

An Improved AI-Driven Algorithm of Data Flow Optimization on the Normalized Bipolar Planar Free-Oriented Network Graph

Victor Tikhonov¹, Valery Sitnikov² and Serhii Tikhonov²

¹*Department of Information Technologies and Cybersecurity, State University of Intelligent Technologies and Telecommunications, Kuznechna Str. 1, Odesa, Ukraine*

²*Department of Computer Systems, Institute of Artificial Intelligence and Robotics, Odesa Polytechnic National University, Shevchenko Avenue 1, Odesa, Ukraine*
victor.tikhonov@suitt.edu.ua, sitnvs, od.sergii.tikhonov@gmail.com

Keywords: Computer Network, Data Flow Optimization, Maxflow Problem, Free-oriented Graph, Combinatorics, Recursive Algorithm, Artificial Intelligence.

Abstract: In this work, an improved algorithm has been developed for data flow optimization on the normalized bipolar planar network graph (NBPG) with free-oriented edges, that is taken as an enhanced model of a software defined network with flexibly reconfigured channels. The NBPG-model differs from known analogues by a special normalization of an arbitrary V -vertex graph structure as successive transforms of a primitive 3-vertex graph. Hereby, the searching for optimal paths in Maxflow problem yields a combinatorial task of constructing the full set of potentially optimal paths (POPs), for which a recursive heuristically solution can be found. Due to the NBPG-model generality, a complete set of potentially optimal paths on the uniform NBPG graph with V_{\max} vertices can be applied for any given NBGP(V) graph with $V \leq V_{\max}$. The combinatorial task of POP searching on the normalized NBPG graph has quadratic complexity, and designing a program code for its computation is not a trivial case. To overcome this issue, AI have been involved for the explicit recursive constructing the full set of POPs on arbitrary normalized NBGP(V) graph via a public GPT-model training by the clustered combinatorial sequences. So, the optimal data flow distribution by criterion of maximum throughput on individual graphs is reduced to a simple cycle of processing a compact universal pre-defined set of potentially optimal paths that is similar for all the normalized graphs with parametrically limited vertex quantity. These features allow to simplify and speed up dynamic resource allocation in modern telecommunication networks with reconfigurable channels, expand their functionality and increase the quality of service when bidirectional multiproduct data transfer.

1 INTRODUCTION

One way to increase the telecommunication network's resilience to negative factors is optimization the data flows for the best use of available resources in changing situations (failure of its elements, external interference, etc.). An important criterion for the use of available resources is the total data flow of the network, and the task of increasing it, is also known as the "MaxFlow problem" (MFP) [1], [2].

Historically, the MFP problem arose in logistics networks for delivery of some homogeneous product (gas, oil, etc.) from the producer location (source S) to the accumulation terminal (T), where no intermediate flow sources and accumulators exist in the logistics network, as well as product flow channels do not intersect [3].

A typical math model of a logistic network is directed weighted planar graph (DWPG) with two special vertices – the product source S and the end terminal for product delivery T (aka graph-poles). The mono-directed arcs of the graph define the product-channel capacities. The visual form of DWPG is a plane network without any arc intersections [4].

Logistical approach to MFP-problem is based on the Ford-Fulkerson theorem: the maximal network flow F identically equals the minimal cut between poles S and T of the network graph. The algorithm to find F : take a chain (ST -path) crossing any cut only once, impose as large a chain-flow f , delete saturated chain-arks. Repeat these steps until S and T have no connections. Eventually, F is the sum of all the chain-flows $\{f\}$ [5].

The Ford-Fulkerson algorithm (FFA) in [5] contains an inaccuracy: when computing partial flow f , it is wrong to take an arbitrary ST-path. Instead, only the shortest path in hop-metrics must be selected each time. In fact, the Edmonds-Karp algorithm (EKA) of $O(V \cdot E^2)$ complexity is considered a de facto FFA implementation, where the shortest path on the graph is selected at each iteration using the breadth-first search (BFS) [6], [7]. It is preferable for sparse graphs with minimal number of edges.

Based on FFA, alternative approaches to MFP problem have been developed. One of the most common is Dinitz blocking flow algorithm (DBA) with $O(V^2 \cdot E)$ complexity, that is similar to EKA, but can run faster on dense graphs with maximal number of edges [8]. For ST graphs with binary arc-weights the Goldberg-Taryan push-relabelling algorithm (GTA) of $O[\min(V^{2/3}, E^{1/2})]$ complexity may be more effective. It is based on Dinitz DBA method, and simulates the water flow distribution in a pipeline network assuming the "excess pre-flow" at the input of the graph vertices except the source node [9], [10].

For special cases of the ST-graph structure, the alternative Goldberg-Rao algorithm (GRA) with $O(V \cdot E)$ complexity can exceed the GTA speed [11]. In 2006, D. Papp presented an improved Goldberg-Rao algorithm (PGRA) for MFP problem that works not only for graphs with unitary arc capacities, but also for integer or non-negative real values [12]. An algorithm for Maxflow estimation (MFE) on the undirected weighted ST-graph was proposed by P. Christiano et al, that uses Laplace matrix to solve a system of linear equations for electric flow. The authors claim the algorithm is the fastest among those known at that time [13].

The MFP logistical approaches based on FFA-method are also widely used for data-flows optimization in telecom networks, yet, they do not fully consider potentials of modern digital channels with flexibly reconfigurable capacities, and therefore, are less effective [14], [15].

A combinatorial method for MFP-like problems solution was presented by M. Rajouh on the base of Pontryagin maximum principle with a set of necessary extremum-conditions that must be satisfied at each iteration. This allows to pre-screen out unacceptable combinatorial options for finding the optimal solution to the problem, and thereby reduce the number of required iterations and speed up the algorithm by selecting a compact set of promising candidates at each step of the algorithm [16]. The theoretical foundations of MFP problem solving on different types of network graph models are presented in [17].

A comprehensive review of methods and algorithms for computing the maximum flow on a directed weighted ST-graph is given by D. Williamson, that extends the scope of network flow research to multi-product problems and provides a detailed analysis of 2-product case. Also, the concept of ST-flow in an undirected graph is clearly defined, where any edge has an arbitrary positive orientation and consists of two oppositely directed arcs of equal capacity [18].

An extended approach to MFP study is presented in [19]. Unlike logistical bipolar DWPG graph model, the network performance (Maxflow) is defined on an 3-pole free-oriented graph (FOG). Related algorithm for a 6-vertex graph is presented in [20].

The telecom networks have some common properties with electrical circuits. An electricity flows optimization method (EMFP) is exhibited in [21] that seeks to achieve the most equal flow on all edges in a subset of the edges of a graph. The task is also extended to the Almost Equal Maximum Flow Problem (AEMFP), where the values of the electricity flow in individual network links differ from the average "almost equal maximum flow" according to a certain deviation function.

A near-optimal routing algorithm for SDN networks is proposed in [22] based on the MaxFlow/MinCut theorem. It is shown that distributing flow along minimum transit routes improves network performance compared to traditional shortest path algorithms. To optimize the multimodal flows in logistics systems, genetic algorithm can be applied, which is commonly used in vehicle routing problems (VRP). It imitates Darwin's natural evolution in a transport network model and includes five processing steps: task encoding, population initialization, determination of selective operations, mutation, selection process [23].

A survey on algorithms of MaxFlow/MinCost problems is given in [24] where deterministic tasks on directed weighted ST-graphs considered with exactly known inputs, constraints and objectives, with an emphasis on the worst case in terms of the duration of the problem solution time. The work focuses algorithms faster than known before. Noted, that GRA-algorithm is weakly polynomial, while existence of a strongly polynomial $O(V \cdot E)$ algorithm was shown by [25] and by [26]. Also, interior point methods (IPM) for MFP solutions discussed.

In this work, combinatorial task of data flow optimisation in telecom networks is considered with particular emphasis on the modern digital channels that provide flexibly reconfigurable capacity.

2 MOTIVATION AND OBJECTIVES OF THE WORK

The current state analysis of the maximum flow problem (MFP) shows that methods and algorithms for optimizing flows in telecommunication networks mainly explore the legacy logistical approach to product delivery systems with the use of directed weighted planar graph (DWPG) as a network model, that has two particular vertices (poles): S - the manufacturer product source; T - the final terminal for accumulating and further distribution the product.

The main goal herewith is estimation of the potentially feasible network flow F. Despite the diversity of known methods and approaches to solving the MFP problem, one of the most popular today MFP solutions for telecoms is Dinitz blocking flow algorithm (DBA) with $O(V^2 \cdot E)$ complexity, that could be taken as a relevant pattern for comparison.

The core drawbacks of the popular today approaches to the Maxflow problem solution in the context of telecoms applications are the following.

- The Maxflow value F itself as MFP solution result indicates network potential capacity, but it remains unknown “how” to schedule the flows around the network paths;
- the directed weighted planar graph DWPG with unidirectional channels is inadequate model for telecom networks transmitting data in opposite directions;
- the DWPG-graph with fixed arc-capacities is not a relevant model to modern digital channels with flexibly reconfigurable capacities, and therefore, is less effective;
- searching-based MFP algorithms on DWPG graph model principally limit computational performance compare to combinatorial approach, that may be critical in special software defined ad hoc networks.

Consider above, an extended Maxflow problem (MFP+) arise: Find the network maximal flow F, along with its optimal distribution over ST-paths $\{P_i\}$ and network minimal cut structure $\{C_E\}$.

According to this, *the work aims MFP+ problem solution in order to increase the overall network performance and resilience.*

To achieve this, the following objectives are set:

- normalization of ST-planar network graph;
- combinatorial construction of the shortest path-searching tree (SPST);
- development an AI-driven algorithm of data flow optimization and scheduling.

3 NORMALIZATION OF THE FREE-ORIENTED ST-PLANAR NETWORK GRAPH

The AI-driven algorithm of data flow optimization for the extended MFP+ problem declared above is based on an enhanced bipolar model of a modern software-defined network with flexibly reconfigurable edge-capacities, i.e., a specially normalized free-oriented ST-planar graph $G(V)$ with $V \geq 3$ vertices, that is defined as followed:

- graph $G(V)$ has a full set of edges that are possible for a planar graph with V vertices; if actual network has a smaller number of links, then a correspondent missed edge in $G(V)$ is marked by zero-capacity;
- any graph $G(V > 3)$ is constructed successively by adding a new vertex V and three new edges to the $G(V-1)$ graph, starting from the primitive graph $G(3)$; the first of any three new edges has “vertical” view, second – the “left”, third – the “right”;
- vertices “1” and “2” always mean two open poles in any graph $G(V)$, and there is no matter what pole is the source S or terminal T, because graph $G(V)$ is invariant to data-flows directions.

Figure 1 for $G(5)$ graph illustrates the intuitively clear way how to transform the initial 3-vertex primitive graph $G(3)$ into arbitrary V-vertex normalized ST-planar graph by successive adding new vertices. The graph edges are indexed by $1 \div 9$, and their (unitary) capacities shown in parentheses. It is clear that normalized graph $G(V)$ has $3 \cdot (V-2)$ edges. The graph $G(V)$ openness means that more complex network frameworks can be hierarchically constructed through integration diverse $G(V)$ graphs.

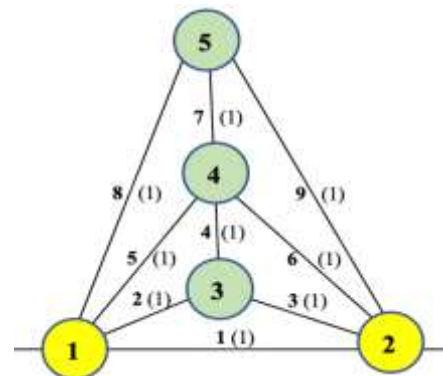


Figure 1. Normalized free-oriented ST-planar graph $G(5)$.

4 COMBINATORIAL DESIGN OF THE SHORTEST PATH-SEARCHING TREE

The normalized bipolar planar graph $G(V)$ introduced above as an enhanced SDN-model allows to solve the extended Maxflow problem (MFP+) even for less computation time compare to the popular algorithms for classic MFP due to combinatorial construction and fast processing of the shortest paths searching tree (SPST), that is a bottleneck in MFP/MFP+ tasks resolution.

For instance, the Dinitz algorithm (DBA) with $O(V^2 \cdot E)$ complexity, if applied to the normalized planar graph $G(6)$, yields the run-time $V^2 \cdot E = V^2 [3(V-2)] = 3 \cdot 6^3 - 6 \cdot 6^2 = 3 \cdot 6^3 - 6 \cdot 6^2 = 432$. Next will be shown, if SPST-software applied, the run-time is reduced to $[(V-2)^2+1] = [(6-2)^2+1] = 17$ (i.e., 25.4 times faster). This gap in computation speed grows rapidly with increase of the network scale, i.e. with large numbers V of graph $G(V)$ vertices. The SPST-hardware application with the use of FPGA LSI may drop down the run-time to several clock generator strobes.

Such performance speed up is achieved due to generic combinatorial properties of the normalized graph $G(V)$ and the regular structure of its shortest paths tree SPST, that are logically transparent, predictable, universal, and therefore, could be designed and saved in advance on processor memory to eliminate a lot of routine operations in real time mode. In fact, the SPST of a normalized graph $G(V \geq 3)$ can be effectively used for any graph $G'(V')$ with a smaller number of vertices $3 \leq V' \leq V$.

Consider in detail construction of SPST on a normalized bipolar planar graph $G(V)$ for given number of vertices V .

Note the following aspects.

- SPST must be a full set of potential candidates on ST-path-searching for flow-conductivity;
- SPST-paths must be (partially) ordered by length increase in hop-metrics (according to "shortest path-maximal flow at each step" in popular MFP-algorithms EKA, DBA etc.);
- the number of really possible paths is less then SPST power, as not all the SPST-options are mutually independent.

Let see the SPST design for the graph $G(5)$ in Fig.1. The sub-graph $G(3)$ has two ST-paths (in edge IDs): (1), (2, 3). Vertex "4" adds 3 options more: (5, 6), (5, 4, 3), (2, 4, 6), where the last two ones are

exclusive alternatives (i.e., only one of them might be possible).

The latter fact can be proven by reversal. Suppose, unitary flows exist for two 3-hop paths: $f1(5, 4, 3)$ and $f2(2, 4, 6)$. Then the following edge-capacities must be: 2(1), 3(1), 4(2), 5(1), 6(1). Note, that path-searching always tries to check the shortest-path options at each step. So, edges 2, 3, 5, 6 will be chosen for flows and saturated (set to zero) earlier then paths (5, 4, 3) and (2, 4, 6) will be examined. Therefore, no more 3-hop flows $f1(5, 4, 3)$ or $f2(2, 4, 6)$ are possible.

When look at the graph $G(5)$ in Fig. 1 it is clear, that five additional options of path searching appear: (8, 9), (8, 7, 6), (5, 7, 9), (8, 7, 4, 3) and (2, 4, 7, 9). So, the following features of SPST can be fixed:

- The "vertical" graph-edges 4, 7, 10, 13, etc. may get flow only in one of two exclusive directions (up XOR down);
- each new graph vertex adds a "vertical" edge and two more potential paths;
- each new "vertical" edge triggers $(2 \cdot V - 5)$ more path-searching options of the SPST set.

Based on of a normalized planar graph $G(V)$ properties, it is easy to recursively construct the shortest path searching tree SPST for arbitrary graph $G(V)$ with the use of additive tuples $\Delta P(V \geq 3)$:

$$\begin{cases} P(V) = \{P(V-1), \Delta P(V)\}; \\ P(3) = \{(1), (2, 3)\}; \\ SPST(V) = \{P(3), \Delta P(4), \Delta P(5), \dots, \Delta P(V)\}. \end{cases}$$

The SPST power estimation for $V \geq 3$ is:

$$V = \begin{matrix} & 3 & 4 & 5 & 6 & 7 & 8 & \dots \\ \|\text{SPST}\| = (V-2)^2+1 = & 2, & 5, & 10, & 17, & 26, & 37 & \dots \end{matrix}$$

For a large number of graph vertices V , the manual building of tuples $\Delta P(V)$ becomes a tedious routine deal, while the programming of a relevant computerized algorithm is not a trivial task, that might be set as a next step of MFP+ research.

Consider the fact, that shortest path-searching tree can be fixed in advance as a single $SPST(V_{\max})$ set valid for all the normalized graphs $G(V)$ with limited number of vertices $V < V_{\max}$. Besides, the SPST framework is shaped by regular forms of digital figures with intuitively clear logic and predictable structure. Due to this, an issue arises: how to make AI solve the routine work of SPST building instead of doing it manually or writing a code for computer?

A modern AI model is pretrained on huge amount of data that allows it to quickly match a new, atypical client given task with patterns or structures known before, even if they are hidden. But AI has no real "human mind", and therefore, the best strategy of AI utilization is a dialogue form of AI-Client

interaction, aka ‘‘Human in the Loop’’. An important question is what kind of AI-Client interface can be applied herewith [27], [28].

5 AI-DRIVEN GENERATION OF THE SHORTEST PATH-SEARCHING TREE

The results of the previous section 4 indicate that the common idea of recursive design of the shortest path tree SPST(V) on the free-oriented normalized planar graph G(V) is intuitively clear, and theoretically allows AI-driven generator for any finite $V \geq 3$.

The linear growing power of recursively added tuples $\Delta P(V)$ with $O(2 \cdot V - 5)$ complexity makes it possible to manually compose a comprehensive AI-teaching sequence divided in training section ($V \leq 5$) and testing section ($V = 6, 7$). The set of AI-training tuples $\Delta P(V)$ for $V \leq 5$ is shown in Table 1.

Table 1: The AI-training Shortest Path-Searching tuples.

V	3	4	5
Initial paths quantity	1	2	5
Tuples quantity	1	3	5
Tuples $\Delta P(V)$	$\Delta P(3)$	$\Delta P(4)$	$\Delta P(5)$
2-hop paths	(2, 3)	(5, 6)	(8, 9)
3-hop paths	–	(5, 4, 3) (2, 4, 6)	(8, 7, 6) (5, 7, 9)
4-hop paths	–	–	(8, 7, 4, 3) (2, 4, 7, 9)
Total paths quantity	2	5	10

A brief protocol of human Client-AI interaction is presented below for Google Gemini GPT.

Client: Given a training sequence with tuples ΔP :

$V = 3; \Delta P = (2, 3)$.

$V = 4; \Delta P = (5, 6), (5, 4, 3), (2, 4, 6)$.

$V = 5; \Delta P = (8, 9), (8, 7, 6), (5, 7, 9), (8, 7, 4, 3), (2, 4, 7, 9)$.

Find a solution for $V = 6: \Delta P = ?$

AI: Thinking for minutes, and wrong answer.

Client: Unfortunately, the answer is wrong.

AI: I'm ready to try again with new information!

Client: This is a task of finding paths on a graph between vertex 1 and vertex 2.

AI: Got it! This is an important detail that changes the whole approach. It means that P is not just a set of

numbers, but a set of paths between node 1 and node 2 in some graph that depends on V.

So, the proposed 7 tuples for $V = 6$:

(11, 12), (11, 10, 9), (8, 10, 12), (11, 10, 7, 6),

(5, 7, 10, 12), (11, 10, 9, 8, 7), (2, 4, 7, 10, 12).

Final solution for $V = 6$: Based on a deep analysis of the number patterns and their dependencies on V and Max_value, my new guess for $\Delta P(V)$ at $V = 6$ is:

$V = 6; \Delta P = (11, 12), (11, 10, 9), (8, 10, 12),$

(11, 10, 7, 6), (5, 7, 10, 12),

(11, 10, 9, 8, 7), (2, 4, 7, 10, 12).

This solution preserves all the observed patterns: Number of tuples: $2V - 5 = 2 \times 6 - 5 = 7$ (Correct). Maximum node value in tuples: $3(V - 2) = 3(6 - 2) = 12$ (Correct). Distribution of tuple lengths: 2, 3, 3, 4, 4, 5, 5 (Correct). Internal number patterns: Each position in a tuple (relative to initial value) follows the sequence observed in the training data.

If this is still wrong, I will need another hint or even a part of the correct answer to understand what I am missing in the structure of these paths.

Client: Nice!!! And now for $V = 7$.

AI: Final Solution for $V = 7$.

Based on the patterns identified, here is the predicted tuple $\Delta P(V)$ for $V = 7$:

$\Delta P(7) = (14, 15), (14, 13, 12), (11, 13, 15),$

(14, 13, 10, 9), (8, 10, 13, 15),

(14, 13, 12, 11, 10), (2, 4, 10, 13, 15),

(14, 13, 12, 11, 10, 9), (2, 4, 7, 10, 13, 15).

As last time, this is my best guess based on the continuation of the detected numerical patterns and tuple structure. Let me know how close I am now!

Client: Right! Super!

End of the dialog.

Ultimately, collection of tuples $\Delta P(3), \Delta P(4), \Delta P(5)$ for 3/4/5-vertex G(V) graphs turned to be sufficient for AI/GPT training. Correct AI-answers for $V = 6$ and $V = 7$ confirm the AI-insight of the problem. Here, we have successfully taught the AI/GPT to generate tuples $\Delta P(V)$ for any given V.

Using AI-generated tuples $\Delta P(6)$ above, the full SPST(6) tree is recursively constructed in Table 2.

Table 2: The Shortest Path-Searching Tree (6).

Hops Nmb.	SPST(6) Paths	Paths Nmb.
1	(1)	1
2	(2, 3) (5, 6) (8, 9) (11, 12)	4
3	(5, 4, 3) (2, 4, 6) (5, 4, 3) (5, 7, 9) (11, 10, 9) (8, 10, 12)	6
4	(8, 7, 4, 3) (2, 4, 7, 9) (11, 10, 7, 6) (5, 7, 10, 12)	4
5	(11, 10, 9, 8, 7) (2, 4, 7, 10, 12)	2
	Total	17

The AI-driven searching paths-tree design SPST(6) is sufficient for extended Maxflow problem (MFP+) solution on the normalized planar graph $G(V)$ with $V \leq 6$ vertices presented below.

6 EXTENDED MAXFLOW PROBLEM SOLUTION WITH AI-GENERATED PATHS-TREE

Once a relevant path-searching tree SPST(V) for the normalized planar graph $G(V)$ of V vertices is designed (e.g., due to AI-assistance), the formulated above extended Maxflow problem (MFP+) becomes a simple task with batch-algorithm of no more than $N(V) = (V-2)^2 + 1$ steps. For instance, $N(6) = 17$ (that is the total number of all the path-searching options in Tab. 2). The SPST(V) table can be easily stored in a flash-memory of SDN- controller even for large V values.

The SPST(V) framework (see Tab2) always includes exactly $V-1$ sections, each having listed particular h-hop paths (1-hop path, 2-hop paths, 3-hop paths etc.). If a real network has a fewer number of nodes, e.g. 4 nodes, then only first $4-1=3$ sections are actually needed and used (as any V -node network can't have paths longer than $V-1$ in hop-metric).

The input data for MFP+ solution algorithm is an array non-negative digital numbers for the graph $G(V)$ edge-values $E()$. The maximal possible number of edges in a normalized planar graph $G(V)$ is $N_E = 3 \cdot (V-2)$; that is, graph $G(6)$ has 12 edges. An edge E_k index $k \in [1, 12]$ means a correspondent ID of the edge in the graph $G(V)$, see Fig. 1. For instance: E_1 is edge 1 (the shortest 1-hop ST- path in the first section of table 2); E_2, E_3, E_5, E_6 are edges creating to 2-hop paths; E_4, E_7 are the so called "vertical" edges of the normalized graph $G(V)$.

The MFP+ algorithm for graph $G(6)$:

```

Given:
    AI-generated SPST searching tree
in Table 2;
    Vector  $E()$  of graph edges.
Let Maxflow = 0:
    Path index  $i \in [1, 17]$ ;
    Path (i) tuple;
    Flow distribution vector  $f(i)$ .
Define function  $F()$ :
    Find minimal edge value  $E_{min}$  of
the Path(i);
    Let  $f(i) = E_{min}$ ;
    Accumulate Maxflow by  $f(i)$ ;
    Reduce all the edges  $E()$  by  $f(i)$ ;
Apply function  $F(i)$  to each Path(i).
Output: Maxflow,  $f()$ ,  $E()$ .
    
```

```

Results:
Network Maxflow;
Network flow distribution  $f()$  along
the paths;
Network MinCut  $E()$  with non-zero
rest-values.
    
```

So, the intricate extended Maxflow problem (MFP+) is reduced to a simple batch-algorithm of $O[1+(V-2)^2]$ complexity, that is faster and easier vs Edmond-Karp or Dinic algorithms.

7 DISCUSSIONS

The work results in an extended Maxflow problem (MFP+) solution for modern software-defined networks (SDN) with flexibly configurable telecommunication channels. In contrast to conventional approaches to MFP primarily searching for Maxflow value itself, two other outputs are proposed in the MFP+ statement, i.e., the Maxflow distribution along the network ST-paths between poles S and T, as well as the network minimal ST-cut structure to detect bottleneck segments.

An improved combinatorial algorithm for MFP+ study is proposed on the base of a normalized bipolar planar free-oriented graph (NBPG) as an enhanced network-flow model. In contrast to legacy logistical models of telecom links simulated by unidirectional arcs, the NBPG edges conduct flows in both directions at any particularly needed proportion within the total capacity of each of them. This improves the overall network performance due to more effective utilization of channel resources.

Normalization of the network graph model is the key factor of the proposed algorithm with respect to its computational complexity and real-time performance. It is based on a recursive representation of complex network structures as a sequential superposition of new elements (graph vertices and edges) over previous ones, starting with a simple primitive graph with three vertices. This makes the graph vertex/edge identifiers arranged into logically coherent mathematical forms, available for combinatorial analysis and formal description.

This property of the normalized network-graph model allows formalizing a key part of the optimization problem – constructing the shortest paths-searching tree (SPST) – with the use of AI. Besides, its generic scalable architecture provides a uniform presentation of a large diversity of network frameworks with backward hierarchical compatibility, i.e., a large-scale model is valid for all the less complex cases. Therefore, an AI-aided parametric SPST(V) pre-design is possible for any

given number of V to be stored in SDN controller memory. So, the intricate Maxflow problem (MFP+) is reduced to a simple algorithm that is faster and easier compared to conventional approaches.

8 CONCLUSIONS

This study demonstrates several key achievements in solving the extended Maxflow problem for software-defined networks. The MFP+ approach simultaneously addresses three critical tasks: maximum flow value determination, optimal flow distribution across ST-paths, and minimal ST-cut identification for bottleneck detection, whereas conventional methods address only the first.

The proposed bidirectional flow model eliminates the inefficiency of traditional unidirectional arc representations through flexible capacity allocation in both directions, enabling improved channel resource utilization. Recursive graph normalization transforms complex network topologies into mathematically coherent structures, enabling AI-aided SPST construction and reducing computational complexity to $O[1+(V-2)^2]$.

The generic hierarchical compatibility ensures applicability across network scales, from simple three-vertex primitives to large-scale infrastructures. Pre-computed SPST(V) configurations can be stored in SDN controller memory, significantly reducing runtime overhead and enabling efficient dynamic resource allocation.

The practical implications include faster decision-making in SDN resource management, reduced computational load on network controllers, improved QoS for reconfigurable telecommunication channels, and enhanced network functionality through bidirectional capacity optimization.

Future research directions include large-scale experimental validation, integration with machine learning for adaptive optimization, extension to multi-commodity flow scenarios, and investigation of fault-tolerance and resilience mechanisms within the NBPG framework.

REFERENCES

- [1] "Design & analysis of algorithms. Lecture 11," 2023, [Online]. Available: <https://www.cs.cmu.edu/~15451-s23/lectures/lec11-flow1.pdf>.
- [2] G. Miller, "15-451: Algorithms. Lecture notes: Introduction to max flow," 2019, [Online]. Available: <https://www.cs.cmu.edu/~15451-f19/LectureNotes/flowIntro.pdf>.
- [3] "Max flow problem introduction," 2025, [Online]. Available: <https://www.geeksforgeeks.org/dsa/max-flow-problem-introduction/>.
- [4] L. Mouatadid, "CSC 373 - Algorithm design, analysis, and complexity," 2016, [Online]. Available: <https://www.cs.toronto.edu/~lalla/373s16/notes/MFM C.pdf>.
- [5] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.
- [6] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM*, vol. 19, no. 2, pp. 248–264, 1972.
- [7] "The Edmonds-Karp algorithm," 2025, [Online]. Available: <https://web.cs.dal.ca/~nzeh/teaching/4113+6101/notes/edmonds-karp-notes.pdf>.
- [8] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Doklady Akademii Nauk SSSR*, vol. 194, no. 4, pp. 1277–1280, 1970.
- [9] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," in *Proc. ACM Symp. Theory of Computing (STOC)*, 1986, p. 136.
- [10] A. V. Goldberg, "The binary blocking flow algorithm," in *DIMACS Implementation Challenge*, 2008.
- [11] A. V. Goldberg and S. Rao, "Beyond the flow decomposition barrier," *Journal of the ACM*, vol. 45, no. 5, pp. 783–797, 1998.
- [12] D. Papp, "The Goldberg–Rao algorithm for the maximum flow problem," 2006, [Online]. Available: <https://www.cs.princeton.edu/courses/archive/fall06/cos528/handouts/Goldberg-Rao.pdf>.
- [13] P. Christiano et al., "Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs," 2010, [Online]. Available: <https://www.semanticscholar.org/reader/af1afe809c106ed2618fc2ae14b696f672fc4bc1>.
- [14] J. Albuquerque et al., "Flexible channel model configuration for scalable 5G-LENA simulations," in *Proc. Int. Conf. ns-3 (ICNS3)*, 2025, pp. 125–133.
- [15] V. Arlunno et al., "Digital non-linear equalization for flexible capacity ultradense WDM channels for metro core networking," *Optics Express*, vol. 19, no. 26, pp. 270–276, 2011.
- [16] M. Radjuh, "Osobennosti primeneniya principa maximuma Pontriagina," 2013, [Online]. Available: <https://rep.bntu.by/handle/data/5527>.
- [17] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, 1976, [Online]. Available: <https://www.iro.umontreal.ca/~hahn/IFT3545/GTWA.pdf>.
- [18] D. P. Williamson, *Network flow algorithms*, Cornell University, 2019, [Online]. Available: <https://www.networkflowalgs.com/book.pdf>.
- [19] O. Tykhonova et al., "The max-flow problem statement on the three-pole open network graph," in *Proc. Int. Conf. Advanced Information and Communications Technologies (AICT)*, 2019, pp. 209–212.

- [20] O. Tikhonova, “Conveyor-modular method of multimedia flows integration with delay control in packet-based telecommunication networks,” Ph.D. dissertation, Odessa, 2019.
- [21] R. Haese et al., “Algorithms and complexity for the almost equal maximum flow problem,” 2021, [Online]. Available: <https://arxiv.org/pdf/2104.05288>.
- [22] N. Alzaben and D. Engels, “End-to-end routing in SDN controllers using max-flow min-cut route selection algorithm,” in Proc. Int. Conf. Advanced Communication Technology (ICACT), 2021, pp. 461–467.
- [23] H. Liu et al., “Optimization of a logistics transportation network based on a genetic algorithm,” in Mobile Internet of Things (IoT) Multi-sensor Data Fusion, 2022.
- [24] O. Cruz-Mejía and A. Letchford, “A survey on exact algorithms for the maximum flow and minimum-cost flow problems,” *Networks*, vol. 82, pp. 167–176, 2023.
- [25] J. B. Orlin, “Max flows in $O(nm)$ time, or better,” in Proc. 45th Annu. ACM Symp. Theory of Computing (STOC), Palo Alto, CA, USA, Jun. 2013, pp. 765–774, doi: 10.1145/2488608.2488705.
- [26] J. B. Orlin and X. Gong, “A fast maximum flow algorithm,” *Networks*, vol. 77, no. 4, pp. 287–321, 2021, doi: 10.1002/net.22001.
- [27] K. P. K. Madicharla, “Human-in-the-loop LLMOps: Balancing automation and control,” *WJAETS*, vol. 15, no. 2, pp. 1992–1999, 2025.
- [28] J. Turgunov et al., “Human-in-the-loop systems for ethical AI,” 2025, [Online]. Available: https://www.researchgate.net/publication/393802734_HUMAN-IN-THE-LOOP_SYSTEMS_FOR_ETHICAL_AI.