




Generalized Maximum Capacity Path Problem with Loss Factors

Javad Tayyebi¹^a, Mihai-Lucian Rîtan²^b and Adrian Marius Deaconu²^c

¹Department of Industrial Engineering, Birjand University of Technology, Industry and Mining Boulevard, Ibn Hesam Square, Birjand, Iran

²Department of Mathematics and Computer Science, Transilvania University of Braşov, Iuliu Maniu st. 50, Braşov, Romania

Keywords: Capacity Path Problems, Combinatorial Optimization, Polynomial Algorithms.

Abstract: The focus of this paper is on an extension of the maximum capacity path problem, known as the generalized maximum capacity path problem. In the traditional maximum capacity path problem, the objective is to find a path from a source to a sink with the highest capacity among all possible paths. However, this extended problem takes into account the presence of loss factors in addition to arc capacities. The generalized maximum capacity path problem is regarded as a network flow optimization problem, where the network comprises arcs with both capacity constraints and loss factors. The main goal is to identify a path from the source to the sink that allows for the maximum flow along the path, considering the loss factors while satisfying the capacity constraints. The paper introduces a zero-one formulation for the generalized maximum capacity path problem. Additionally, it presents two efficient polynomial-time algorithms that can effectively solve this problem.


1 INTRODUCTION


Combinatorial optimization is a special class of mathematical program that consists of finding an optimal object among a finite set of specific-structured objects. Some most prominent problems of this class are shortest path (SP) problems, maximum reliability path (MRP) problems, and maximum capacity path (MCP) problems. In these problems, the goal is to find an optimal path from an origin to a destination under a special objective function as follows:


1. $\min_{P \in \mathbb{P}} \sum_{(i,j) \in P} l_{ij}$ for SP problems,
2. $\max_{P \in \mathbb{P}} \prod_{(i,j) \in P} p_{ij}$ for MRP problems,
3. $\max_{P \in \mathbb{P}} \min_{(i,j) \in P} u_{ij}$ for MCP problems,

where the set \mathbb{P} consists of all paths from the origin to the destination. In this context, the variables l_{ij} , p_{ij} , and u_{ij} represent the length, reliability, and capacity of arc (i, j) respectively. Fortunately, these problems are tractable and there exist polynomial-time algorithms to solve them. For instance, the

shortest path problem can be solved using Dijkstra's algorithm with a Fibonacci-heap implementation, which has a complexity of $O(m + n \log(n))$ if the lengths, l_{ij} , are nonnegative. In case the lengths can be negative, the best-known algorithm is a FIFO implementation of the Bellman-Ford algorithm, with a complexity of $O(mn)$, where n and m represent the number of nodes and arcs, respectively. The maximum reliability path problem can be transformed into a shortest path problem by defining l_{ij} as $-\log(p_{ij})$ for every arc (i, j) . Consequently, it can be solved similarly to the shortest path problem, especially when p_{ij} is less than 1. Additionally, both the maximum reliability path problem and the maximum capacity path problem can be solved directly by modifying the shortest path algorithms. This is because they share similar optimality conditions with the shortest path problem (refer to Ahuja, 1988 for more details). However, the best-known algorithm for solving the maximum capacity path problem in an undirected network does not rely on this concept. Instead, it employs a recursive

^a <https://orcid.org/0000-0002-7559-3870>

^b <https://orcid.org/0009-0007-4601-6533>

^c <https://orcid.org/0000-0002-1070-1383>

algorithm with a linear complexity of $O(m)$ (Punnen, 1991).

The maximum capacity path problem finds its application in various domains. For instance, let us consider a network that represents connections between routers on the Internet. In this context, each arc in the network denotes the bandwidth of the corresponding connection between two routers. With the maximum capacity path problem, our objective is to discover the path between two Internet nodes that offers the highest possible bandwidth. This network routing problem is well-known in the field. Apart from being a fundamental network routing problem, the maximum capacity path problem also plays a crucial role in other areas. One noteworthy application is within the Schulze method, which is utilized for determining the winner of a multiway election (Schulze, 2011). In this method, the maximum capacity path problem aids in resolving ties and determining the strongest path among multiple alternatives. Additionally, the maximum capacity path problem finds application in digital compositing, wherein it assists in combining multiple images or video layers into a final composite image or sequence (Fernandez, 1998). By identifying the path with the maximum capacity, the compositing process can ensure the most efficient allocation of computational resources. Moreover, the problem contributes to metabolic pathway analysis, which involves studying chemical reactions within biological systems. In this context, the maximum capacity path problem aids in understanding the flow of metabolites through various pathways and identifying the most influential pathways in terms of capacity (Ullah, 2009). In summary, the maximum capacity path problem has extensive applications ranging from network routing on the Internet, multiway election methods, digital compositing, to metabolic pathway analysis. Its capability to identify and utilize paths with the highest capacity proves valuable across these diverse domains.

In this paper, a new combinatorial optimization problem called the generalized maximum capacity path (GMCP) problem is introduced. It is a more intricate version of the problem that involves finding a directed path from a given source node s to a given sink node t , with the minimum loss among all available directed paths from s to t (Deaconu, 2023). The GMCP problem is defined on a network where each arc is characterized by two attributes: capacity and loss factors.

The capacity of an arc represents the maximum flow value that can be transmitted through it. On the other hand, the loss factor of an arc indicates the flow

value that arrives at the tail node when one unit of flow is sent through the arc. The objective of the generalized maximum capacity path problem is to discover a path that is capable of transmitting the maximum flow while considering the loss factors.

This problem is inspired by an extension of maximum flow problems that incorporates loss factors, known as the generalized maximum flow problem (Ahuja 1993). Therefore, the algorithms developed for solving the GMCP problem can also be utilized as subroutines for addressing generalized maximum flow problems.

Moreover, the GMCP problem can be viewed as an extension of the maximum reliability path (MRP) and maximum capacity path (MCP) problems. It becomes equivalent to the MRP problem when capacities are infinite and transforms into the MCP problem when the loss factors are equal to 1. Thus, the GMCP problem expands upon the scope of both MRP and MCP problems, encompassing their characteristics and generalizations.

Overall, the GMCP problem introduces a novel combinatorial optimization problem that extends the concepts of maximum flow, MRP, and MCP by incorporating loss factors. The algorithms developed for GMCP can be utilized for generalized maximum flow problems, making it a versatile and applicable problem in various contexts.

The rest of this paper is organized as follows. In Section 2, we provide the necessary background information and definitions to lay the foundation for the research work. Section 3 clearly defines the research problem and outlines its significance. Section 4 describes in detail the proposed algorithms to solve the problem. Section 5 presents the experiments conducted to validate and evaluate the proposed algorithms. Finally, Section 6 summarizes the main findings of the paper and discuss their implications and potential future directions.

2 PRELIMINARIES

Consider a directed and connected network $G = (V, A, u)$, where V represents the set of nodes, A represents the set of arcs (each arc $a = (i, j)$ starts from node i and terminates at node j), and u is a capacity function mapping arcs to non-negative real numbers. Within this network, there are two special nodes: s , referred to as the source node, and t , referred to as the sink node. Let n denote the total number of nodes in the network ($|V|$), and m represent the number of arcs ($|A|$).

A path, denoted as P , from a node $w \in V$ to a node $v \in V$ in network G is defined as a sequence of nodes $P: (w = i_1, i_2, \dots, i_l = v)$, where l is equal to or greater than 1, and each consecutive pair of nodes (i_k, i_{k+1}) belongs to A , for every $k = 1, 2, \dots, l - 1$.

For the sake of simplicity, from now on, we refer to a path from s to t as an " st -path".

The capacity of an st - path P is denoted by $u(P)$ and is given by the minimum of its capacities, that is,

$$u(P) = \min \{u(a) | a \in P\}.$$

The maximum capacity path problem (MCP) in the network G is to find an st -path \tilde{P} having the maximum capacity among all st -paths:

$$u(\tilde{P}) = \max\{u(P) | P \text{ is an } s - t \text{ path}\}.$$

This problem is also called the widest path problem, the bottleneck shortest path problem, and the max-min path problem in the literature.

3 PROBLEM FORMULATION

In this section, we will delve into the problem of the Generalized Maximum Capacity Path (GMCP). Let us formally define the problem considering a connected and directed network denoted as $G(V, A, u, p)$. Here, p is the loss factor parameter.

For each arc $(i, j) \in A$, there are two key parameters associated with it. The first one is the capacity, denoted as u_{ij} , which represents the maximum amount of flow that can be sent along the arc. The second parameter is the loss factor, denoted as p_{ij} , which lies in the interval $(0, 1]$. The loss factor captures physical transformations such as evaporation, energy dissipation, breeding, theft, or interest rates (Tayyebi, 2019).

Considering the flow along the arcs, if x_{ij} units of flow enter arc (i, j) , only $p_{ij}x_{ij}$ units of flow are actually delivered to node j . This implies that $(1 - p_{ij})x_{ij}$ units of flow are absorbed or lost along the arc due to the specified loss factor.

The aim of the generalized maximum capacity problem is to find an st -path that enables the transmission of the maximum possible flow while taking the loss factors into consideration.

To formulate this problem in a precise manner, we introduce the following variables:

- x_{ij} , representing the flow entering arc (i, j) .

- y_{ij} , a binary variable that determines whether or not arc (i, j) carries a positive flow (1 if it does, 0 otherwise).

Now, we can express the GMCP as a mixed zero-one linear programming model, which can be stated as follows:

$$\max z = v_t \tag{1a}$$

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} p_{ji}x_{ji} = \begin{cases} v_s & i = s, \\ 0 & i \neq s, t, \forall i \in V, \\ -v_t & i = t, \end{cases} \tag{1b}$$

$$\sum_{j:(i,j) \in A} y_{ij} \leq 1, \quad \forall i \in V \setminus \{t\}, \tag{1c}$$

$$0 \leq x_{ij} \leq u_{ij}y_{ij}, \quad \forall (i, j) \in A, \tag{1d}$$

Let us break its points down into separate parts:

1. The variables v_s and v_t represent the flow leaving the source node s and the flow entering the sink node t , respectively.
2. Constraints (1b) and (1d) correspond to the balanced flow constraint and the bound constraints commonly found in maximum flow problems (Ahuja, 1988).
3. Constraint (1c) ensures that at most one outgoing arc from any node is capable of sending flow. This constraint guarantees that flow is sent only along a single st -path.
4. The formulation (1) of the GMCP problem closely resembles that of generalized maximum flow problems, with the added inclusion of zero-one variables y_{ij} and the constraint (1c) (Ahuja, 1988).

Remark 1: While we have assumed that $p_{ij} \leq 1$, constraint (1d) does not account for the possibility of a flow increment on arc (i, j) when $p_{ij} > 1$. In such cases, it should be written as $0 \leq \max \{x_{ij}, p_{ij}x_{ij}\} \leq u_{ij}y_{ij}$. To handle this situation without loss of generality, we can redefine the capacity of arc (i, j) as $\min \{u_{ij}, u_{ij}/p_{ij}\}$.

4 ALGORITHMS

This section focuses on the development of two algorithms to solve the GMCP problem in polynomial time. This provides evidence that the problem is tractable, similar to the MCP, SP, and MRP problems.

We start with a simple observation: if we send the maximum flow along a path, then at least one of its arcs will be saturated. The capacity of this saturated arc determines the flow value along the path. Let (i_p, j_p) be the last saturated arc in an st -path $P = s - \dots - i_p - j_p - \dots - k - t$. The flow value along path P is then equal to $u_{i_p j_p} \times (p_{i_p j_p} \times \dots \times p_{kt})$. An interesting insight is that if we remove the arc (i_p, j_p) from the network and add a new arc (s, j_p) with capacity $u_{s j_p} = u_{i_p j_p}$, loss factor $p_{s j_p} = p_{i_p j_p}$, then we can send the same flow value along the new path $s - j_p - \dots - k - t$. This simple idea leads to a polynomial-time algorithm for solving problem (1).

In the first algorithm, disregarding arc capacities, we aim to find a maximum reliability path from s to t , which is a path P where the value of the product $\prod_{(i,j) \in P} p_{ij}$ is maximized. This can be achieved by assigning arc lengths $l_{ij} = -\log(p_{ij})$ and finding the shortest st -path based on these arc lengths. Let P be the shortest path obtained. Then, we identify the last arc (i_p, j_p) in P that would become saturated if we were to send the maximum flow along P . We remove arc (i_p, j_p) from the network and introduce an artificial arc (s, j_p) with a loss factor $p_{s j_p} = p_{i_p j_p}$, capacity $u_{s j_p} = u_{i_p j_p}$ and a weight of $-\log(u_{i_p j_p} p_{i_p j_p})$. We repeat this process until we find a path P in which the last saturated arc is one of the artificial arcs.

Considering the unique characteristics of our algorithm, it is noteworthy to underscore that the negative weights exclusively pertain to arcs emanating from the source node s . This crucial distinction enables the seamless application of Dijkstra's algorithm, as its efficacy is contingent upon the absence of negative cycles within the graph. To further streamline the application of Dijkstra's algorithm and eliminate negative arcs altogether, we propose a judicious adjustment to the arc weights. Specifically, we suggest augmenting all arcs with the minimum value among the arcs originating from s , denoted as $\min(s, j)$ for the pair (s, j) . This augmentation ensures the absence of negative weights in the entire graph, rendering it amenable to Dijkstra's algorithm without any reservations.

Subsequently, upon identifying the optimal path and obtaining the computed result, we advocate for subtracting the added value—representative of the minimum value among the source-emerging arcs. This corrective measure guarantees the restoration of the original, unaltered values on the optimal path while harnessing the benefits of an adjusted graph

conducive to the successful application of Dijkstra's algorithm.

It is important to note that the optimal value of problem (1) is equal to the maximum flow along the last path found by the algorithm. To obtain the optimal path, we need to save the segment of path P from s to i_p whenever (i_p, j_p) is removed and (s, j_p) is added. This can be accomplished by introducing an additional parameter $P_{s j_p}$ for each artificial arc. Therefore, if the algorithm finds path P in the last iteration, the optimal solution will be a path that includes arcs from $P_{s j_p}$ and P , excluding (s, j_p) .

Algorithm 1 provides a formal description of our first algorithm. Since an arc is removed in each iteration, the number of iterations is at most equal to m (the total number of arcs). As a result, we can conclude the following:

Theorem 1: The complexity of Algorithm 1 is $O(mS(n, m))$ in which $S(m, n)$ is the complexity of finding the shortest path in the network.

Algorithm 1.

Input: An instance of the generalized MCP problem

Output: An optimal path

for $(i, j) \in A$:

if $i == s$:

 Set $\bar{l}_{ij} = -\log p_{ij} u_{ij}$ and $P_{ij} = (i, j)$

else:

 Set $\bar{l}_{ij} = -\log p_{ij}$ and $P_{ij} = \emptyset$

while True:

 Find a shortest path P with respect to \bar{l}_{ij}

if $P_{s j_p} \neq \emptyset$:

 The optimal path is $P_{s j_p} \cup P \setminus \{(s, j_p)\}$

else:

 Find the last arc (i_p, j_p) of P to be saturated.

 Remove (i_p, j_p) .

 Add an artificial arc (s, j_p)

 Set $\bar{l}_{s j_p} = -\log(u_{i_p j_p} p_{i_p j_p})$

In the followings, we discuss optimality conditions for problem (1) and present an algorithm with a time complexity of $O(S(m, n))$, which improves the complexity of Algorithm 1 by a factor of m .

To begin, we introduce a label $d(j)$ for each node $j \in V$. During intermediate stages of computation, the label $d(j)$ serves as an estimate (or an upper bound) of the maximum flow sent from the source node s to node j along a single path. At the termination of the algorithm, the label $d(j)$ represents the optimal value of problem (1). Our objective is to establish

necessary and sufficient conditions for a set of labels to accurately represent the maximum flow.

Let $d(j)$ denote the value of the maximum flow sent from the source node to node j (where we set $d(s) = +\infty$). In order for the labels to be optimal, they must satisfy the following necessary optimality conditions:

- ✓ Constraint (1c): For each node $j \neq s$, there exists at most one outgoing arc with positive flow. This condition ensures that the flow is sent only along a single st -path.
- ✓ Capacity Constraint: For each arc (i, j) , the flow through the arc must not exceed its capacity. Mathematically, this can be written as $f_{ij} \leq u_{ij}$, where f_{ij} represents the flow on arc (i, j) and u_{ij} represents the capacity of arc (i, j) .
- ✓ Flow Conservation: The flow conservation principle must be satisfied at every node (except the source and sink nodes). For any node $j \neq s$ and $j \neq t$, the sum of incoming flows must equal the sum of outgoing flows. Mathematically, this can be expressed as $\sum_{(i,j) \in A} f_{ij} - \sum_{(j,k) \in A} f_{jk} = 0$.
- ✓ Optimality Condition: For each node $j \neq s$, the label $d(j)$ represents the maximum flow sent from the source node s to node j . Therefore, we have $d(j) = \sum_{(i,j) \in A} f_{ij} - \sum_{(j,k) \in A} f_{jk}$, where f_{ij} represents the flow on arc (i, j) and f_{jk} represents the flow on arc (j, k) .

By satisfying these necessary optimality conditions, we can ensure that the labels $d(j)$ accurately represent the maximum flow in the network.

If the labels are optimal, they must satisfy the following necessary optimality conditions:

$$d_j \geq p_{ij} \min\{u_{ij}, d_i\}.$$

This is an extension of the optimality conditions of both the MCP and MRP problems. On the optimal path, the inequality is satisfied in the equality form. It states that the label of node j is either $p_{ij}d_i$ or $p_{ij}u_{ij}$. In the case that the flow value arrived at node i is less than u_{ij} , (namely, $d_i < u_{ij}$), this arc is not saturated and consequently, $d_j = p_{ij}d_i$. In the other case, (i, j) is saturated, and it interdict sending flow more than its capacity. So, $d_j = p_{ij}u_{ij}$ in this case.

Since this optimality condition is similar to that of the SP problem, we can apply the concept of Dijkstra's algorithm to solve problem 1. This is presented in Algorithm 2.

Theorem 2: Algorithm 2 solves the problem in $O(n^2)$ time.

Proof. Considering the provided information, we can deduce that in the given for loop, each arc is checked only once. As a result, the number of iterations executed by the two last lines of the loop is at most $O(m)$ which is also less than or equal to $O(n^2)$, considering the worst-case scenario.

Conversely, the node selection process, where the node with the minimum label is chosen, requires $O(n)$ time in each iteration. Taking into account that the number of iterations is $O(n^2)$, we can conclude that the most time-consuming operation in this algorithm is the node selection, which takes $O(n^2)$ time.

Algorithm 2.

Input: An instance of the generalized MCP problem

Output: An optimal path

```

for  $i \in V$ :
    Set  $d(i) = 0$ 
Set  $d(s) = +\infty$ 
Set  $S(0) = s; \bar{S} = V$ 
while  $|S| < n$ :
    Let  $i \in S$  be a node for which
     $d(i) = \min\{d(j) : j \in S\}$ ;
     $\bar{S} = \bar{S} \setminus \{i\}$ ;
    for each  $j \in V : (i, j) \in A$ :
        if  $d_j < p_{ij} \min\{u_{ij}, d_i\}$ :
            Update  $d_j = p_{ij} \min\{u_{ij}, d_i\}$ 
            Update  $S = S \cup \{i\}$ ;
    
```

We can also use the available implementations of Dijkstra's algorithm for the complexity improvement of Algorithm 2. For example, the Fibonacci heap implementation reduces the complexity to $O(m + n \log n)$.

5 EXPERIMENTS & DISCUSSIONS

Based on the findings presented in Theorem 1 and Theorem 2, it is obvious that Algorithm 2 outperforms Algorithm 1 in terms of speed. However, Algorithm 1 offers an advantage in that it can be effectively parallelized on GPUs by utilizing classical Dijkstra's algorithm (Ortega-Arranz, 2013) in a sequential manner. So, the implementation of CUDA-based Dijkstra's algorithm resulted in a noteworthy acceleration of Algorithm 1.

The CUDA version exhibited significantly improved performance compared to Algorithm 2, as demonstrated in Table 2. Particularly, for larger

network instances with 10,000 nodes or more, the CUDA implementation of Algorithm 1 was up to 25.7 times faster than Algorithm 2.

Table 1: Network generator parameters used for experiments.

No. of nodes	No. of Instances	No. of paths	No. of cycles	Erdős–Rényi prob.	No.
1000	10000	500	100	0.5	1
		500	100	0.7	2
		500	100	0.9	3
2000	1000	1000	100	0.1	4
		1000	250	0.15	5
		1000	500	0.6	6
		2500	1000	0.1	7
5000	100	2500	1000	0.2	8
		2500	1000	0.3	9
		5000	2500	0.15	10
10000	5	5000	2500	0.3	11
		5000	2500	0.5	12
		7500	1000	0.15	13
20000	2	7500	1000	0.15	14
25000	1	8000	1500	0.15	15

Table 2: Running times (ms) comparison between Algorithm 1 (CPU and GPU) and Algorithm 2 CPU.

No.	Alg. 1 CPU	Alg. 1 GPU	Alg. 2 CPU	Alg. 1 GPU vs Alg. 2 CPU(times)
1	165.75	187.30	18.32	0.10
2	121.1	138.66	8.24	0.06
3	188.96	311.78	14.84	0.05
4	30.22	52.40	6.86	0.13
5	120.00	209.34	63.00	0.30
6	193.50	315.60	19.50	0.06
7	183.00	71.52	40.15	0.56
8	284.30	103.41	41.7	0.40
9	626.6	226.74	61.4	0.27
10	671.10	89.21	98.01	1.10
11	677.14	67.78	216.00	3.19
12	940.20	76.15	358.01	4.70
13	1306.07	49.41	238.10	4.82
14	2965.04	53.48	452.11	8.46
15	3549.10	32.11	826.04	25.72

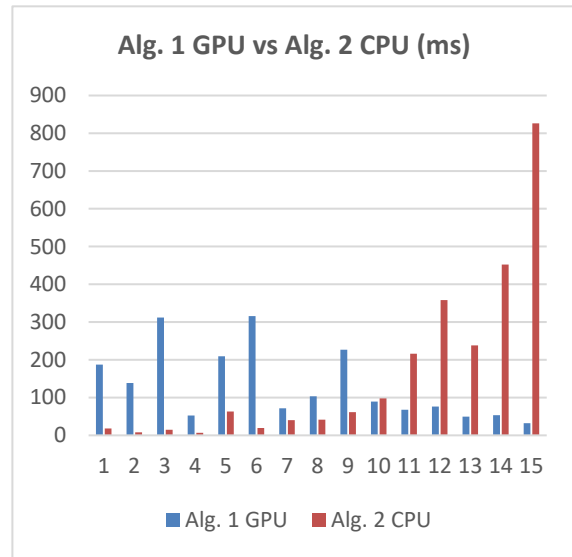


Figure 1: Running times (ms) comparison between Algorithm 1 GPU and Algorithm 2 CPU.

We used different instances of random generated networks having the number of nodes varying from 1000 to 25000 (see Table 1). The networks were generated using the network random generator from (Deaconu, 2021).

It is imperative to acknowledge that, throughout various experimental analyses, graph's density played an important role, and was calibrated using the Erdős–Rényi probability distribution. The Erdős–Rényi variable, a continuous parameter spanning the interval $[0, 1]$, played a pivotal role in these experimental investigations.

In this context, it is essential to elucidate that the Erdős–Rényi variable assumes a value of 0 when the graph lacks any new arcs, signifying minimal density. Conversely, a value of 1 denotes the graph's attainment of maximum density, highlighting the comprehensive spectrum of density exploration in our experimental framework.

The experiments were performed using a PC with an Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz, 24 GB RAM, and an NVIDIA GeForce GTX 1070 TI graphics card. The algorithms were programmed in Visual C++ 2022 under Windows 10.

6 CONCLUSIONS

In this paper, a novel combinatorial optimization problem was introduced. The main objective is to identify a path that can transmit the maximum flow, taking into account both the capacities and loss

factors of the arcs. This problem is known as the generalized maximum capacity path problem (GMCP).

The paper presents two strongly polynomial algorithms to address the GMCP. The first algorithm has a time complexity of $O(mS(n, m))$, where $S(m, n)$ represents the time complexity of finding the shortest path in the network. On the other hand, the second algorithm has a more efficient time complexity of $O(n^2)$. However, it is worth noting that when the first algorithm is implemented on GPUs, it performs substantially faster than the second algorithm, especially for large network instances.

REFERENCES

- Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows*, Prentice Hall.
- Deaconu, A.M., & Spridon, D., (2021). *Adaptation of Random Binomial Graphs for Testing Network Flow Problems Algorithms*. Mathematics, 9(15), 1716.
- Deaconu, A.M., Ciupala, L., & Spridon, D., (2023). *Finding minimum loss path in big networks*. 22nd International Symposium on Parallel and Distributed Computing (ISPDC), Bucharest, Romania, pp. 39-44.
- Fernandez, E., Garfinkel, R., & Arbiol, R. (1998). *Mosaicking of aerial photographic maps via seams defined by bottleneck shortest paths*. Operations Research, 46(3), 293-304.
- Ortega-Arranz, H., Torres, Y., Llanos, D.R., & Gonzalez-Escribano, A., (2013). *A new GPU-based approach to the Shortest Path problem*. 2013 International Conference on High Performance Computing & Simulation (HPCS), Helsinki, Finland, pp. 505-511.
- Punnen, A. P. (1991). *A linear time algorithm for the maximum capacity path problem*. European Journal of Operational Research, 53(3), 402-404.
- Schulze, M. (2011). *A new monotonic, clone-independent, reversal symmetric, and Condorcet-consistent single-winner election method*. Social choice and Welfare, 36, 267-303.
- Tayyebi, J., & Deaconu, A. (2019). *Inverse generalized maximum flow problems*. Mathematics, 7(10), 899.
- Ullah, E., Lee, K., & Hassoun, S. (2009, November). *An algorithm for identifying dominant-edge metabolic pathways*. In 2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers (pp. 144-150). IEEE.