Technology, Birjand, Iran*

[2] *Department of Mathematics and Computer Science, Transilvania University, Brasov, 500091, Romania*

[3] *Department of Mathematics, Kosar University of Bojnord, Bojnord, Iran*

[4] *Department of Industrial Engineering, Arak University, Arak, Iran*

javadtayyebi@birjandut.ac.ir, a.deaconu@unitbv.ro, e.hosseinzadeh@kub.ac.ir, a-golmohammadi@araku.ac.ir

Abstract

The maximum capacity path problem (MCPP) is a classical combinatorial optimization problem that seeks to find a path with the maximum capacity in a network. In this paper, we consider a fuzzy extension of the MCPP, where the capacities are given as arbitrary fuzzy numbers. Unlike previous approaches that rely on ranking functions or specific orderings, we formulate the fuzzy MCPP as a bi-objective path-finding problem, where one objective is to maximize the nominal capacity and the other is to optimize the reliability value of the path. We propose an efficient algorithm that can find a Pareto optimal path for any aggregation function between the two objectives. We also analyze the special case where the network is acyclic and show that the algorithm can be specialized to run in strongly polynomial time. Furthermore, we present an application of the fuzzy MCPP to the field of optimal control, where we use a discretization algorithm to transform a continuous routing problem into a discrete one and solve it using the proposed algorithm as a subroutine. To implement this application in practice, we run the algorithm on an old real-world project in Iran called Iranrud. Moreover, we report some computational results on grid networks with different sizes that illustrate the performance of the proposed algorithm.

Keywords: Routing, maximum capacity path, fuzzy capacity, optimal control, Iranrud project.

1 Introduction

The Maximum Capacity Path Problem (MCPP) is a well-known combinatorial optimization problem. The goal of the MCPP is to find the path in a given network that has the maximum capacity, where the capacity represents the ability of an arc to send a certain amount of flow or resources. The capacity of a path is defined as the minimum of the capacities of the arcs on this path. The MCPP can be seen as a routing problem related on the shortest path problem, where instead of minimizing the total length of a path, it maximizes the bottleneck capacity of a path [43]. The MCPP can be solved by adapting most shortest path algorithms, such as Dijkstra's algorithm, to use the bottleneck distance instead of path length [24, 41]. However, in some cases, faster algorithms are possible that exploit the properties of the problem. For example, in undirected graphs, it can be solved by a recursive algorithm in linear time [42].

The MCPP not only stands on its own but also serves as a crucial sub-problem within broader problems in network optimization. These include the classic maximum flow problem discussed by Ahuja, Magnanti, and Orlin in [1] and Edmonds and Karp in [22], the optimization of quickest paths seen in the works of [11, 34], and the k-splittable flow problem as studied in [6]. The interconnection of these problems highlights the significance of the problem in diverse applications of network theory and optimization.

Corresponding Author: J. Tayyebi

Received: June 2024; Revised: August 2024; Accepted: September 2024.

https://doi.org/10.22111/ijfs.2024.49145.8665

The MCPP also has many applications in various real-world scenarios across different domains. One of the most prominent domains where the MCPP arises is telecommunication networks, where the routing of information packets between network nodes is subject to the network's limited transmission capacity. A transmission path is a sequence of interconnected links, each of which has a specific available bandwidth that indicates how much data can be transmitted per unit of time. When packets need to be routed from a source node to a destination node, transmission mechanisms, called protocols, are used to determine the optimal path for each packet. These mechanisms also aim to maintain a certain level of quality of service, which is often defined by a guaranteed transmission bandwidth that ensures the delivery of packets within a specified time frame. An example of a protocol that uses the MCPP to route packets is Multiprotocol Label Switching (MPLS), which establishes paths for packets between two nodes (the entry and exit routers) through what is known as a label-switched path [35].

Another application is the design of reliable data transmission paths, where the quality of a transmission path is measured by the minimum transmission reliability across its constituent links. The transmission reliability of a link is the probability that the link will successfully transmit a packet without any errors or failures. The transmission reliability of a path is the product of the transmission reliabilities of the links on the path. The higher the transmission reliability of a path, the lower the probability of losing or corrupting packets during transmission. Therefore, the MCPP can be used to find the path with the highest transmission reliability between two nodes in a network [47].

The MCPP also has applications in the efficient routing of specific service units like police and firefighters, where the effectiveness of a route is influenced by factors such as the maximum distance or response time to a potential service location. For example, in a city, there may be multiple police stations and fire stations that can respond to emergency calls from different locations. The MCPP can be used to find the optimal route for each service unit to reach the location of the emergency, where the capacity of a route is the inverse of the distance or the response time [7].

This paper considers a variant of the MCPP where the capacities of the arcs are represented by fuzzy numbers. Fuzzy numbers are mathematical objects that can capture the uncertainty or imprecision of real-world information. The paper extends the MCPP to the fuzzy environment for the following reasons:

- *Uncertainty in capacity:* The actual capacities of the arcs may not be known exactly or deterministically in reality, especially in subways. There may be factors that affect the capacities, such as errors in measurement, noise in transmission, or human judgment. Fuzzy numbers allow us to model the uncertainty in the capacities and express the possibility of different capacity values.

- *Variations in capacity:* In many real-world situations, the capacities of the arcs may vary over time or depend on the environmental conditions. For example, the capacity of a route may change due to the traffic congestion, the weather, or the maintenance schedule. Fuzzy numbers enable us to account for the variations in the capacities and reflect the dynamic nature of the problem [45].

- *Incomplete information:* Sometimes complete information about the capacities may not be available or accessible. For example, in military applications, we may not have complete information about the enemy's area, such as the location, the strength, or the strategy of the enemy. Using fuzzy numbers allows us to deal with the incomplete information and incorporate it into the decision-making processes [37, 46].

1.1 A short literature review

The classical network optimization problems, such as maximum flow, shortest path, and spanning tree, with fuzzy arc parameters were initially introduced and explored in the late 1970s and early 1980s (see references [10], [18], [29]). Subsequently, fuzzy network optimization problems are interested by many researchers. Table 1 briefly outlines some recent research on network optimization in fuzzy environments.

As far as we know, there is only one paper addressing fuzzy maximum capacity path problems [45]. This study focused on time-varying maximum capacity path problems with zero waiting times, where arc capacities are represented by generalized trapezoidal fuzzy numbers. The problem involves finding a path from source node s to sink node t that maximizes the path capacity while adhering to a specified integer time limit T. An algorithm proposed in the paper, based on a recursive relation for determining the maximum capacity of a path from source node s to node i at time t denoted as $\xi(i,t)$, addresses this problem. By computing $\xi(i,t)$, the fuzzy capacity of the maximum time-varying path from s to i is derived as $\xi^*(i) = \max_{0\leq t\leq T} \xi(i,t)$.

1.2 Key contributions and novel insights

This paper introduces a novel approach for solving the fuzzy MCPP. The concept is to redefine the problem as a bi-objective routing problem. One objective aims to maximize the nominal capacity of the path, while the other focuses

Problem	Ref.	Year	Type of fuzzy parameters	Approach
Minimum spanning tree problems	[16]	2024	Interval-Valued Neutrosophic	a Dhouib-Matrix-MSTP method
	[36]	2023	Pentagonal intuitionistic	A novel defuzzification method
	[12]	2021	type-2	A defuzzification using a critical value and centroid method
	[15]	2020	Type-2	A genetic algorithm
	[13]	2019	Trapezoidal	An extension of Kruskal's algorithm
	[14]	2016	Trapezoidal	An extension of Prim's algorithm
	[4]	2005	Triangular	A genetic algorithm
Shortest path problems	[19]	2024	Intuitionistic	A PSO-based approach
	[8]	2023	Triangular neutrosophic	An ant colony algorithm
	[17]	2022	Trapezoidal and normal	An ant colony algorithm
	[21]	2021	Mixed interval-valued	A bee colony algorithm
	[38]	2021	Trapezoidal picture	An extension of Bellman-Ford algorithm
	[2]	2021	Trapezoidal picture	An extension of Dijkstra algorithm
	[30]	2020	Normal fuzzy	A ranking function
Maximum flow problems	[33]	2018	Random fuzzy variables	A genetic algorithm
	[32]	2012	Triangular	A ranking function
	[31]	2010	Triangular	A ranking function
	[25]	2007	Trapezoidal	An extension of Ford-Fulkerson algorithm
Transportation problems	[39]	2024	Interval-valued trapezoidal	Novel formulations
	[3]	2023	Fermatean fuzzy sets	An ordering-based compromise approach
	[5]	2022	Triangular	A fuzzy DEA models
Maximum capacity path problems	[45]	2016	Generalized trapezoidal	A combinatorial algorithm

Table 1: A background of network optimization problems in fuzzy environments

on optimizing the reliability value of the path. An efficient algorithm is put forward in the paper to find a Pareto optimal path for the fuzzy MCPP. This algorithm is capable of accommodating any aggregation function between the two objectives, such as the weighted sum and the weighted product. Additionally, an examination is conducted on the special scenario in which the network is acyclic. It is demonstrated that in such instances, the algorithm can be customized to operate in polynomial time. Moreover, the paper presents an application of the fuzzy MCPP in the field of optimal control. It employs a discretization algorithm to convert the continuous problem into a discrete format by dividing the area into smaller segments. Subsequently, the proposed algorithm for the fuzzy MCPP is used as a subroutine to determine the optimal path. To implement this application in practice, a project, called Iranrud, is examined, with the objective of finding a route for constructing a canal that connects the Caspian Sea to the Persian Gulf in Iran [48]. Finally, computational analyses are included to demonstrate the performance of the proposed algorithms on grid networks.

The subsequent sections of this paper are organized as follows: Section 2 formally presents the problem and casts it as a bi-objective zero-one linear programming model. In Section 3, the proposed algorithm is detailed, and the computational results are reported on different grid networks to evaluate the performance of the algorithm. Section 4 showcases an application involving optimal routing control, specially, it implements the application for the Iranrud project. Finally, Section 5 provides closing remarks to encapsulate the key findings and implications discussed throughout the paper.

2 Problem statement

This section introduces the fuzzy MCPP in a formal manner and presents the mathematical formulation of the problem as a bi-objective zero-one linear programming model.

Let $G(V, A, \mathbf{c})$ be a connected directed network with node set $V = \{1, 2, \ldots, n\}$ and arc set A of size m. Each arc (i, j) has a capacity c_{ij}, which denotes the maximum amount of flow that can pass through it. Let s and t be the origin and destination nodes, respectively. A path P is a sequence of vertices $v_1, v_2, \cdots, v_k$ such that (v_l, v_{l+1}) is an arc for all $l = 1, 2, \cdots, k-1$. A cycle C is a path that starts and ends at the same node, i.e., $v_1 = v_k$. A simple path is a path that does not repeat any node, except possibly the first and last. A simple path from s to t is called an st-path. This paper enforces the following assumption to eliminate additional arcs from the network:

Assumption 1. Every arc is part of at least one st-path.

The capacity of an st-path P is the minimum capacity of its arcs, i.e., $\min_{(i,j)\in P} c_{ij}$. The goal of the MCPP is to

find an st-path with the maximum capacity. This problem can be written as the following combinatorial optimization problem:

$$\max_{P\in\mathcal{P}} \min_{(i,j)\in P} c_{ij}, \tag{1}$$

where $\mathcal{P}$ is the set of all st-paths in G.

In this paper, the MCPP is generalized to the situation where the capacity of an arc (i,j) is given as a fuzzy number $\tilde{c}_{ij}$ with membership function $\mu_{\tilde{c}_{ij}} : R^{\geq 0} \to [0,1]$. The core of the fuzzy number is assumed to be a singleton set $\{c^2_{ij}\}$, and its support is the interval (c^1_{ij}, c^3_{ij}).

A potential solution to this problem is to find the st-path with the largest capacity using the nominal capacity values c^2_{ij}, which are the most likely values in fuzzy numbers. However, this path may not be trustworthy, so it is needed to establish a reliability measure for each path selected in this manner. To do this, let P be any st-path with a capacity of $c(P) = \min_{(i,j)\in P} c^2_{ij}$. Then, for every arc (i,j) on this path, the equality $c^2_{ij} \geq c(P)$ holds. Hence, there are two different cases for the left extension of each edge:

$\mathbf{c^1_{ij} \leq c(P)}$: This implies that the actual capacity of the arc (i,j) may be a number in the interval $[c^1_{ij}, c(P)]$ and thus there is a risk of reducing the capacity of this path. To represent this uncertainty, we take the value

$$risk_{ij}(P) = \frac{\int_{c^1_{ij}}^{c(P)} \mu_{\tilde{c}_{ij}}(x)dx}{\int_{c^1_{ij}}^{c^3_{ij}} \mu_{\tilde{c}_{ij}}(x)dx}$$

as the probability that shows the uncertainty caused by the occurrence of this arc on the path (see Figure 1).

$\mathbf{c^1_{ij} > c(P)}$: This means that $c(P)$ does not belong to the support of $\tilde{c}_{ij}$. So, any likely value in (c^1_{ij}, c^3_{ij}) will not alter the capacity of this path and will not result in another path being chosen. So, we set $risk_{ij}(P) = 0$.

Although the second case does not pose any problem, the first case may reduce the estimated capacity for the path P and consequently this path may not remain as the path with the maximum capacity. Therefore, we consider the following expression as the overall probability that models the reliability of this path:

$$\begin{aligned} r(P) &= \prod_{(i,j)\in P} (1 - risk_{ij}(P)) = \prod_{(i,j)\in P} 1 - \frac{\int_{c^1_{ij}}^{c(P)} \mu_{\tilde{c}_{ij}}(x)dx}{\int_{c^1_{ij}}^{c^3_{ij}} \mu_{\tilde{c}_{ij}}(x)dx} \\ &= \prod_{(i,j)\in P} \frac{\int_{c(P)}^{c^3_{ij}} \mu_{\tilde{c}_{ij}}(x)dx}{\int_{c^1_{ij}}^{c^3_{ij}} \mu_{\tilde{c}_{ij}}(x)dx}. \end{aligned}$$

Hence, the problem becomes a bi-objective problem, where the first objective function is to maximize the minimum nominal capacity of the desired path and the second objective function is to maximize the overall reliability of the path. With these explanations, the problem can be formulated as a zero-one programming problem as follows:

$$\max \quad (z_1 = c(P), z_2 = \prod_{(i,j)\in P} (1 - \frac{\int_{c^1_{ij}}^{c(P)} \mu_{\tilde{c}_{ij}}(x)dx}{\int_{c^1_{ij}}^{c^3_{ij}} \mu_{\tilde{c}_{ij}}(x)dx})) \tag{2a}$$

$$\text{s.t.} \quad c(P) \leq c^2_{ij} + \mathcal{C}(1 - x_{ij}) \quad \forall (i,j) \in A, \tag{2b}$$

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1 & i = s \\ 0 & i \notin s,t \\ -1 & i = t \end{cases} \quad \forall i \in V, \tag{2c}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A. \tag{2d}$$

where $\mathcal{C} > 0$ is a significantly large number, and $c(P) = \min c^2_{ij} + \mathcal{C}(1 - x_{ij})$. Here, we can assume that $\mathcal{C} = \max_{(i,j)\in A} c^2_{ij}$.

3 Algorithm design

The provided formulation for the problem is a bi-objective optimization model in which the first objective function appears in the second objective function, as the bound of the integral. So, if one wants to solve the problem by solvers, then he/she requires integer nonlinear optimization solvers which can compute the integral of a function. However,

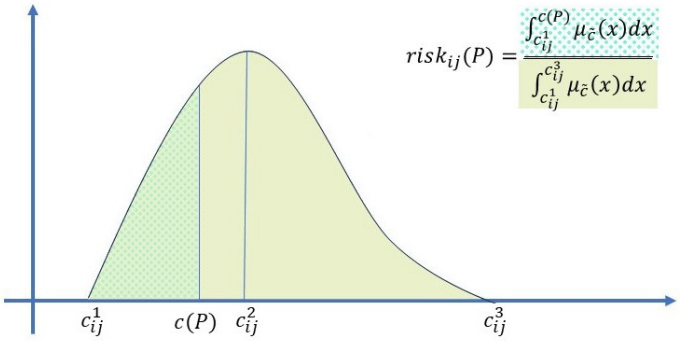

Figure 1: The definition of risk for an arc (i, i) along an st-path P

this section presents an algorithm which computes the value of integrals implicitly, and it only require integer linear optimization solvers whose algorithms are efficient in practice. Moreover, it proposes a specification of the proposed algorithm for acyclic graphs which can solve the problem in polynomial time.

Another challenge in solving bi-objective problems lies in how to combine the objective functions into a single function. Various methods, such as weighting method, lexicographic method, etc., exist for this purpose [28]. In this paper, a general aggregation function $f(z_1, z_2)$ for objective functions is considered, encompassing different categories of multi-objective methods. The proposed algorithm effectively solve the problem under this general aggregation function. For instance, the function $f(z_1, z_2)$ can take one of the following forms based on the application type or any other form:

- Weighted function $f(z_1, z_2) = wz_1 + (1 - w)z_2$: This method can be used to calculate strong Pareto points, but it requires the decision maker to estimate the weight w.
- Lexicographic function $f(z_1, z_2) = Mz_1 + z_2$: Here, M is a significantly large constant. This function represents the lexicographic method, indicating that the first objective function is much more important than the second one.
- Nonlinear function $f(z_1, z_2) = z_1 - M \times \max\{0, z_2^0 - z_2\}$: In this function, M is a large constant, and z_2^0 is an estimate of the decision maker's satisfaction level for the second objective function. This function models the prominent ϵ-constraint method.

In the proposed algorithm, we first sort the individual nominal capacity values of arcs in ascending order and assume $u_1 < u_2 < \ldots < u_l$ is the desired list. The algorithm iterates through each value u_k in this list. It focuses on solving a model where the value of the first objective function is set to u_k and optimizes only the second objective function. This model is described as follows:

$$\min \quad v = \sum_{(i,j)\in A} -log(1 - \frac{\int_{c_{ij}^1}^{u_k} \mu_{\tilde{c}_{ij}}(x)dx}{\int_{c_{ij}^1}^{c_{ij}^3} \mu_{\tilde{c}_{ij}}(x)dx})x_{ij}, \tag{3a}$$

$$\text{s.t.} \quad u_k \le c_{ij}^2 x_{ij} + \mathcal{C}(1 - x_{ij}) \quad \forall (i,j) \in A, \tag{3b}$$

$$c_{ij}^2 x_{ij} \le u_k y_{ij} + \mathcal{C}(1 - y_{ij}) \quad \forall (i,j) \in A, \tag{3c}$$

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1 & i = s \\ 0 & i \notin s,t \\ -1 & i = t \end{cases} \quad i \in V, \tag{3d}$$

$$\sum_{(i,j)\in A} y_{ij} \ge 1, \tag{3e}$$

$$x_{ij}, y_{ij} \in \{0,1\} \quad \forall (i,j) \in A. \tag{3f}$$

The difference between this model and the previous one is that the first objective function has a predetermined value u_k and the zero-one variables y_{ij} together with constraints (3c) and (3f) ensure that the nominal capacity of at least one arc along the path is equal to u_k. Therefore, the path found has a nominal capacity exactly equal to u_k. By taking the logarithm and changing the sign, the second objective function of problem (2) is converted into a minimization objective function that does not depend on the first objective value $c(P)$. So $z_2 = -e^v$ obviously. Now, the independence of the integral bounds from the decision variables allows the coefficients of the objective function, $-log(1 - \frac{\int^{c(P)} c^1 ij \mu_{\tilde{c}_{ij}}(x)dx}{\int^{c_{ij}^3} c^1 ij \mu_{\tilde{c}_{ij}}(x)dx})$,

to be calculated in a preprocessing step. Subsequently, model (3) can be optimized using linear optimization software such as Lingo [44] and GAMS [9].

If a programming language that enables manipulation of the underlying graph is preferred, the formulation of model (3) can be simplified further. To achieve this, let's consider that the algorithm generates a subgraph $G_k(V, A_k, \tilde{\mathbf{c}})$ with the same set of nodes and fuzzy capacity vector as the original network G, but containing only arcs with nominal capacities greater than or equal to u_k. Consequently, every path in this subgraph will have a minimum nominal capacity of u_k. As a result, a lower bound of u_k can be set for the first objective function. By introducing this auxiliary network, model (3) can be reformulated as follows:

$$\min \quad v = \sum_{(i,j)\in A_k} -log(1 - \frac{\int_{c_{ij}^1}^{u_k} \mu_{\tilde{c}_{ij}}(x)dx}{\int_{c_{ij}^1}^{c_{ij}^3} \mu_{\tilde{c}_{ij}}(x)dx})x_{ij}, \tag{4a}$$

$$\sum_{j:(i,j)\in A_k} x_{ij} - \sum_{j:(j,i)\in A_k} x_{ji} = \begin{cases} 1 & i = s \\ 0 & i \notin s, t \\ -1 & i = t \end{cases} \quad \forall i \in V, \tag{4b}$$

$$\sum_{(i,j)\in \bar{A}_k} x_{ij} \geq 1, \tag{4c}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A_k. \tag{4d}$$

in which $\bar{A}_k$ is the set of arcs having the nominal capacity exactly equal to u_k. Here, the decision variables y_{ij} have been removed, and constraint (4c) ensures that the path's nominal capacity is u_k. In the realm of combinatorial optimization, model (4) represents a constrained shortest path problem that can also be solved by dynamic programming techniques [20].

Our proposed algorithm finds the optimal solution by solving problem (4) on the network $G_k(V, A_k, c)$ repeatedly and comparing the results. Algorithm 1 shows how this works. Steps 3 and 4 have been included to comply with Assumption 1, while Steps 15-17 decrease the iteration number of the For loop by taking into account that model (4) is infeasible for certain arcs.

Our proposed algorithm iteratively solves a sequence of problems (4) to determine the optimal solution for the original problem. Specifically, the algorithm constructs a series of subnetworks $G_k(V, A_k, c)$ by progressively removing arcs based on their capacity characteristics. For each subnetwork, problem (4) is solved, and the corresponding objective value is evaluated using the aggregation function $f(z_1, z_2)$. The optimal solution is identified as the one yielding the maximum aggregated value among all iterations.

Algorithm 1 provides a detailed implementation of this approach. Steps 3 and 4 are essential for ensuring that the problem structure satisfies Assumption 1, a crucial prerequisite for the subsequent optimization steps. The efficiency of the algorithm is enhanced by Steps 15-17, which intelligently prune the search space by eliminating arcs that are guaranteed to not contribute to feasible solutions. This pruning strategy effectively reduces the computational burden by avoiding unnecessary iterations.

Table 2: Notations and Variables used in Algorithm 1

Symbol	Description
s	Source node in the network
t	Target node in the network
V	Set of nodes in the network
A	Set of arcs in the network
$c_{ij}^1, c_{ij}^2, c_{ij}^3$	Parameters associated with fuzzy capacity $\tilde{c}_{ij}$ of arc (i, j)
$\mu_{\tilde{c}_{ij}}(x)$	Membership function
u_k	k-th distinct value of c_{ij}^2
$\bar{A}_k$	Set of arcs with $c_{ij}^2 = u_k$
A_k	Set of remaining arcs in iteration k
G_k	Subgraph induced by V and A_k
v	Optimal value of problem (4)
z	Optimal value of the overall problem
$f(z_1, z_2)$	Aggregation function

Algorithm 1 To solve bi-objective optimization problem (2)

Input: A instance of problem (2) as well as a aggregation function $f(z_1, z_2)$.
Output: The optimal value for problem (2) with the single objective function $f(z_1, z_2)$.
Applying a traversal algorithm such as BFS, label all arcs accessible from s and label all arcs from which t is accessible.
Remove arcs that are not labeled twice. Let A be the arc set.
Sort the distinct elements of $\{c^2_{ij} : (i,j) \in A\}$ in ascending order. Let $u_1, u_2, \ldots, u_l$ be the sorted list.
Set $z^* = -\infty$ and $A_k = A$.
for $k = 1, 2, \ldots, l$ **do**
    Obtain the set $\bar{A}_k$ containing arcs (i,j) with $c^2_{ij} = u_k$.
    Compute the objective coefficients $-log(1 - \frac{\int_{c^1_{ij}}^{u_k} \mu_{\tilde{c}_{ij}}(x)dx}{\int_{c^1_{ij}}^{c^3_{ij}} \mu_{\tilde{c}_{ij}}(x)dx})$ for every $(i,j) \in A_k$.
    Solve problem (4) on $G_k = (V, A_k, \tilde{\mathbf{c}})$. Let its optimal value be v^*.
    if $f(u_k, e^{-v^*}) > z^*$ **then**
        Update $z^* = f(u_k, e^{-v^*})$.
    end if
    Remove arcs of $\bar{A}_k$ from A_k.
    if There is no st-path in the network **then**
        Break.
    end if
end for

Theorem 3.1. *Algorithm 1 solves problem* (2) *in* $O(mS(n,m)))$ *time and linear space where* $S(n,m)$ *is the time required for solving problem* (4)*.*

Proof. The proof contains three parts:

Part I (Correctness of the Algorithm): The algorithm starts by identifying a finite set of capacity values from the arcs in the network, denoted as $U = \{u_1, u_2, \ldots, u_l\}$. For each unique capacity u_k, the algorithm proceeds to find an st-path such that the second objective function is minimized. Given that the algorithm evaluates all possible values in (U) for the first objective function and minimizes the second objective function accordingly, it guarantees finding the optimal solution to the problem described in (2).

Part II (Time Complexity Analysis): The outer loop of the algorithm iterates over all unique capacity values in the set U. The number of unique capacity values is at most equal to the number of arcs, m. Hence,

$$l = |U| \leq m \quad \Longrightarrow \quad O(l) = O(m).$$

For each iteration k, the algorithm calls a routine to solve the subproblem defined by model (4), which takes $S(n,m)$ time. This means that for each iteration of the loop, we must account for the time required to solve this subproblem. Given the outer loop runs $l = O(m)$ times and each call to solve the subproblem takes $O(S(n,m))$, the total time complexity is derived as follows:

$$\text{Total Time} = O(l \cdot S(n,m)) = O(m \cdot S(n,m)).$$

Part II (Space Complexity Analysis): The algorithm only requires space to hold the graph's arc representation and additional data structures for managing labels and capacities. The same space is used to solve the problem (2) by either SSP or solver. Since the main data structures managing arcs are proportional to the number of arcs (m), the space complexity due to these elements is $O(m)$. Consequently, the overall space complexity of the algorithm is $O(m)$. □

3.1 Acyclic case

The paper now concentrates on problem (2) in the case where the graph is acyclic. It asserts that Algorithm 1 is capable of solving this problem in polynomial time under this assumption.

It's worth noting that the bottleneck operation of Algorithm 1 involves solving problem (4), which relies on the performance of the optimization solver being utilized. Nevertheless, problem (4) can be solved in linear time when the graph is acyclic. Let us discuss this issue in more details.

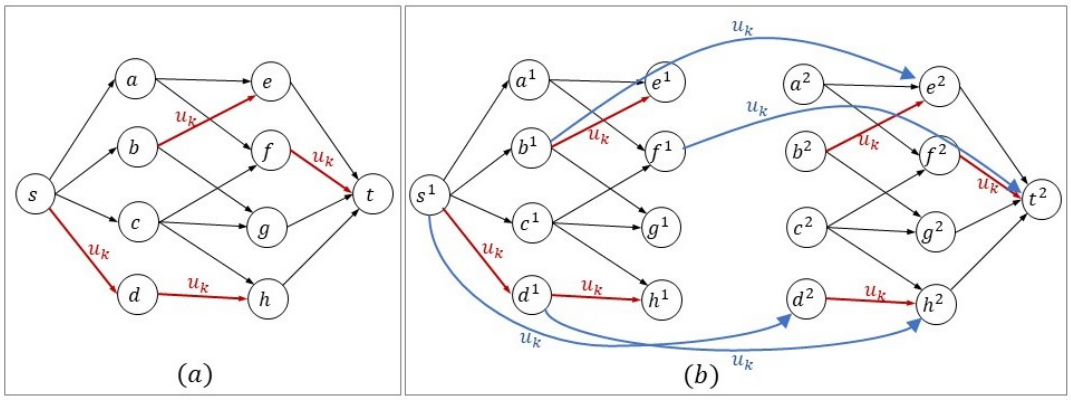

Figure 2: Illustration of Graph Transformation

By neglecting constraint (4c), the problem defined in (4) becomes a shortest path problem on the graph $G(V, A_k)$ with arc lengths given by $-log(1 - \frac{\int_{c_{ij}^1}^{u_k} \mu_{\tilde{c}_{ij}}(x)dx}{\int_{c_{ij}^1}^{c_{ij}^3} \mu_{\tilde{c}_{ij}}(x)dx}) > 0$. Constraint (4c) ensures that any viable path must include at least one arc with capacity u_k. To model this, we modify the underlying graph in the following manner (see Figure 2 for an example):

1. Create two duplicates of the graph. Label the nodes of the first (second) duplicate with a superscript 1 (2).
2. Exclude node t^1 from the first duplicate and node s^2 from the second.
3. For any arc $(i, j) \in \bar{A}_k$, add an arc (i^1, j^2) to the new graph, serving as a bridge between the two duplicates, maintaining the same capacity.

This approach effectively models the requirement for a path to traverse an arc with capacity u_k by altering the graph structure accordingly. Since the modified graph consists of precisely $2n - 2$ nodes and no more than $3m$ arcs while remaining acyclic, it is deduced that a shortest path can be computed in $O(m)$ time. Consequently, we can immediately deduce the following theorem:

Theorem 3.2. *Algorithm 1 efficiently solves problem* (2) *on acyclic graphs within* $O(m^2)$ *time.*

Remark 1. Applying the same methodology to general graphs may result in a path from s to t that includes some cycles, rather than a (simple) st-path.

3.2 Computational results

This subsection presents computational results for evaluating the performance of Algorithm 1. The algorithm was implemented on a Windows 11 64-bit laptop powered by an Intel Core $i7 - 1165G7$ CPU (2.80 GHz) with 8 GB of RAM. The programming language Python 3.11, along with the libraries Gurobipy, Matplotlib, and NetworkX, was utilized for algorithm development. Networkx and Matplotlib are utilized for network manipulation and visualization, while Gurobipy serves as an optimization solver. The results presented are averages across ten networks of the same size and sparsity.

To perform first computational experiments, we utilized a special class of undirected graphs called square grid graphs. These graphs have nodes corresponding to points in the plane with integer coordinates. The x-coordinates range from 1 to g_1, while the y-coordinates range from 1 to g_2. Two nodes are connected by an edge if the corresponding points are at a unit distance.

Any edge of the graph is oriented from the endpoint with the smaller coordinate to its other endpoint. Clearly, the directed graph generated in this way is acyclic. The node with coordinates $(1, 1)$ is considered as the origin, and the node with coordinates (g_1, g_2) is considered as the destination. Each arc $(i, j) \in A$ is assigned a triangular fuzzy capacity generated randomly whose components are composed of three random integer numbers generated within the interval $[1, g_1^2 \times g_2^2]$.

Table 3: Computational results of Algorithm 1 for square grid graphs and $f(z_1, z_2) = z_1^{z_2}$

g_1	g_2	n	m	Length of List	Num. of iter.	Solver		SPP		Ratio of Times
						Opt. value $f(z_1^*, z_2^*)$	Time (Sec.)	Opt. value $f(z_1^*, z_2^*)$	Time (Sec.)	
10	10	100	180	177.8	57.7	355.431	0.689	355.431	0.144	4.786
20	10	200	370	368.3	105.8	564.207	3.787	564.207	0.476	7.951
20	20	400	760	757.6	219.7	1673.522	25.214	1673.522	1.866	13.512
30	10	300	560	558.2	135.4	454.967	10.284	454.967	0.909	11.315
30	20	600	1150	1147.7	335	2605.569	62.292	2605.569	3.85	16.181
30	30	900	1740	1736.9	576.9	7095.149	243.899	7095.149	11.045	22.082
40	10	400	750	748	141.3	100.796	13.116	100.796	1.093	11.999
40	20	800	1540	1537.7	419.9	2715.498	143.992	2715.498	6.829	21.087
40	30	1200	2330	2327.7	697.5	–	–	6434.611	21.12	–
40	40	1600	3120	3117.5	930.6	–	–	9360.483	51.029	–
50	10	500	940	937.8	153	54.454	24.609	54.454	1.753	14.035
50	20	1000	1930	1928.3	544.3	1137.937	297.081	1137.937	11.053	26.878
50	30	1500	2920	2918.7	730.9	–	–	4342.347	23.342	–
50	40	2000	3910	3907.9	1136	–	–	10007.785	48.903	–
50	50	2500	4900	4897.9	1361.4	–	–	12468.371	77.435	–

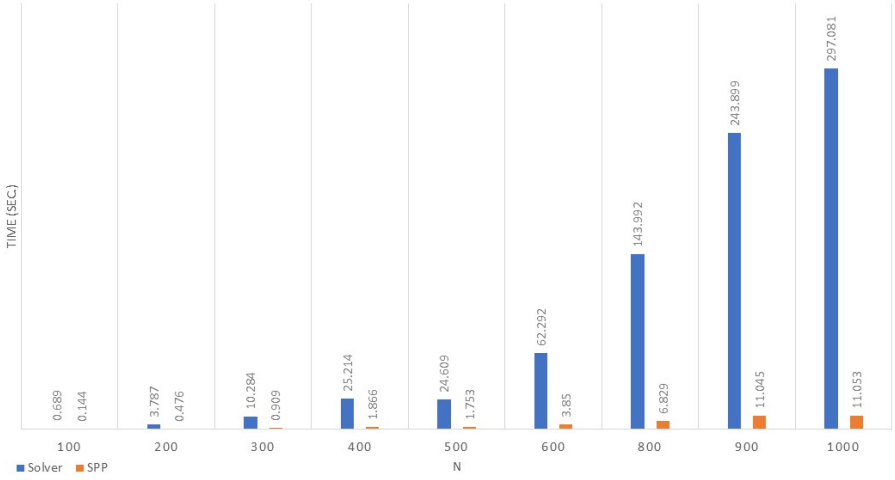

Figure 3: Comparative analysis of Algorithm 1: Solver vs. SPP for grid networks

Table 3 and Figure 3 report computational results for Algorithm 1 in two ways: solving problem 4 by the Gurobipy solver (II) solving it by converting the problem to a shortest path problem (SPP). Consistent objective function values reflect the precision of both methodologies. Given the use of Gurobi's limited edition for computations, which is limited to problems with a maximum of 2000 variables and constraints, certain outcomes could not be determined for the solver. Nonetheless, the comparative analysis reveals a marked advantage of the SPP in relation to the Solver.

As second computational experiments, we employ undirected binomial random graphs as introduced by Erdös and Rúnyi [23] to generate random instances of varying arc densities. These graphs are characterized by two parameters: the number of nodes n, and the edge probability $p \in [0, 1]$. Increasing p leads to denser graphs, with $p = 1$ representing a complete graph. In all cases, Node 1 is designated as the origin ($s = 1$) and Node n as the destination ($t = n$). To create directed edges, we assigned an orientation to each undirected edge (i, j) such that it points from i to j if $i < j$. This process results in acyclic networks which is suitable for testing our algorithm on both solver and shortest path problem (SPP) instances. The key components of triangular fuzzy capacities are integer and generated randomly within the interval $[1, n^2]$.

Table 4 presents the results for values of $n = 40, 60, 80$ and $p = 0.2, 0.4, 0.6, 0.8, 1$. Due to Gurobi's limited edition, we were unable to obtain the final results for the solver. Figure 4 displays a comparison of the running time between the Solver, and the SPP. It is evident that the SPP performs significantly faster than the solver. Notably, the ratio of their running times is approximately 4, which is lower than the computational results observed for grid networks in previous analysis.

Table 4: Computational results on random binomial networks

n	p	m	Length of List	Num. of iter.	Solver		SPP		Ratio of Times
					Opt. value $f(z_1^*, z_2^*)$	Time (Sec.)	Opt. value $f(z_1^*, z_2^*)$	Time (Sec.)	
40.00	0.20	154.00	144.50	94.30	223.41	0.29	223.41	0.07	3.88
	0.40	318.70	285.80	228.40	353.96	1.19	353.96	0.36	3.27
	0.60	467.40	397.50	347.70	458.17	2.72	458.17	0.82	3.31
	0.80	628.10	504.30	451.90	450.62	6.45	450.62	1.99	3.24
	1.00	780.00	598.60	531.00	545.20	8.32	545.20	2.63	3.16
60.00	0.20	351.20	331.10	241.10	585.81	2.49	585.81	0.66	3.77
	0.40	712.20	637.30	567.30	989.05	8.48	989.05	2.19	3.87
	0.60	1061.20	895.30	800.40	1141.05	16.78	1141.05	4.58	3.66
	0.80	1418.20	1136.90	1045.30	1327.31	39.30	1327.31	11.36	3.46
	1.00	1770.00	1345.20	1247.70	1552.46	57.90	1552.46	15.98	3.62
80.00	0.20	618.70	582.50	459.30	1125.33	9.52	1125.33	2.17	4.38
	0.40	1262.80	1121.30	968.40	1799.52	34.25	1799.52	7.97	4.30
	0.60	1907.70	1606.00	1462.30	2359.35	57.63	2359.35	14.55	3.96
	0.80	2533.00	2024.80	1898.50	–	–	2678.29	35.15	–
	1.00	3160.00	2386.50	2273.80	–	–	2678.87	39.07	–

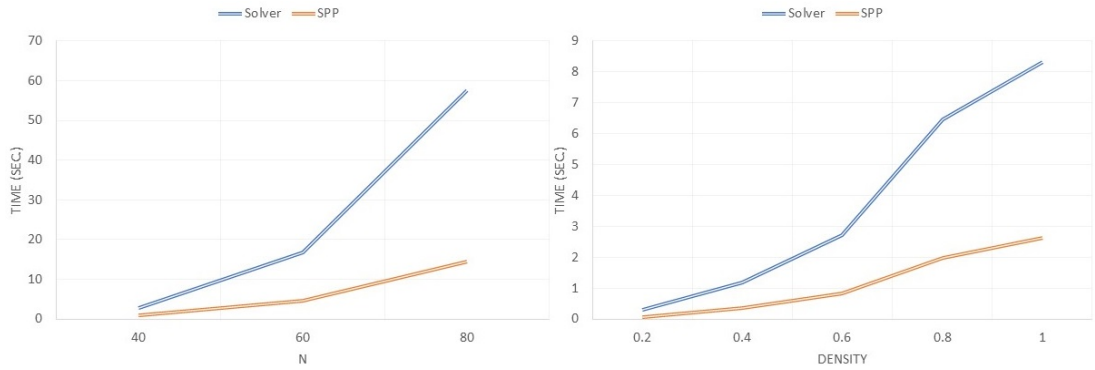

Figure 4: Comparative analysis of Algorithm 1: Solver vs. SPP for random binomial networks: (left) different node numbers and; (right) different densities

4 Application in optimal control

This section explores an application of optimal control in identifying a path with the maximum capacity on a continuous region. Here, "capacity" may be interpreted as various other notions like the degree of safety or the value of reliability, which are encapsulated within a mathematical framework that seeks to optimize a bottleneck objective function.

A discrete-time routing control system is defined as the following equation, [26, 27]:

$$x_{k+1} = f(x_k, v_k), \quad k = 0, 1, 2, \ldots \tag{5}$$

In this equation, x_k denotes the position, and v_k denotes the velocity at the k-th time step. The function $f : X \times V \to \mathbb{R}^d$ is continuous and dictates the subsequent position. The set $X \subseteq \mathbb{R}^d$ represents a bounded region, and $V \subseteq \mathbb{R}^m$ is the set of permissible and bounded velocities. The objective is to determine a sequence of velocities $\mathbf{v} = (v_0, v_1, \ldots) \in V$, starting from an origin point $s \in X$, such that the resulting sequence $(x_k)_{k\in\mathbb{N}}$, defined by:

$$x_0 = s \quad \text{and} \quad x_{k+1}(x, \mathbf{v}) = f(x_k(x, \mathbf{v}), v_k),$$

remains inside the region X. Moreover, the aim is for the sequence $(x_k)_{k\in\mathbb{N}}$ to converge to the destination point t as k tends to infinity. If there exists a velocity vector $\mathbf{v}$ that accomplishes this, the point x is deemed accessible.

A capacity value $c(x)$ is associated with each position x. The system's routing is designed not only to navigate but also to maximize the minimum capacity level. Therefore, we define the capacity $C(\mathbf{v})$ of a route $(x_k)_{k\in\mathbb{N}}$, derived from equation (5), as:

$$C(\mathbf{v}) = \inf_{k=0,1,\ldots,\infty} c(x_k(x, \mathbf{v}), v_k).$$

Thus, a "maximum capacity path" is a trajectory that not only meets the routing requirements but also maximizes the capacity. Our goal of this application is to estimate the value of

$$\bar{C} = \sup_{\mathbf{v}=(v_0,v_1,\ldots)\in V} C(\mathbf{v}).$$

Figure 5: Illustration of constructing a finite graph from the region X: (a) the region X, (b) the partitions of X, (c) the graph $\tilde{G}$

For this purpose, one can visualize the dynamics described by equation (5) using a directed graph representation. Let us define the graph $G = (V, A)$, where V represents the set of nodes, as points of the region, and A represents the set of arcs defined as

$$\{(x, f(x, \mathbf{v})) : x \in X, f(x, \mathbf{v}) \in X, \mathbf{v} \in V\}, \tag{6}$$

capturing the possible transitions from one point to another. To incorporate the capacity considerations, we assign a capacity to any arc which is the minimum of its endpoints' capacity. So, each arc $e = (x, f(x, \mathbf{v}))$ has a capacity

$$c_e = c_{xf(x,\mathbf{v})} = \min\{c(x), c(f(x, \mathbf{v}))\}. \tag{7}$$

In this graph representation, we seek to evaluate the value $\bar{C}$, which quantifies the maximum capacity achievable in the system. In the other word, $\bar{C}$ is obtained by considering the maximum capacity value among all possible paths connecting s to t within the graph G.

In order to estimate the value $\bar{C}$, it is necessary to approximate the infinite graph G by a finite graph (see Figure 5 for an example). To accomplish this, assume R is the smallest rectangle such that $X \subseteq R$. To partition X into manageable subsets, the rectangle R is initially divided into four equal rectangles in such a manner that two different rectangles comprise the origin and destination. Subsequently, a fuzzy capacity is calculated for each rectangle using an approach based on the Monte Carlo method. The calculation involves determining the capacity at several randomly selected points within the rectangle, with the number of points being dependent on the area of the rectangle. The entire rectangle is then considered to possess a triangular fuzzy capacity $\tilde{c} = (c_1, c_2, c_3)$, where c_i are the ith quartile of M, and M denotes the set of randomly selected points at the rectangle. If the length of $supp(\tilde{c})$ exceeds a certain threshold $\epsilon > 0$, the rectangle is subdivided into four other rectangles, and the aforementioned process is repeated.

In order to construct a graph from the obtained rectangles, a node is considered at the center of each rectangle (except the rectangles that include the origin s and the destination t), and an edge is considered between every two vertices whose rectangles share a common edge. The capacity of the rectangle is allocated to each node, and the capacity of each arc is determined based on (7). Through this process, an undirected graph is obtained. To direct the graph, each edge is replaced by two opposite arcs with equivalent capacities. This approximation captures the essential features of G while allowing for computationally feasible analysis.

The objective of this finite approximation is to achieve equilibrium between accurately representing the essential dynamics of the system and minimizing computational complexity. This equilibrium is attained by establishing the value of ϵ.

4.1 Iranrud project

In this section, we describe a practical example that demonstrates a real-world instance of the application. One of the long-standing proposals in Iran is known as the Iranrud project, which aims to construct a canal connecting the Caspian Sea to the Persian Gulf. The concept of connecting the two shores through Iranian territory was first proposed in the 19th century, and the first professional study was conducted in the 1960s [48]. Currently, there are various discussions ongoing about its construction [40].

This section describes the routing problem of this channel from the south of Iran to the capital of Iran, Tehran, which is considered a main phase of this work. For this purpose, the origin point is considered to be near the city of Tehran, and the destination point near of the city Bandar Lengeh located in the south of Iran. Since the elevation

above sea level is very important in the process of digging the ground to create a channel, finding a route that passes through areas with the lowest elevation will significantly reduce the excavation costs. It is notable that if there are other constraints, such as passing through other cities, the problem can be divided into more phases. Therefore, in this problem, the goal is to find a route that passes through areas with the lowest elevation relative to sea level to facilitate the creation of the channel more easily. Actually, this problem is a min-max path problem which can be simply converted into a maximum capacity path problem by negating the coefficients of the objective function.

Here, we use only a map of Iran as the dataset, in which higher points are colored with warmer colors compared to lower points. In Figure 6.a, you can see this map along with the two origin and destination points. We use only this map to create our desired graph for the fuzzy maximum capacity path problem. For this purpose, we first assign an approximate elevation to each color. To do this, we convert the RGB code of each point to the HSV code belonging to $[0, 360]$ and use its Hue code as a representative of the color. Then, according to the map's legend, we scale the numbers on an interval $[0, 2200]$. The formulation of this conversion is as

$$Hght = \begin{cases} \frac{2200(270-h)}{270} & h \leq 270 \\ 0 & 270 < h \leq 360 \end{cases}$$

in which $h = RGBtoHue(R, G, B))$ is the Hue code of the color.

We then apply the gridding approach presented in the previous section with two stopping conditions for re-gridding:

1. The elevation difference at points should not exceed 300.
2. The length and width of the square should not exceed 50 pixels.

The first condition is the same as the one mentioned in the algorithm of the previous section, and the second condition is to reduce unnecessary calculations because when the rectangles are very small, and consequently, they can be considered as a single point. Figure 6.b depicts the map after gridding. It is clear from the map that desert areas have a larger grid, and foothill areas have a smaller grid.

Now, the desired graph is constructed similar to the descriptions in the previous section, where each edge has a fuzzy capacity that shows three components of the fuzzy number: minimum, core, and maximum, respectively. To convert this graph into a directed graph, for each edge (i, j), we obtain four distances $d_{si}, d_{sj}, d_{it}, d_{jt}$, where d_{lk} represents the Euclidean distance between vertex l and k. Then, if $\frac{d_{si}}{d_{si}+d_{it}} \leq \frac{d_{sj}}{d_{sj}+d_{jt}}$, we orient the edge from i to j, and otherwise, we orient it in the opposite direction. Figure 6.c shows the graph which has 1495 nodes and 2538 arcs. This gridding method has two advantages. First, the directed graph is acyclic, which can be easily proven by the triangle inequality and is clearly visible in Figure 6.c. Second, the edges are oriented in such a way that paths with the shortest distance between the origin and destination are created.

The problem was solved using the aggregation function $f(z_1, z_2) = z_1 \times z_2$ and utilizing the version of Algorithm 1 for acyclic graphs, yielding the optimal solution. This solution is depicted in Figure 7. As evident from the optimal solution on the map, the selected path lies in low-elevation areas. The approximate fuzzy capacity of certain arcs located along the optimal path is illustrated below:

$$\{(1064, 1246, 1385), (1020, 1102, 1278), (1109, 1255, 1371), \ldots, (1525, 1649, 1661), \ldots, (473, 542, 604), (213, 388, 695)\}.$$

The optimal values for the two objective functions are as follows:

$$z_1^* = 1649, \quad z_2^* = 0.67840.$$

Moreover, Table 5 provides the optimal values for different aggregation functions.

Methodology	$f(z_1, z_2)$	Elevation(z_1^*)	Reliability (z_2^*)
Weighted product method	$z_1 \times z_2$	1649	0.67840
Weighted sum method	$z_1 + z_2$	1504	0.3852
Lexicographic method	$Mz_1 + z_2$	1504	0.3852
ϵ-constraint method	$z_1 - M \max\{0, 0.9 - z_2\}$	2198	0.91123

Table 5: Optimal objective values for different methodology in multi-objective optimization

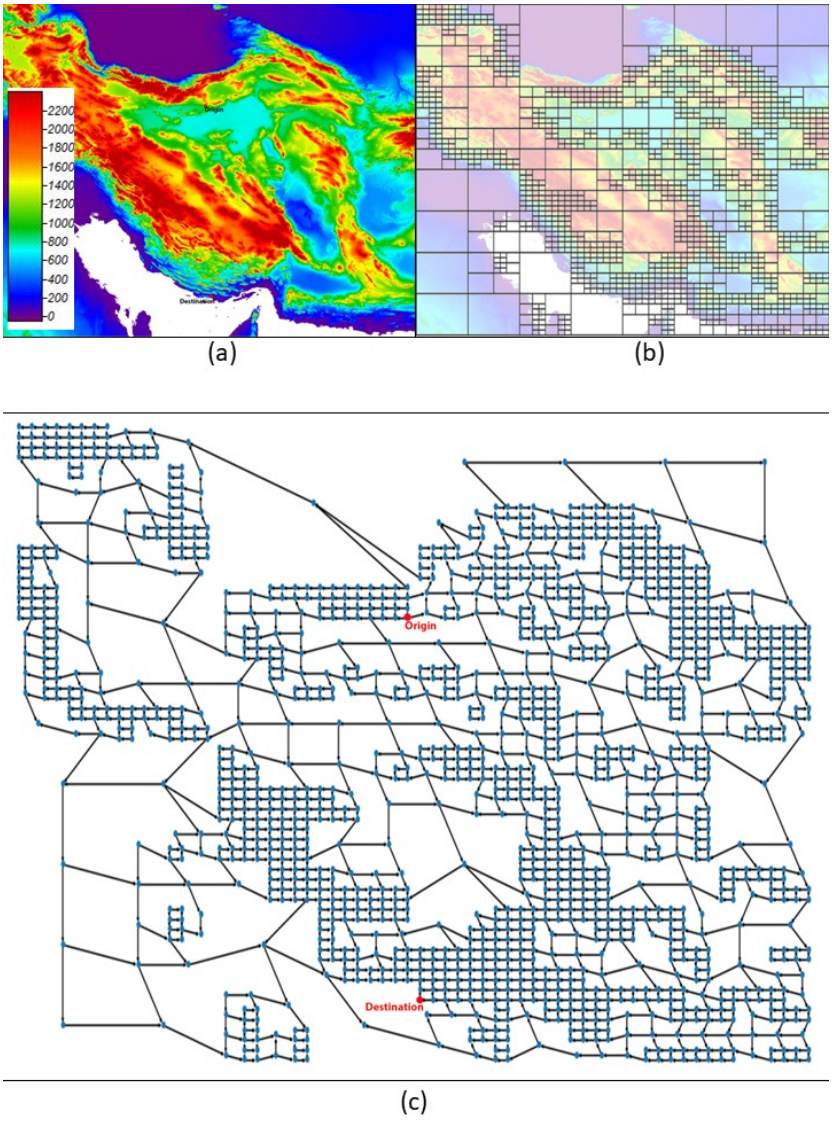

Figure 6: a) Elevation-based color-coded map of Iran; b) Grid-lined map of Iran; c) Graph derived from gridlines on map of Iran

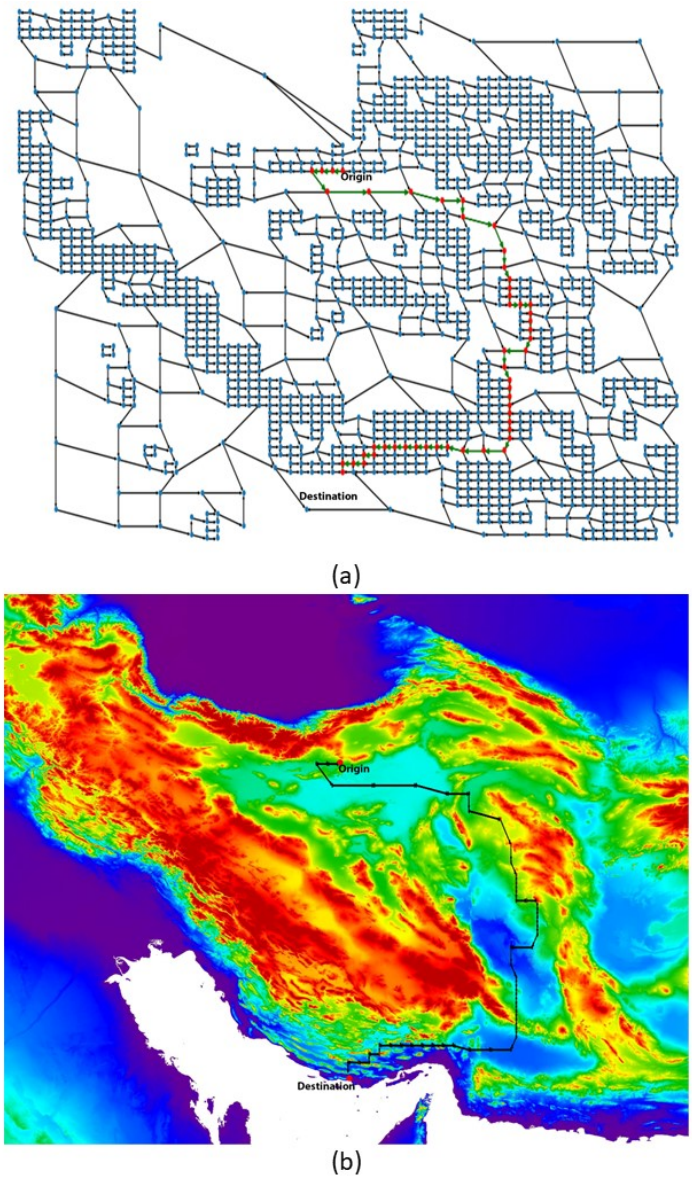

Figure 7: a) Optimal path on the graph; b) Optimal path on the map

5 Concluding remarks

This paper introduced a novel approach to the maximum capacity path problem in a network where arc capacities are represented by fuzzy numbers. By formulating this as a bi-objective optimization problem, balancing nominal capacity and path reliability, an efficient algorithm is proposed to apply shortest path techniques without resorting to ranking functions. The approach is modified for the special case that the network is acyclic. This method offers several advantages, including the ability to accommodate a wide range of fuzzy number types and aggregation operators.

Our primary contributions include (1) the development of a new bi-objective formulation for the maximum capacity path problem in fuzzy networks, and (2) the design of an efficient algorithm that effectively solves this formulation, (3) a modification of the algorithm for acyclic networks. Computational experiments have demonstrated the algorithm's performance in terms of both solution quality and computational time. To demonstrate the application of the algorithm, we concentrated on an ambitious project known as Iranrud. We used the algorithm to identify an appropriate route for building a canal that connects southern and northern Iran.

Future research can explore several promising directions. First, we plan to expand our approach to more complex network structures, including those with changing arc capacities or multiple goods. Second, we intend to examine the

use of our algorithm in real-world situations, particularly in transportation and supply chain management. Additionally, looking into other multi-objective optimization methods, such as evolutionary algorithms, could provide valuable insights into the problem. It would also be worthwhile to consider other bottleneck optimization problems that can be addressed similarly.

References

[1] R. Ahuja, T. Magnanti, J. Orlin, *Network flows: Theory, algorithms, and applications*, Prentice-Hall, 1995. http://hdl.handle.net/1721.1/49424

[2] M. Akram, A. Habib, J. C. Alcantud, *An optimization study based on Dijkstra algorithm for a network with trapezoidal picture fuzzy numbers*, Neural Computing and Applications, **33**(4) (2021), 1329-1342. https://doi.org/10.1007/s00521-020-05034-y

[3] M. Akram, S. Shahzadi, S. M. U. Shah, T. Allahviranloo, *An extended multi-objective transportation model based on Fermatean fuzzy sets*, Soft Computing, (2023), 1-23. https://doi.org/10.1007/s00500-023-08117-9

[4] T. A. Almeida, V. N. Souza, F. M. S. Prado, A. Yamakami, M. T. Takahashi, *A genetic algorithm to solve minimum spanning tree problem with fuzzy parameters using possibility measure*, In NAFIPS 2005-2005 Annual Meeting of the North American Fuzzy Information Processing Society, IEEE. (2005, June), 627-632. https://doi.org/10.1109/NAFIPS.2005.1548610

[5] M. Bagheri, A. Ebrahimnejad, S. Razavyan, F. Hosseinzadeh Lotfi, N. Malekmohammadi, *Fuzzy arithmetic DEA approach for fuzzy multi-objective transportation problem*, Operational Research, (2022), 1-31. https://doi.org/10.1007/s12351-020-00592-4

[6] G. Baier, E. Köhler, M. Skutella, *The k-splittable flow problem*, Algorithmica, **42**(3-4) (2005), 231-248. https://doi.org/10.1007/s00453-005-1167-9

[7] O. Berman, G. Y. Handler, *Optimal minimax path of a single service unit on a network to nonservice destinations*, Transportation Science, **21**(2) (1987), 115-122. https://doi.org/10.1287/trsc.21.2.115

[8] S. Broumi, P. K. Raut, S. P. Behera, *Solving shortest path problems using an ant colony algorithm with triangular neutrosophic arc weights*, International Journal of Neutrosophic Science, **20**(4) (2023), 128-28. https://doi.org/10.54216/IJNS.200410

[9] M. R. Bussieck, A. Meeraus, *General algebraic modeling system (gams)*, Springer US, Boston, MA, (2004), 137-157. https://doi.org/10.1007/978-1-4613-0215-5-8

[10] S. Chanas, W. Kolodziejczyk, *Maximum flow in a network with fuzzy arc capacities*, Fuzzy Sets and Systems, **8**(2) (1982), 165-173. https://doi.org/10.1016/0165-0114(82)90006-9

[11] J. C. N. Clímaco, M. M. B. Pascoal, J. M. F. Craveirinha, M. E. V. Captivo, *Internet packet routing: Application of a k-quickest path algorithm*, European Journal of Operational Research, **181**(3) (2007), 1045-1054. https://doi.org/10.1016/j.ejor.2006.03.013

[12] S. Dan, S. Majumder, M. B. Kar, S. Kar, *On type-2 fuzzy weighted minimum spanning tree*, Soft Computing, **25** (2021), 14873-14892. https://doi.org/10.1007/s00500-021-06052-1

[13] A. Dey, S. Mondal, T. Pal, *Robust and minimum spanning tree in fuzzy environment*, International Journal of Computing Science and Mathematics, **10**(5) (2019), 513-524. https://doi.org/10.1504/IJCSM.2019.103679

[14] A. Dey, A. Pal, *Prim's algorithm for solving minimum spanning tree problem in fuzzy environment*, Annals of Fuzzy Mathematics and Informatics, **12**(3) (2016), 419-430.

[15] A. Dey, L. H. Son, A. Pal, H. V. Long, *Fuzzy minimum spanning tree with interval type 2 fuzzy arc length: Formulation and a new genetic algorithm*, Soft Computing, **24** (2020), 3963-3974. https://doi.org/10.1007/s00500-019-04166-1

[16] S. Dhouib, S. Broumi, M. Talea, *Solving the minimum spanning tree problem under interval-valued fermatean neutrosophic domain*, Neutrosophic Sets and Systems, **67**(1) (2024), 2. https://doi.org/10.5281/zenodo.11098903

[17] D. Di Caprio, A. Ebrahimnejad, H. Alrezaamiri, F. J. Santos-Arteaga, *A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights*, Alexandria Engineering Journal, **61**(5) (2022), 3403-3415. https://doi.org/10.1109/ICCCNT56998.2023.10308130

[18] D. Dubois, H. Prade, *Shortest path algorithms with fuzzy data (in French)*, RAIRO-Oper. Res., **12**(2) (1978), 214-227. https://doi.org/10.1051/ro/1978120202131

[19] C. Dudeja, *PSO based constraint optimization of intuitionistic fuzzy shortest path problem in an undirected network*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, **32**(03) (2024), 303-323. https://doi.org/10.1142/S0218488524500120

[20] I. Dumitrescu, N. Boland, *Algorithms for the weight constrained shortest path problem*, International Transactions in Operational Research, **8**(1) (2001), 15-29. https://doi.org/10.1111/1475-3995.00003

[21] A. Ebrahimnejad, M. Enayattabr, H. Motameni, H. Garg, *Modified artificial bee colony algorithm for solving mixed interval-valued fuzzy shortest path problem*, Complex and Intelligent Systems, **7** (2021), 1527-1545. https://doi.org/10.1007/s40747-021-00278-0

[22] J. Edmonds, R. M. Karp, *Theoretical improvements in algorithmic efficiency for network flow problems*, Journal of the ACM, **19**(2) (1972), 248-264. https://doi.org/10.1007/3-540-36478-1_4

[23] P. Erdos, A. Rényi, *On the evolution of random graphs*, Publ. Math. Inst. Hung. Acad. Sci, **5**(1) (1960), 17-60. https://doi.org/10.1515/9781400841356.38

[24] H. N. Gabow, *Scaling algorithms for network problems*, Journal of Computer and System Sciences, **31**(2) (1985), 148-168. https://doi.org/10.1016/0022-0000(85)90039-X

[25] F. Hernandes, M. T. Lamata, M. T. Takahashi, A. Yamakami, J. L. Verdegay, *An algorithm for the fuzzy maximum flow problem*, 2007 IEEE International Fuzzy Systems Conference, (2007), 1-6. https://doi.org/10.1109/FUZZY.2007.4295464

[26] S. Hong, J. K. Hedrick, *Roll prediction-based optimal control for safe path following*, 2015 American Control Conference (ACC). IEEE, (2015), 3261-3266. https://doi.org/10.1109/ACC.2015.7171835

[27] A. Iftar, E. J. Davison, *A decentralized discrete-time controller for dynamic routing*, International Journal of Control, **69**(5) (1998), 599-632. https://doi/abs/10.1080/002071798222596

[28] A. A. Keller, *Multi-objective optimization in theory and practice II: Metaheuristic algorithms*, Bentham Science Publishers, 2019. https://books.google.com/books?id=3uyRDwAAQBAJ

[29] W. Kolodziejczyk, *The shortest spanning tree problem in a network with fuzzy parameters*, Report 44, Institute of Production Engineering and Management, Technical University of Wroclaw, **44** (1984).

[30] R. Kumar, S. Jha, R. Singh, *A different approach for solving the shortest path problem under mixed fuzzy environment*, International Journal of Fuzzy System Applications (IJFSA), **9**(2) (2020), 132-161. https://doi.org/10.4018/IJFSA.2020040106

[31] A. Kumar, M. Kaur, *An algorithm for solving fuzzy maximal flow problems using generalized trapezoidal fuzzy numbers*, International Journal of Applied Science and Engineering, **8**(2) (2010), 109-118. https://doi.org/10.6703/IJASE.2010.8(2).109

[32] A. Kumar, M. Kaur, *An improved algorithm for solving fuzzy maximal flow problems*, International Journal of Applied Science and Engineering, **10**(1) (2012), 19-27. https://doi.org/10.6703/IJASE.2012.10(1).19

[33] S. Majumder, B. Saha, P. Anand, S. Kar, T. Pal, *Uncertainty based genetic algorithm with varying population for random fuzzy maximum flow problem*, Expert Systems, **35**(4) (2018), e12264. https://doi.org/10.1111/exsy.12264

[34] E. Q. V. Martins, J. L. E. Santos, *An algorithm for the quickest path problem*, Operations Research Letters, **20**(4) (1997), 195-198. https://doi.org/10.1016/S0167-6377(97)00008-4

[35] D. Medhi, K. Ramasamy, *Network routing: Algorithms, protocols, and architectures, second edition*, Morgan Kaufmann Publishers, Cambridge, MA (2017). https://doi.org/10.1016/C2013-0-18604-7

[36] A. Mert, *Defuzzification of non-linear pentagonal intuitionistic fuzzy numbers and application in the minimum spanning tree problem*, Symmetry, **15**(10) (2023), 1853. https://doi.org/10.3390/sym15101853

[37] A. Mohammadi, J. Tayyebi, *Maximum capacity path interdiction problem with fixed costs*, Asia-Pacific Journal of Operational Research, **36**(4) (2019), 1950018. https://doi.org/10.1142/S0217595919500180

[38] M. Parimala, S. Broumi, K. Prakash, S. Topal, *Bellman–Ford algorithm for solving shortest path problem of a network under picture fuzzy environment*, Complex and Intelligent Systems, **7** (2021), 2373-2381. https://doi.org/10.1007/s40747-021-00430-w

[39] Z. Peng, M. Nikbakht, A. Ebrahimnejad, F. Hosseinzadeh Lotfi, T. Allahviranloo, *Fully interval-valued fuzzy transportation problems: Development and prospects*, Computational and Applied Mathematics, **43**(1) (2024), 15. https://doi.org/10.1007/s40314-023-02523-3

[40] A. Plus, *What is of construction of shipping canal linking Caspian Sea and Persian Gulf to Tajikistan*, (2022). https://www.asiaplustj.info/en/news/tajikistan/economic/20221213/what-is-of-construction-of-shipping-canal-linking-caspian-sea-and-persian-gulf-to-tajikistan

[41] M. Pollack, *The maximum capacity through a network*, Operations Research, **8**(5) (1960), 733-736. https://doi.org/10.1287/opre.8.5.733

[42] A. P. Punnen, *A linear time algorithm for the maximum capacity path problem*, European Journal of Operational Research, **53**(3) (1991), 402-404. https://doi.org/10.1016/0377-2217(91)90073-5

[43] R. Ramaswamy, J. B. Orlin, N. Chakravarti, *Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs*, Mathematical Programming, **102**(2) (2005), 355-369. https://doi.org/10.1007/s10107-004-0517-8

[44] J. Shekel, *LINGO–A programming language for linear network analysis at a remote teletype terminal*, Proceedings of the IEEE, **55**(11) (1967), 2014-2015. https://doi.org/10.1109/PROC.1967.6034

[45] G. H. Shirdel, H. Rezapour, *Time-varying maximum capacity path problem with zero waiting times and fuzzy capacities*, SpringerPlus, **5**(1) (2016), 1-9. https://doi.org/10.1186/s40064-016-2654-y

[46] J. Tayyebi, A. Mitra, J. A. Sefair, *The continuous maximum capacity path interdiction problem*, European Journal of Operational Research, **305**(1) (2023), 38-52. https://doi.org/10.1016/j.ejor.2022.05.028

[47] S. Tragoudas, *The most reliable data-path transmission*, IEEE Transactions on Reliability, **50**(3) (2001), 281-285. https://doi.org/10.1109/24.974124

[48] Wikipedia contributors, *Iranrud — Wikipedia, The Free Encyclopedia*, (2023) [Online; accessed 19-March-2024]. https://en.wikipedia.org/w/index.php?title=Iranrud&oldid=1192043540