Computer Networks 247 (2024) 110451

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A novel method for solving the multi-commodity flow problem on evolving networks

Huilong Fan [a], Chongxiang Sun [a], Jun Long [b], Shangpeng Wang [a,*], Fei Zeng [a]

[a] School of Computer Science and Engineering, Central South University, Changsha, 410075, China
[b] The Big Data Institute, Central South University, Changsha, 410075, China

ARTICLE INFO

Keywords:
Multi-commodity flow
Evolving networks
Minimum cost
Dynamic graph

ABSTRACT

The minimum cost multi-commodity flow (MMCF) problem on evolving networks is the latest challenge to solving the min-cost network flow problem of multi-commodity flowing from multi-source and multi-sink on an evolving network. Previous researches primarily focus on the solution of MMCF problem within static networks. However, researchers ignored the MMCF problem within networks where the structure dynamically changes over time. Evolving networks have a dynamically changing and unpredictable network structure. The existing methods must recalculate when the network structure changes, leading to high computational complexity. This manuscript proposes a dynamic graph computing model for adaptive dynamic path selection and optimization. First, the model selects the path with the least cost for each commodity, assuming the route can transport a specified number of such commodities to find the optimal path faster. Second, we design path selection and flow update strategies for each node in the graph to adapt to changes in the graph structure. At the same time, we can prevent multiple commodity conflicts from exceeding the marginal capacity limit. Finally, the optimal path of each commodity is dynamically pruned based on the original optical path to obtain the tree structure and determine the absolute path to meet the specified transportation quantity of each commodity. Performance analysis results show that our method can efficiently cope with the dynamic changes of the network structure, which provides ideas for solving the multi-commodity network flow problem in dynamic networks.

1. Introduction

The minimum cost multi-commodity flow problem on evolving network (MMCFDN) [1–4] has recently become active essential research [5]. MMCFDN has an extensive range of applications, such as satellite networks [6,7], sensor networks [8–10], flow networks [11], social networks, etc [12]. The minimum-cost multi-commodity flow problem is a generalization of the minimum-cost flow problem. It refers to the fact that multiple commodities flow from multiple designated sources to sinks at a minimum cost and meet the arc-related capacity constraints and network flow conservation. Network routing, path planning, and network design problems can all be modeled as MMCFDN [5]. However, with the explosive growth of data, people have put forward higher requirements for the timeliness of these systems. In spatial information networks, satellite payload resources are more precious, and inter-satellite data transmission and path optimization usually require faster response time and lower transmission costs. In transportation networks, people need to obtain route planning routes quickly with the least transportation costs. Sensor networks need to calculate the least cost-critical path in the network faster. In other words, saving costs and obtaining planning results more quickly are problems that need to be solved urgently. Therefore, the minimum cost multi-commodity flow problem on evolving networks has attracted significant attention in academic and industrial circles.

The minimum cost multi-commodity flow problem on evolving networks has been extensively studied over time. MMCFDN is an NP-complete [13] mathematical programming problem [14], especially when the quantity of commodities is large, it will become even more catastrophic. Fortunately, the problem can theoretically be solved within a strongly polynomial time [15]. The solution to this problem can be roughly divided into three categories, namely basis partitioning method, resource-directive method, and price-directive method [2]. The basis partitioning method divides the basis by the inverse method of the basis to achieve more comprehensive and more use of the network structure characteristics, thereby making the basis inversion more efficient [16,17]. The resource-directive method finds the most reasonable commodity capacity allocation based on resources [18,19]. The price-directive method calculates the optimal penalty value for violating the bundling constraint, then uses the primal–dual method

* Corresponding author.
E-mail address: 204701019@csu.edu.cn (S. Wang).

https://doi.org/10.1016/j.comnet.2024.110451
Received 15 December 2023; Received in revised form 1 April 2024; Accepted 19 April 2024
Available online 22 April 2024

to improve the dual and maintain the complementary slack condition [16]. However, although these methods have satisfactory results in solving the MMCFDN problem, they cannot handle the dynamic change characteristics of the network structure in the evolving networks. These methods need to recalculate the minimum cost for each network update. In addition, the amount of calculation faced is enormous. Therefore, the abovementioned methods cannot be directly applied to solve the minimum cost problem on dynamic networks.

After that, some researchers paid attention to the network routing over time and proved that the problem could be solved in polynomial time using one min cost flow [20]. Since then, many researchers have conducted much research on time-varying networks. Some researchers have studied the time-expanded network method to solve the minimum cost flow problem of time-varying networks. They have given an approximate algorithm for the change of network flow over time [21–23]. Another commonly used solution method is to decompose the network based on the decomposition principle, then use the dynamic path flow model to solve the minimum cost problem [5].

However, there are three main challenges in solving the minimum cost multi-commodity flow problem in the dynamic network:

1. The minimum cost multi-commodity flow problem of a dynamic network is an NP-hard problem. How to calculate the optimal solution faster is a hot research problem that has always been a concern [5,24].
2. Solving the minimum cost multi-commodity flow problem of a large-scale dynamic network is catastrophic [2,5].
3. The situation where the demand for commodities changes dynamically and the change time of the network structure is discrete is almost ignored.

In a large-scale network, the new network formed by the edges involved in the flow distribution may be only a tiny network, resulting in a few edges related to problem-solving. As we all know, human beings can predict what will happen or actions, which allows us to predict the general progress of things in advance and help our following actions. If the computer flow distribution system can have the “prejudgment” ability to predict the edges relevant to the problem-solving in advance, then it can only focus on calculating a small range of edges, which can reduce a lot of unnecessary calculation work. So, why not add “prejudgment” capability to the computer flow distribution system to reduce the scope of calculation significantly? The current minimum cost multi-commodity flow problem (MMCF) solving methods for static networks and time-varying networks are only based on the idea of solving the problem from the perspective of mathematical modeling or decomposition, and the algorithm still needs to do a lot of useless calculations. Quickly calculating the multi-target network flow of a dynamic network has been a research hotspot in recent years. Therefore, in this manuscript, in response to the third challenge mentioned above, we focus on the method of adaptively calculating locally highly correlated data to solve the computational disaster problem caused by the huge network scale. By studying forecasting strategies, we can solve the above three challenges.

This manuscript proposes a model with the attention mechanism that can autonomously find and update the optimal path for dynamically solving the multi-commodity minimum cost flow problem of a dynamic spatiotemporal network (PAFUDN). PAFUDN uses the cost of edges as the weight from the original network to find the shortest paths from sources to sinks to form the core network rather than directly solving the original problem. Meanwhile, we establish an attention mechanism and pay attention to and calculate the nodes and edges related to the solution target to calculate the core network of the original network. After that, the model only needs to confirm whether the core network composed of the shortest path can withstand the circulation of all commodity demands, thus saving a lot of unnecessary calculations. If the core network can meet all commodity needs, we only need to calculate the core network flow distribution. On the contrary, if we cannot meet all commodity needs, we need to borrow a few edges from the original network and expand the total capacity of the core network to update the existing core network. Deleting and inserting edges in the dynamic network only needs to update the core network. Therefore, the model can significantly reduce the amount of calculation to solve the original problem and effectively improve the calculation efficiency and memory space occupation.

Our contributions can be briefly summarized as follows:

1. We propose a PAFUDN model that can predict the core network of the original network to solve the MCNF problem. PAFUDN model includes core network composition, flow distribution, and edge and core network dynamic maintenance update methods.
2. We have escaped the confinement of mathematical modeling ideas for solving NP-hard problems. We combine the predictive ability to transform the source network into a streamlined core network, which can avoid a large number of unnecessary calculations and achieve the purpose of solving the problem quickly.
3. We have solved the flow distribution problem of the dynamic update of commodity demand and the dynamic deletion and insertions of edges in the dynamic network with the passage of discrete time.
4. We empirically show that the proposed PAFUDN model has better performance and is better than the existing baseline method.

2. Related work

Most research has focused on solving the multi-commodity network flow (MCNF) problem of static networks in the past few decades, while the MMCF problem on evolving networks has little attention.

For static networks, Researches mainly focus on the three classic problems of the maximum MCNF problem, the max-concurrent flow problem, and the minimum cost multi-commodity Flow (MMCF) problem [25]. We focus on the investigation of the MMCF problem. Several researchers use basis partition or factorization methods to simplify the special network structure and solve the problem faster and more efficiently. However, these methods are no longer sufficient to deal with large-scale cases as the network scale increases. Some papers propose parallel computing techniques or large-scale nonlinear program algorithms to address the problems [2,26,27]. Meanwhile, some researchers proposed the fuzzy concept of network flow cost and designed algorithms such as fuzzy number comparison and fuzzy shortest path to improve the algorithm’s calculation efficiency more effectively, [28,29]. In recent years, several works of literature have focused on using the MMCF model to solve the problems in different fields and different industries and have made improvements to the MMCF method for specific problems in specific fields [30,31]. Some literature attempts to optimize the MMCF problem in software-defined networks by using greedy heuristics or k-splittable methods [32–34]. There are also researchers and scholars who use both CPLEX Optimizer [35,36] and Interior Point Method Solver (IPM) [37,38] to solve the MMCF problem. Early research mainly focused on the MMCF problem of static network flow, but these methods have passive performance when applied to dynamic networks.

Researchers discuss the maximum flow multi-commodity problem for time-varying networks [39], which slove the max flow from the source to the sink in the given time horizon T. Since then, it has attracted a large number of researchers to study time-varying networks. The method [40] mainly studied the method for solving the fastest path and proposed a polynomial algorithm. The method [41] studied the minimum flow problem in time-varying networks where the edges change with time. The method [42] proposed two methods to solve the time-varying network problem. The first one is an improved method that was inspired by Ford and Fulkerson and proposed. The

second method is to solve any intensive solution in polynomial time through appropriate discretization time. The method [43] mainly studies the shortest path between a specific node and all other nodes. The method [44] enhances the research status of time-varying networks by studying network flow theory in static networks as time goes by. When the capacity and cost of the edge in a dynamic network change dynamically, the distribution of flow is much more complicated than in a static network. In particular, it is an NP-hard problem to solve the minimum cost from the source node to the sink node of the commodity flow over time, even for an elementary series or parallel network [45]. Traditionally, the time extension method is the most commonly used method for time-varying networks. Methods [42,45,46] propose a condensed variant of time expansion and an approximate algorithm for time-varying network flow distribution. Assuming that all edges in the time-varying network have the same transmission time, the flow distribution problem of the time-based time-varying network is still polynomial under this condition. In order to solve this problem, [47] solved it by generating a set of copies of each discrete-time network, but the fatal disadvantage of this method is that it cannot handle a huge time-varying network.

The security implications of research on dynamic network algorithms cannot be ignored, and in the field of cyber security, Bisheh-Niasar et al. [48] investigated FPGA-based EdDSA. Wan et al. [49] investigated AI-enhanced TLS crypto processors to accelerate cryptographic computation. Lee et al. [50] contributed to the hardware/software co-design and GPU acceleration of post-quantum cryptography. Taken together, these efforts emphasize the need for cryptographic agility. Furthermore, research on fault-based attacks by Guo et al. [51] and Kaur et al. [52] and automated frameworks by Roy et al. [53] highlight the importance of adaptive cryptographic measures against complex threats in dynamic networks.

In summary, the static network multi-commodity minimum cost flow solution method is used to solve many problems in the dynamic network, and it cannot meet the actual needs in terms of computational efficiency. It is usually powerless in the face of large dynamic networks. The solution method of the time-varying network does not confirm that the demand for commodities changes dynamically with time, the deletion and the addition of edges in the network changes with the passage of discrete-time. In addition, the solution method of the MMCFDN problem cannot effectively deal with the large-scale network, and the method is usually incapable of dealing with the large-scale dynamic network. Most of the above methods deal with NP-hard problems from mathematical modeling and have not constructed an intelligent and efficient model to solve the multi-commodity minimum cost flow problem of large-scale dynamic space–time networks. Therefore, in response to these problems, this manuscript proposes an intelligent and efficient calculation model based on the attention mechanism to solve problems quickly and efficiently.

3. Problem formulation

Definition 1 (*The Minimum Cost Multi-commodity Flow in Dynamic Network Problem (MMCFDN)*)**.** The MMCFDN addresses the optimization of multi-commodity flows within networks that undergo temporal changes. These dynamic networks are characterized by variability in network topology, including fluctuations in edge capacities and costs, as well as variations in commodity demands over time. The objective is to determine the most cost-effective strategy for routing various commodities from their respective origins to destinations, adapting to network dynamics.

3.1. Mathematical model

We model the network as a directed graph $G(V, E)$, where each edge $(u, v) \in E$ is associated with a capacity $cap(u, v)$. Let $K = \{(s_i, t_i, d_i)|i = 1, \ldots, n\}$ denote the set of commodities, where each commodity K_i originates from source vertex s_i, targets sink vertex t_i, and has a demand of d_i. The flow of commodity K_i through edge (u, v) is represented by $f_i(u, v)$, with $c(u, v)$ signifying the unit cost for transporting on this edge. The challenge lies in minimizing the total transportation cost across all commodities while satisfying the following constraints:

Capacity limitation:

$$\sum_{i=1}^{k} f_i^t(u, v) \le cap^t(u, v), \tag{1}$$

Flow conservation:

$$\sum_{w \in V} f_i^t(u, w) = 0, \quad \text{for } u \ne s_i, \sigma_i, \tag{2}$$

Demand satisfaction:

$$\sum_{w \in V} f_i^t(s_i, w) = d_i^t \Leftrightarrow \sum_{w \in V} f_i^t(w, \sigma_i) = d_i^t, \tag{3}$$

Thus, the minimum cost multi-commodity flow in dynamic network problem (MMCFDN) can be formulated as follows:

$$\min \sum_{t=0}^{T} \sum_{(u,v) \in E} \left(c_{(u,v)} \sum_{i=1}^{k} f_{(u,v)}^{i}(t) \right), \forall \tau = \{0, 1, \ldots, T\} \tag{4}$$

We can observe that the MMCF problem is mainly solved by minimizing network costs under the premise of meeting all commodities requirements. For the minimum cost multi-commodity flow problem in the dynamic network, we need to assume that the cost function defined on the edge of the graph is non-linear and that the function depends on time and flow. In addition, we suppose that the demand function also depends on time.

The MMCFDN problem is particularly important in the practical application scenario of satellite networks. In this context, the nodes of the network correspond to the satellites, the edges of the network represent the communication links between the satellites, and the multiple data types (e.g., remotely sensed image data, command data, etc.) with changing demands are equivalent to commodities with different priorities or weights. The continuous dynamic changes in satellite positions lead to fluctuations in link capacity and communication costs, making it a crucial task to determine the optimal data transmission paths at different points in time to minimize the total cost. The optimization problem in such dynamic network environments not only needs to take into account the most efficient use of various resources, but also to cope with the constant changes in network state and demand. By accurately modeling and solving this problem, the data transmission efficiency and performance of satellite networks can be significantly improved to ensure that important data can be transmitted efficiently and reliably among different locations.

4. Methodology

The MMCF problem is regarded as NP-complete. The more intricate problem we face is how to solve the MMCF problem in a dynamic network. The main problem of MMCF is to design the paths of the commodities on the entire network to minimize the cost. Therefore, we transform the problem of solving the minimum cost of multi-commodities network flow in a dynamic network into solving the problem of maximum flow distribution of commodities on the core path and the problem of dynamic update of the core path. When we solve the multi-object network flow problem, the computational efficiency and time consumption are tremendous when the objects to be processed are all the nodes and edges in the network. In particular, we must deal with the catastrophic data latitude when encountering large networks. Therefore, we must transform the overall network flow problem into a partial one. First, we find multiple shortest paths from multi-source to multi-sink to form the core network. Meanwhile, the direction of the core network flow will be changed. Then, we need to solve the

maximum flow in the core network, which is the maximum number of commodities circulating in the core network. After that, we can determine the circulation path of partial commodities, but how can to deal with the remaining commodities that have not been circulated in the core network?

To deal with this problem, we propose a network adaptive update method in which the brief idea can be observed in Fig. 1. The concept can be articulated more precisely by illustrating that in a satellite network architecture, and the backbone comprises high-frequency communication satellites interconnected by fluid, dynamic links. These nodes and links represent the satellites and their communication pathways, respectively. The network's core is perpetually fine-tuned in response to real-time satellite positional updates and alterations in the link statuses—this includes integrating new links and removing defective ones. Diverse data types, such as satellite-captured imagery and command signals, are assigned varying levels of importance, influencing their prioritization as they traverse these evolving connections. The system is meticulously designed to continuously monitor and adapt to these fluctuations, ensuring efficient and reliable data exchange across satellites. By navigating through the most effective pathways, the network strives to fulfill the optimal objective of minimizing the cost associated with the multi-commodity flow, thereby enhancing overall operational efficiency.

Based on the above solution idea, we can re-express the original problem as follows:

$$\sum_{t=0}^{T}\left(\max \sum_{k\in K'}\sum_{(i,j)\in E'} c^k_{(i,j)}(t) f^k_{(i,j)}(t) + \sum_{k\in K''}\sum_{(u,v)\in E''} c^k_{(u,v)}(t) f^k_{(u,v)}(t)\right) \tag{5}$$

$$s.t.$$

$$\begin{aligned}&\sum_{\delta\in V'} f^k_{(s_i,\delta)}(t) + \sum_{\delta\in V''} f^k_{(s_i,\delta)}(t) = d_i\\ &\Leftrightarrow \sum_{\delta\in V'} f^k_{(\delta,\sigma_i)}(t) + \sum_{\delta\in V''} f^k_{(\delta,\sigma_i)}(t) = d_i,\end{aligned} \tag{6}$$

$$\sum_{l=1}^{m} f^l_{(p,q)}(t) \le cap(p,q), \quad \forall (p,q)\in E, \tag{7}$$

$$\sum_{\delta\in V'} f^k_{(i,\delta)}(t) = 0, \quad \forall i \ne s_k, \sigma_k, \tag{8}$$

$$\sum_{\vartheta\in V''} f^k_{(u,\vartheta)}(t) = 0, \quad \forall u \ne s_k, \sigma_k, \tag{9}$$

$$G(t) \ne G(t+1), \; c^k_{(i,j)}(t) = c^k_{(i,j)}(t+1) \tag{10}$$

$$\forall \tau = \{0,1,\ldots,T\}, \; K'\subseteq K, K''\subseteq K, E'\subseteq E, E''\subsetneq E, \forall t\in\tau \tag{11}$$

where Eq. (5) denotes the min-cost calculation method in a certain period on the network G. This method divides the network G into two parts to calculate all costs, which are the costs on the subgraph $G'(V', E')$ composed of the shortest path and the costs incurred by a small number of other paths. K' denotes the set of commodities flowing through the graph G', K'' denotes the remaining commodities that cannot be circulated in the network G'. Eqs. (6) and (7) denote the Capacity limit and demand satisfaction of time moments in dynamic network G. Eqs. (8) and (9) denote the flow conservation. Eq. (10) shows that the network G will change over time. Eq. (11) denotes the time is discrete and bounded by T.

Therefore, we may ignore some constraints and simplify the problem to solve the complex problem of minimum cost in a dynamic network. Then, we gradually restore and solve the original problem based on a simple problem.

For the convenience of explanation, we introduce a case to illustrate the solution process of MMCF. Fig. 2 shows the initial state of G. We assume that two commodities k_1 and k_2 that need to be transmitted in the network. The blue nodes 1 and 2 are the source of the commodities k_1 and k_2, and the yellow nodes are the sinks of all commodities. According to the constraint condition (2) of the MMCF problem, we set the flow flowing into the source nodes should satisfy constraint $f_i(m,n) \ge 0,\ m \in s_i,\ n \ne t_i$. Meanwhile, the flow of the sink nodes should satisfy constraint $f_i(r,n) \le 0,\ n \in t_i,\ r \ne s_i$. We denote the value on the edge as follows.

$$E^W_{(u,v)} = \left(c_{(u,v)}, cap_{(u,v)}\right), \tag{12}$$

In Eq. (5), $c(u,v)$ denotes the unit cost collection of different commodities on edge (u,v), it denotes as follows

$$c(u,v) = \left(c_{k_1}, c_{k_2}, \ldots, c_{k_i}\right), \tag{13}$$

As shown in Fig. 2, we can observe $(2,1,6)$ on the edge $(2,4)$, where '2' means the unit cost of commodity k_1, '1' means the unit cost of commodity k_2, and '6' means the capacity of edge $(2,4)$. As shown in Table 1, we give a simple sample commodity and predefine values on the network G, which includes two commodities, two source nodes, three sink nodes, and the number of commodities demand on each node.

Table 1
Sample commodities.

Commodities K	Sources	Demand
1	1	2
1	2	4
1	7	−4
1	8	−1
1	9	−1
2	1	3
2	2	3
2	7	−1
2	8	−4
2	9	−1

For further illustration, we can materialize in a satellite network, which is embodied in the dynamic graph model shown in Fig. 2. Here, the blue nodes indicate the data commodities departure point such as satellites responsible for data acquisition. In contrast, the yellow nodes indicate the destinations, such as data processing centers or other satellites. Each edge is labeled with the cost of data transmission from one satellite to another and the communication capacity of the link. These values fluctuate due to satellite movement and changes in the communication environment.

Next, we use this example to explain the process of our model.

4.1. Core-network transform and flow assignment

In this section, we mainly introduce how to transform the MMCFDN problem into a simple problem. Before that, we need to give some assumptions, as shown in Assumption 1. Based on these assumptions, we propose a hypothesis that ignores the details of the original problem. As stated in hypothesis 1, we take the cost of the commodities on each edge as the only basis for finding the shortest path and ignore the capacity constraint of Eq. (1) in the process of finding the shortest path. Then, we only need to find the shortest path to the sinks from different sources. Based on the above ideas, these shortest paths can form a core network. Definition 2 shows the detailed definition of the core network. It contains all the shortest paths from multi-source to multi-sinks, and the attributes of the edges are the same as the edges of the network G.

Assumption 1. We assume that all commodities k are only allowed to pass through the shortest paths R and do not conedger commodities that cannot be transmitted on the shortest path. Meanwhile, we do not conedger the specific quantity d_i of the commodities s_i transmitted on the paths R and only focus on the total flow allocated on each edge of the network G'. Then, the MMCF problem can be transformed into solving the maximum flow distribution on the network G'.

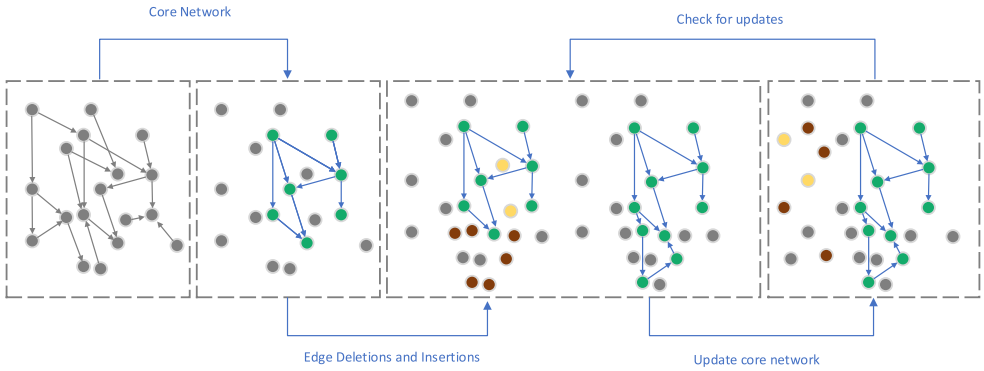


Fig. 1. Schematic graph of the overall idea.

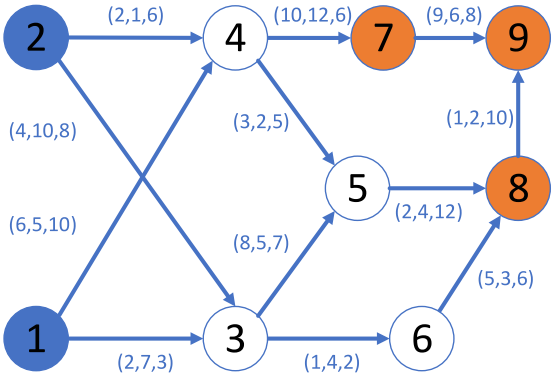


Fig. 2. The initial network state in which the blue and yellow nodes respectively represent the source and sink of the commodities.

Definition 2. If there exists a shortest path r_i from the source s_i to the sink σ_i and the set of all shortest paths is denoted as R on the network G, then all the edges, labels and vertices in R can be defined as a core-network $G'(V', E')$.

Theorem 1. *Given a multi-source and multi-sink network G and its core network G', where G' is composed of all the shortest paths from different sources and sinks in G. If there is a flow f flowing on G, the minimum cost is $T(G, f)$, and the minimum cost flowing G' is $T(G', f)$, then $T(G', f) = T(G, f)$.*

Proof. Assume that $T(G', f)$ is not the minimum cost by the flow f flowing through G, that is, $T(G', f) \neq T(G, f)$.

Set the cost on edge e in graph G is $c(e)$, the cost on edge e' in graph G' is $c(e')$. All paths of flow f flowing through G are $P(G, f) = \{p_1, p_2, \ldots, p_i\}$, and the set of edges contained in path P is $E(P)$. All paths of flow f flowing through G') are $P'(G', f) = \{p'_1, p'_2, \ldots, p'_j\}$, the edges of path P' is $E(P')$.

$\because$ From the meaning of $T(G', f) \Rightarrow T(G, f) = min(P(G, f) \times c(e)) = P_i$, where $e \in E(P)$. Similarly, $T(G', f) = min(P'(G', f) * c(e')) = P'_j$, where $e' \in E(P')$. Also $\because P_i \in G$, $P'_j \in G'$ and $G' \in G$, G has the shortest path S, G' has the shortest path S',

$\therefore P_i = S, P'_j = S'$,

$\because G' = G' \cap G \neq \emptyset$, $S \in G$, $S' \in G'$,

$\therefore S = S'$,

$\therefore P_i = P'_j$,

$\therefore T(G', f) = T(G, f)$.

Also $\because T(G', f) \neq T(G, f)$, produce conflict.

So the hypothesis does not hold. □

In order to generate the core network G', we designed the Alg. 1 based on the Theorem 1 to realize the pruning of all useless edges except the shortest path in the original graph and the inversion of flow on the critical paths.

As shown in Alg. 1, we use the $SPFA$ [54] algorithm to find all the shortest paths $R = r_1, r_2, \ldots, r_p$ (line 2) in the network G. The simple calculation process is shown in Fig. 3, in the sample network G, we take the path $cap(u, v)$ as the weight. Mark nodes '1' and '2' as the source nodes and nodes '7', '8', and '9' as the sink nodes, and then find all the shortest paths in network G that satisfy the above constraints. We can see the details of the shortest path search results in Table 2. Then, perform pruning operations on all edges E'' and node v'' that are not in R to obtain all edges E' and node V'in R, because r_i and r_j may have overlapping edges and nodes. As shown in Fig. 4, Fig. 4 (c) shows the shortest path set of two source nodes and Fig. 4(d) shows the core network obtained after the pruning operation. At this stage, we do not conedger the cost of each commodity k_i in the core network G'. Therefore, E' and V' need to be deduplicated (lines 4–8). Then, a new network G'(lines 9–10) is formed at this time. Finally, obtain the adjacency matrix $A = adj(G')$ of the network G', perform the transposition operation $A = A^T$ on the matrix and convert the direction of the edge in the directed graph G' to the opposite direction (line 11). At a result, as shown in Fig. 4(e), we can obtain the turned core network G'.

Algorithm 1: Network Pruning and Reverse Flow Transform (NPRF)

Input: Directed graph $G(V, E, S, T, c, cap)$,
Output: Adjacency matrix of a new network

```
1  G' = null;
2  for each s ∈ S do
3      SPFA(s,adj,dist,path);
4      for each t ∈ T do
5          V'=pathNode(s,t,path);
6          E'=pathEdge(s,t,path);
7      V'=Unique(V');
8      E'=Unique(E');
9  G'.vertex = V';
10 G'.edges = E';
11 G'.adj = transpose(adj(G'));
12 return G';
```

When using all the shortest paths to build a core network G', we hope to meet all the commodity requirements for sink nodes. Therefore,

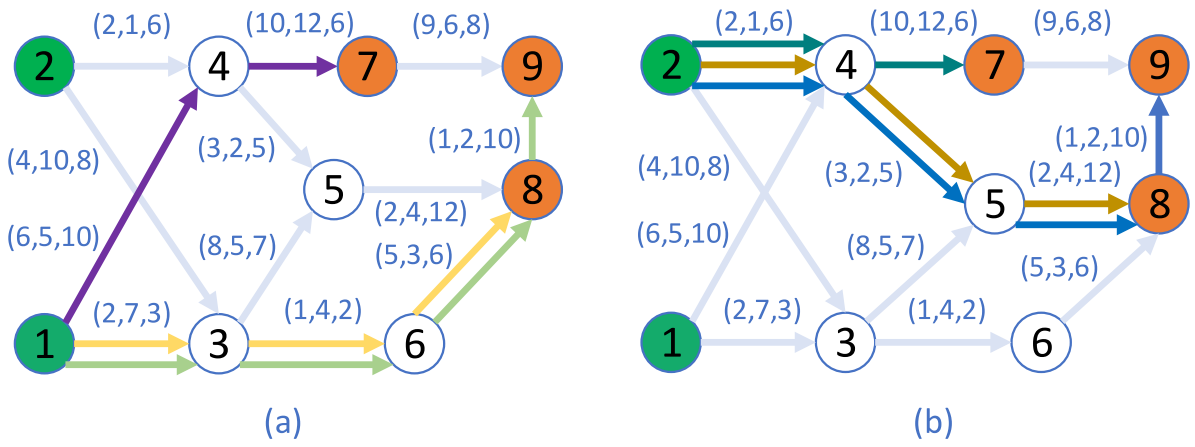


Fig. 3. Subfigure (a) shows the shortest path from node 1 to nodes 7, 8, and 9, Subfigure (b) shows the shortest path from node 2 to nodes 7, 8, and 9.

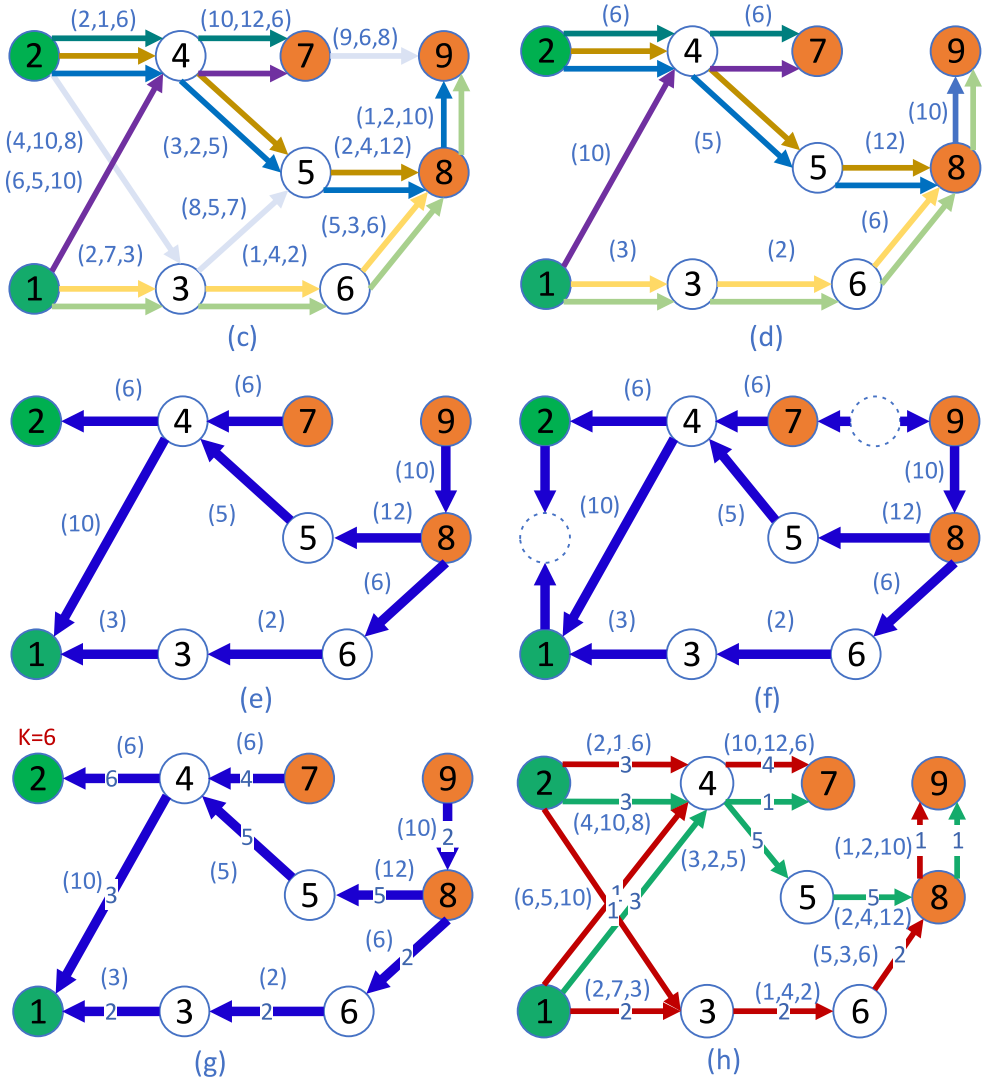


Fig. 4. Subfigure (c) shows all the shortest paths from all source nodes to sink nodes. Subfigure (d) shows the core network G' of network G. Subfigure (e) shows the core network G' after turning. Subfigure (f) shows the core network with the addition of super source and sink nodes. Subfigure (g) shows the maximum flow on the edge of the network G'. Subfigure (h) shows the flow distribution of different commodities under the restriction of the maximum flow on the edge.

Table 2
Sample commodities.

Sources	Sinks	Shortest paths
1	7	1⟶ 4 ⟶ 7
1	8	1⟶ 3 ⟶ 6 ⟶ 8
1	9	1⟶ 3 ⟶ 6 ⟶ 8 ⟶ 9
2	7	2⟶ 4 ⟶ 5 ⟶ 7
2	8	2⟶ 4 ⟶ 5 ⟶ 8
2	9	2⟶ 4 ⟶ 5 ⟶ 8 ⟶ 9

we reverse the direction of the edges of all the shortest paths R. At the same time, the sink node becomes the source node that starts from the new source node s' and flows to the new sink node. As shown in the hypothesis 1, we assume that the core network can meet all the commodity needs on the sink node. Based on this assumption, we first solve the maximum flow that can be accommodated on the network G'. If the demand is less than the maximum flow, we can easily solve it using existing methods. Conversely, if the demand is greater than the maximum flow, we will focus on solving the shortest path of the part of the flow that exceeds the network's maximum capacity to solve the original problem.

To solve the maximum flow of the core network problem, we design the maximum flow distribution in core network algorithm (MFD). As shown in Alg. 2, we add two new nodes, which link all source nodes and sinks (lines 1–7). Then, we add the two nodes to G' (lines 8–9). At this node, we can add the two dotted nodes shown in Fig. 4(f). At this time, we convert the problem of solving the maximum flow of multiple sources and multiple sinks into a single-source maximum flow problem. To solve this problem, we use the Ford-Fulkerson maximum flow solving algorithm to find the maximum flow that can be circulated on the core network G' (lines 10–11). The maximum flow calculation result is shown in Fig. 4(g).

To explain Alg. 1 and Alg. 2 better: to apply to the satellite network by transmitting data from the core network to the satellite network. Fig. 3 and Fig. 4 illustrate the process from multiple data collecting satellites to a receiving point (e.g. a data processing center or another satellite). This emphasizes that the most efficient paths between data sources and destinations are identified and established, thus structuring the central network in the satellite framework.

Fig. 3(a) and Fig. 3(b) delineate the shortest routes from various source nodes to their destination nodes. These routes are meticulously chosen to minimize transmission costs, deliberately overlooking the capacity limitations of the pathways. Subsequently, Fig. 4 showcases the emergence of the shortest paths ensemble, with Fig. 4(c) illustrating all source nodes converging towards the sink and Fig. 4(d) displaying the core satellite network post-pruning. This pruning exercise isolates the essential pathways to streamline the satellite network model.Fig. 4(e) and Fig. 4(f) depict the core network's transformation through reorientation and the incorporation of super-source and sink nodes, respectively. This adjustment simplifies the complex multi-source and multi-sink maximum flow challenge into a tractable single-source maximum flow problem. Furthermore, Fig. 4(g) and Fig. 4(h) highlight the maximum allowable flow distribution within the core network and the allocation of various commodities under this maximum flow constraint.

This series of strategic maneuvers elucidates a comprehensive data flow optimization methodology within actual satellite networks. It involves determining and refining the shortest transmission paths in alignment with actual traffic allocations and network capacities. The numerical values represented in the subfigures denote the distribution of diverse data types across the links, encompassing remote sensing imagery, command instructions, communication signals, and navigation information. Such dynamic adjustments guarantee that the satellite network is adept at managing assorted data demands, thereby ensuring the efficacy and security of data transmission.

Algorithm 2: Maximum flow distribution in core-network (MFD)

Input: Directed graph $G^{'}(V^{'}, E^{'}, S^{'}, T^{'}, c, cap)$,
Output: An array F stores the flow on each edge
1 $sourceNodes = G'$.getSourceNodes();
2 $source$ = new Node();
3 $source$.outDegreeNode(sourceNodes);
4 $sinkNodes = G'$.getSinkNodes();
5 $sink$ = new Node();
6 $source$.outDegreeNode($sourceNodes$);
7 $sink$.inDegreeNode($sinkNodes$);
8 G'=G'.add($source$);
9 G'=G'.add($sink$);
10 F=FordFulkerson(G');
11 return F;

Definition 3. Given a core network G', if the maximum flow f in the core network is known, then the maximum flow allowed to circulate on the edge of the network G' is defined as the adequate flow, denoted as $f(u, v)$.

4.2. Core-path multi-commodity flow computation

However, when we obtain the maximum flow in the core network, each edge is allocated a certain amount of effective flow. As shown in Definition 2, we denoted the effective flow as $f(u, v)$. In the process of allocating flow to each edge of the network G', if all the flow flowing through the network generates the highest cost, then the cost incurred in the network G is the smallest. If the current core network G' is not enough to meet the needs of all commodities, then we need to update the current core network G' so that the network meets the needs of all commodities. On the contrary, if the current core network G' can meet the needs of all commodities, then the current core network flow distribution is recorded as the minimum cost distribution plan. Therefore, how to allocate the flow of each commodity k_i under the condition of the effective capacity $f(u, v)$ of each $edge(u, v)$ is the next problem we need to solve.

As shown in Fig. 5, we decompose the problem of solving the commodity flow of each edge on the core network into three parts, including the flow determination on the out-degree edge of the source node, the flow determination on the intermediate node's edge, and the flow determination on the in-degree edge of the sink.

We assume that the total demand for commodities is D, the maximum flow of the core network is M. If $D \leq M$, the flow on the out-degree edge of the source node is the demand of the source node, denoted as follows,

$$\begin{aligned} &f\left(N_i(in)\right) = D\left(K_s\right) \\ &s.t.\ \forall N_i \notin (S,T),\ K_s \subseteq K, \end{aligned} \tag{14}$$

which N_i represents the neighbor node connected to the source node and K_s represents the commodities on the in-degree edge flowing through the node N_i, which $f\left(N_i(in)\right)$ denotes the flow on the out-degree edge or in-degree edge of node N_i. In addition, the flow on the in-degree edge of the sink and the total demands consumed by the sinks can be expressed as follows,

$$\begin{aligned} &f\left(T_i(in)\right) = D\left(K_s\right) \\ &s.t.\ T_i \in T,\ K_s \subseteq K, \end{aligned} \tag{15}$$

where T_i denotes the sink node, $T_i(in)$ denotes the in-degree flow of sink node and $f\left(T_i(in)\right)$ denotes the flow on the out-degree edge or in-degree edge of node N_i.

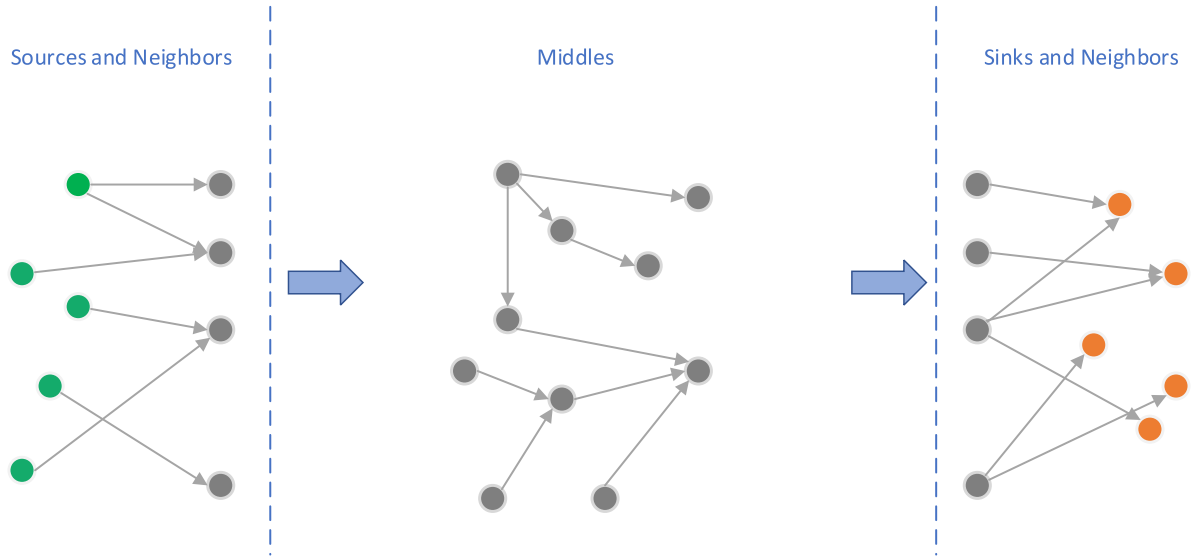


Fig. 5. We divide the core network into three parts: the source node with its neighbor nodes, intermediate nodes, and the sink node with its neighbor nodes. We solve the flow values of different commodities on the edges of the three parts under the maximum flow constraint of each edge.

If $D > M$, the flow from the sources to the neighbor nodes can be denotes as follows,

$$\begin{aligned} &f\left(N_i\left(in\right)\right) = X\left(K_s\right) \\ &s.t.\ N_i \notin (S,T),\ K_s \subseteq K,\ X\left(K_s\right) = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}, \\ &X\left(K_s\right) \times \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = f\left(u,v\right),\ u \in S, v \in N_i, \end{aligned} \tag{16}$$

which $X\left(K_s\right)$ denotes the flow from the source node to the neighbor node and the flow x_i from a source node to a neighbor node is the effective flow $f\left(u,v\right)$, where u belongs to the source node and v belongs to the neighbor node. Meanwhile, the demand for commodities at the meeting node is the sum of the values from the meeting node to the neighbor nodes, and the sum of the demand for commodities at all meeting nodes is M, this can be denoted as follows,

$$\begin{aligned} &f\left(T_i\left(in\right)\right) = X\left(K_s\right) \\ &s.t.\ T_i \in T,\ K_s \subseteq K,\ X\left(K_s\right) = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}, \\ &X\left(K_s\right) \times \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = f\left(u,v\right),\ u \in Nei, v \in T_i, \end{aligned} \tag{17}$$

where Nei denotes the neighbor nodes of T, T_i denotes the sink node. We can observe that the sum of all commodities flows into T. We can denotes as $\sum_{i=1}^{l} T_i\left(in\right) = M$. Next, solving the flow of different commodities on edge between the intermediate nodes is a crucial issue. From Theorem 1, we need to find the shortest paths to meet the minimum cost. Therefore, we define the problem as follows,

$$\begin{aligned} &R = \min\left(AN\left(X\right)\right) \\ &s.t. \\ &A = \begin{bmatrix} a_1 & a_2 & \cdots & a_q \end{bmatrix}, \\ &N\left(X\right) = \begin{bmatrix} n_1\left(X\right) \\ n_2\left(X\right) \\ \vdots \\ n_p\left(X\right) \end{bmatrix}, X = \begin{bmatrix} x_1\left(k\right) \\ x_2\left(k\right) \\ \vdots \\ x_q\left(k\right) \end{bmatrix}, \\ &\forall q \in N^*,\ \forall p \in N^*, \end{aligned} \tag{18}$$

where a_i represents the cost matrix generated by all in-degree edges or out-degree edges of a unit commodity flowing through node N_i, which a_i represents the cost matrix generated by all in-degree edges or out-degree edges of a unit commodity flowing through node N_i. In addition, $n_i\left(X\right)$ denotes the sum of flow on the in-degree edge or out-degree edges in the set of intermediate nodes. $x_i\left(X\right)$ denotes the flow matrix on the in-degree or out-degree edge of node N_i. Therefore, we can transfer the problem as follows,

$$\begin{aligned} &R = \min\left(AN\left(X\right)\right) \\ &\quad = \min \sum_{i=1}^{p} \sum_{j=1}^{q} a_{ij} n_i\left(x_{ij}\right) \\ &s.t. \\ &\quad \sum_{j=1}^{q} a_j n_i^{pre}\left(x_{ij}\right) = f\left(pre, n_i\right),\ pre \notin \left(S, S_{nei}, T, T_{nei}\right), \\ &\quad \sum_{i=1}^{p} a_i n_i\left(x\left(k_m\right)\right) = D_{n_i}\left(k_m\right), \\ &\quad \forall q \in N^*,\ \forall p \in N^*, \end{aligned} \tag{19}$$

where x_{ij} denotes the number of commodities on the nth in-degree edge or out-degree edge of the ith node, which a_{ij} denotes the cost incurred by the unit commodity in x_{ij}, where S_{nei} denotes the source node and the set of neighbor nodes of the source node. T_{nei} denotes the sink node and the set of neighbor nodes of the sink node. Pre only belongs to the set of intermediate nodes, not to S_{nei} and T_{nei}, where k_m denotes the maximum flow of a specific commodity on a certain edge.

In summary, we need to calculate the flow distribution on the edges of the three parts when calculating the commodities flow that can circulate on the core network. First, we need to calculate the flow between the source node S and its neighbor node S_{nei} which $f\left(N_i\left(in\right)\right)$ denotes the calculation result, then we calculate the edge flow between the meeting node T and its neighbor nodes T_{nei}. Finally, we calculate the flow between the intermediate nodes $f\left(T_i\left(in\right)\right)$ and the calculation of the flow R on the intermediate nodes. We assume that the demand for commodities on the meeting node can be satisfied, then the different commodities flow on edge between the meeting node and its neighbor nodes is the demand $D\left(K_s\right)$. The solution of S_{nei} $f\left(N_i\left(in\right)\right)$ and flow R is a typical linear programming problem.

In order to solve the minimum cost multi-commodity flow problem on the core network, we propose the minimum cost multi-commodity flow allocation on the core-network algorithm (MCFD). As shown in Alg. 3, we use the linear programming model to complete the calculation of flow distribution. We obtain the number of neighbor nodes of

the source node, the number of neighbor nodes of the sink node, and the number of intermediate nodes (lines 1–3). If the demand quantity is greater or less than the maximum flow quantity, then use the demand function to assign the maximum flow value to the edge between the source or intermediate node and their neighbor nodes (line 6), and use the linear programming function *linprog* to calculate the flow value of different commodities on each edge (line 7). Similarly, we calculate the maximum flow between the sink node or intermediate node and their neighbor nodes (lines 8–10). Finally, the demand is assigned to the edge formed by the sink node and its neighbor nodes as its flow.

For further explanation, we can continue to interpret Fig. 5 in terms of satellite networks, which are divided in a manner particularly suitable for satellite networks where nodes represent satellites and edges represent satellite links. The source node and neighbor nodes are depicted as the ground station initiating the data transmission and the neighboring satellites serving as entry points for the data flow. Intermediate nodes are satellites installed along the transmission path and play an important role in data transfer. Finally, the sink and neighboring nodes represent the final data receiving station and surrounding satellites that mark the end of the data journey.

Algorithm 3: Minimum cost multi-commodity flow allocation on the core-network (MCFD)

Input: Directed graph $G^{'}(V^{'}, E^{'}, S^{'}, T^{'}, c, cap)$, Maximum flow M, commodities demands D

Output: A dictionary of flow flows $dict$ through which commodities can flow on each edge

1 $sNeiNum$ = size(S_{nei});
2 $tNeiNum$ = size(T_{nei});
3 $midNeiNum$ = size(pre_{nei});
4 **if** $D \geq M$ *or* $D \leq M$ **then**
5 **for** *each* $i \in sNeiNumormidNeiNum$ **do**
6 $f\left(N_i\left(in\right)\right)$ = demand(M_i);
7 X_s = linprog(f_i,A_i,b_i,Aeq_i,beq_i);
8 **for** *each* $j \in tNeiNum$ **do**
9 T_j = demand(D_j);
10 X_t = linprog(f_j,A_j,b_j,Aeq_j,beq_j);
11 $dict$ = [X_s,X_{mid},X_t];
12 return $dict$;

We have found the shortest path of all source nodes. Meanwhile, we can transfer some commodities with minimal cost. Next, we will introduce how to select the other least-cost paths for the remaining commodities and correctly transfer them to meeting sinks.

5. Core-network dynamic update strategy

In a dynamic network, the edges in the network will be dynamically added and deleted. When a node is removed, all the out-degree and in-degree edges of the node need to be deleted. When a new node is added, the new edge that follows will also be connected to the network. Therefore, we can regard the change in the dynamic network as the dynamic increase and decrease of the edge between nodes. In addition, we need to conger that all commodity demand cannot be met when $D > M$. So, updating the current core network to meet the demands of all commodities and updating edges or nodes are critical issues.

5.1. Core-path expansion and flow update for demand update

In Section 4, we introduced that when the demand for commodities is greater than the maximum flow of the core network, there is a problem that some commodities cannot circulate in the network. Therefore, we upgrade the problem to a situation where commodity demand is variable, expressed as the following,

$$\begin{aligned} &D\left(t\right) \neq D\left(t+\delta\right) \\ &s.t.\ \forall t \in N,\ \forall \delta > 0, \end{aligned} \tag{20}$$

where $D\left(t\right)$ denotes the total demand for commodities at time t, $D\left(t+\delta\right)$ denotes the total demand for commodities at time $t+\delta$. We can see that commodities can have different demand at different times, and $D\left(t\right)$ may be greater or less than $D\left(t+\delta\right)$. When $D \leq M$, we have introduced the calculation method of multi-commodity demands on the core network. when $D > M$, how to allocate the excess commodity demand $D - M$ is an important problem.

Theorem 2. *Given a multi-source and multi-sink network G and its core network G', where G' is composed of all the shortest paths from different sources and sinks in G. At time t, the minimum cost of flow flowing through the network is $T(t)$. If the quantity of commodity demand flowing through the network G is updated at time $t+1$, and the set of new edges affected by the commodity update is E, the cost of flow change on E is $C(E)$, then $C(E)+T(t)=T(t+1)$*

Proof. Suppose the minimum cost $T(t+1) \neq C(E)+T(t)$ on the network G at time $t+1$

Set the edge set of G is $A(G)$, and the edge set of G' is $A(G')$,

$\because G' \in G$,

$\therefore G'' = C_G G'$, $C_G G'$ represents the complement of $G's$ subset of G

Suppose the cost change of all edges in the network G'' relative to time t at time $t+1$ is $C(A(G''))$, and the cost change of all edges in the network G' relative to time t is $C(A(G'))$.

$\because G = G' + G''$, and the change in the total cost of the network G at time $t+1$ relative to time t is $C(E)$ $\therefore C(E) = C(A(G')) + C(A(G''))$.

And $\because$ From the meaning of $E \Rightarrow E$ has three situations, namely $E \cap A(G'') = E, (E \cap A(G'')) \cup (E \cap A(G')) = E, E \cap A(G') = E$,

$\therefore$ when $E = E \cap A(G''), C(A(G')) = 0, C(A(G'')) \neq 0$, $C(E) = C(A(G'')), C(A(G'')) \in R^*$.

When $E = (E \cap A(G)) \cup (E \cap A(G')), C(A(G')) \neq 0, C(A(G'')) \neq 0$,
$C(E) = C(A(G')) + C(A(G'')), (C(A(G'')), C(A(G'))) \in R^*$.

When $E \cap A(G') = E, C(A(G'')) = 0, C(A(G')) \neq 0$,
$C(E) = C(A(G')), C(A(G')) \in R^*$.

In summary, when $C(E) > 0$ or $C(E) < 0$, $T(t+1) = T(t) + C(E)$,
So the hypothesis is not true, the proposition is proved □

In order to achieve the purpose of enabling the flow distribution plan to adapt to changes in the quantity of commodities demand and to update the flow distribution plan without recalculating, we creatively proposed a core-network flow adaptive allocation with changes in commodity demand algorithm(CFCD). CFCD improves the current situation of repeated calculations of flow distribution schemes caused by dynamic changes in commodity demand and solves the problem of a large amount of computing power and time consumed. As shown in Alg. 4, we calculate the difference between the demand and the maximum flow to get the amount of data to be allocated (line 5). Then, obtain the source node $PendSources$ (line 6) whose commodity demand in the core network is not 0. If the commodity demands on the source node are 0, it means the source node has been allocated. Then, we get the list of commodities on the $PendSources$ node (line 7). We find the out-degree edge of the PendingSourcesNodes node, which needs to satisfy the edge that is not on the core network G'but exists on the source network G, and obtain the unit commodity cost on edge (lines 8–9). In order to ensure that the total cost is the smallest, we prioritize selecting the edge with the smallest total cost for different commodities to join the core network (lines 10–12), finally updating the flow of each edge for the new core network.

5.2. Core-path dynamic update for edge deletions or insertions

When a node's out-degree and in-degree edges are removed, the node can be regarded as removed from the network G' to adapt to the dynamic update of the origin network G. The dynamic update of network G includes two methods, namely edge deletion, and edge addition. When the out-degree and in-degree edges of a node are all

Algorithm 4: Core-network flow adaptive allocate with changes in commodity demand (CFCD)

Input: Directed graph $G^{'}(V^{'},E^{'},S^{'},T^{'},c,cap)$, Directed graph $G(V,E,S,T,c,cap)$, Commodity demand D on the source node, Maximum flow M, Multi-commodity flow on the core network f

Output: A dictionary of flow flows $dict$ through which commodities can flow on each edge

```
1  sNeiNum = size(S_nei);
2  tNeiNum = size(T_nei);
3  midNeiNum = size(pre_nei);
4  if D ≥ M then
5      RemainDemand = D − M;
6      PendSources = getAllPS(S′);
7      K_p = getcomdies(PendSources);
8      edgesets = OutDegEgs(G,G′);
9      c_k = getEdgeCost(edgesets);
10     for each e ∈ edgesets do
11         cost(e) = K_p × c_k;
12         carray.append(cost(e));
13     for each i ∈ sNeiNumori ∈ midNeiNum do
14         f(N_i(in)) = demand(M_i);
15         X_s = linprog(f_i,A_i,b_i,Aeq_i,beq_i);
16 for each j ∈ tNeiNum do
17     T_j = demand(D_j);
18     X_t = linprog(f_j,A_j,b_j,Aeq_j,beq_j);
19 dict = [X_s,X_mid,X_t];
20 return dict;
```

removed, then the node can be regarded as removed from the network G. The different positions of the added edge and the deleted edge in the network G have different effects. If the edge update is directly related to the node in the core network G', then we need to confirm whether the edge impacts the core network. Suppose the edge update is indirectly related to the nodes in the core network G'. After that, we need to confirm whether the change can replace an edge in the original network G' when calculating the shortest path. As shown in Fig. 6, green nodes are nodes in the core network, Fig. 6(b) is a new example of edges in the core network; Fig. 6(c) is an example of deleting edges in the core network; Fig. 6(f) is an edge except for the core network A new example of; Fig. 6(g) shows an example of deletion of edges other than the core network.

Theorem 3. *Given a multi-source and multi-sink network G and its core network G', where G' is composed of all the shortest paths from different sources and sinks in G. At time t, the minimum cost of flow flowing through the network is $T(t)$. If a new edge is inserted into the network G, and the set of edges affected by the edge insert is E, the cost of flow change on E is $C(E)$, the minimum cost flow on the network after the insert operation is T', then $C(E)+T(t)=T'$*

Proof. Suppose that the minimum cost $T' \neq C(E)+T(t)$ after inserting a new edge on the network G Suppose the edge set of G is $A(G)$, and the edge set of G' is A(G'),

$\because G' \in G$,

$\therefore G'' = C_G G'$, $C_G G'$ represents the complement of $G's$ subset of G

Suppose that after the new edge is inserted, the cost change of all edges in the network G'' relative to time t is $C(A(G''))$, and the cost change of all edges in the network G' relative to time t is $C(A(G'))$.

$\because G = G' + G''$, and the change in the total cost of the network G at time $t+1$ relative to time t is $C(E)$ $\therefore C(E) = C(A(G')) + C(A(G''))$.

$\because$ If after the new edge is inserted, the cost of traffic f flowing through the network G has not changed, then $C(E)=0, T'=T(t)$,

$\therefore T' = T(t) + C(E)$, $C(E)=0$.

If the cost of traffic f flowing through network G changes after the new edge is inserted, then $C(E) > 0$ or $C(E) < 0$ And $\because$ There are two situations for the insertion of a new edge. The first is that the new edge is inserted into the core network G', and the second is that the new edge is inserted and is not allocated to the core network. Based on these two situations, there are three situations in which the range of traffic changes in the network are ⇒

$E \cap A(G'') = E$, $(E \cap A(G'')) \cup (E \cap A(G')) = E$, $E \cap A(G') = E$,

$\therefore$ when $E = E \cap A(G'')$, $C(A(G')) = 0, C(A(G'')) \neq 0$,

$C(E) = C(A(G''))$, $C(A(G'')) \in R^*$.

When $E = (E \cap A(G)) \cup (E \cap A(G'))$, $C(A(G')) \neq 0, C(A(G'')) \neq 0$,

$C(E) = C(A(G')) + C(A(G''))$, $(C(A(G'')), C(A(G'))) \in R^*$.

When $E \cap A(G') = E, C(A(G'')) = 0, C(A(G')) \neq 0$,

$C(E) = C(A(G'))$, $C(A(G')) \in R^*$, then T(t+1)=T(t)+C(E)

In summary, when $C(E) > 0 or C(E) < 0$, $T(t+1) = T(t) + C(E)$,

So the hypothesis is not true, the proposition is proved □

In order to achieve the purpose of quickly updating the network flow after the new edge insertion, we proposed the core-network flow adaptive allocation with an edge insertion algorithm (CFEI). CFEI is designed based on dynamically adjusting the flow conditions of neighbors to balance the flow of the entire network. In addition, If the new edge causes the core network to change, then we need to update the core network and its flow distribution. On the contrary, if there is no impact on the core network, the change can be ignored. The specific process is shown in Alg. 5. First, we judge whether an edge e is in the core network. If both nodes a and b belong to the core network, we compute the neighbor nodes of node b (lines 1–2) and calculate the cost of the neighbor nodes through a to b respectively. If the cost is less than before the edge is added and the edge capacity is greater or equal to the circulated flow on the original edge, we will delete the original edge and insert the new edge into the core network G'(lines 3–10). Then, update the maximum flow and flow distribution on the edge of the core network (lines 11–12). If the newly changed start nodes a and b do not belong to the core network. Then, the cost incurred by the neighbor node of b passing through a to b is reduced due to the increase of edge $e(a,b)$. It is necessary to recalculate the core network for the entire network G, calculate the maximum flow and flow distribution, and finally obtain the minimum cost Flow distribution results (lines 14–19).

Theorem 4. *Given a multi-source and multi-sink network G and its core network G', where G' is composed of all the shortest paths from different sources and sinks in G. At time t, the minimum cost of flow flowing through the network is $T(t)$. If an edge is removed on G, and the set of edges affected by the edge remove is E, the cost of flow change on E is $C(E)$, the minimum cost flow on the network after the remove operation is T', then $C(E)+T(t)=T'$.*

The proof process of Theorem 4 is similar to Theorem 3. Therefore, we omit the proof process here. Similar to the principle of adding edges, we designed an algorithm to update network flow online after edge deletion. As shown in Alg. 6, for the deletion process of edge $e(a,b)$ on the original network G,if the deleted edge e belongs to the core network, then we need to recalculate the core network (lines 1–3). If the new core network and the original core network are consistent, then the distribution of the minimum cost flow remains unchanged. If they are inconsistent, we need to recalculate the maximum flow and flow distribution results (lines 4–9) for the new core network.

To further illustrate, we apply the Dynamic Core Network Update Strategy to the satellite network, Fig. 6 shows how the core network adapts to dynamic changes in scenarios where external edges are added and removed. This adaptation is an essential part of the continuous evolution of the satellite network. It reflects scenarios that may occur in real satellite operations: addition of new satellites, removal of existing satellites, or changes in communication links. In particular, Fig. 6(a) shows the original state of the core network. At the same time, Fig. 6(b)

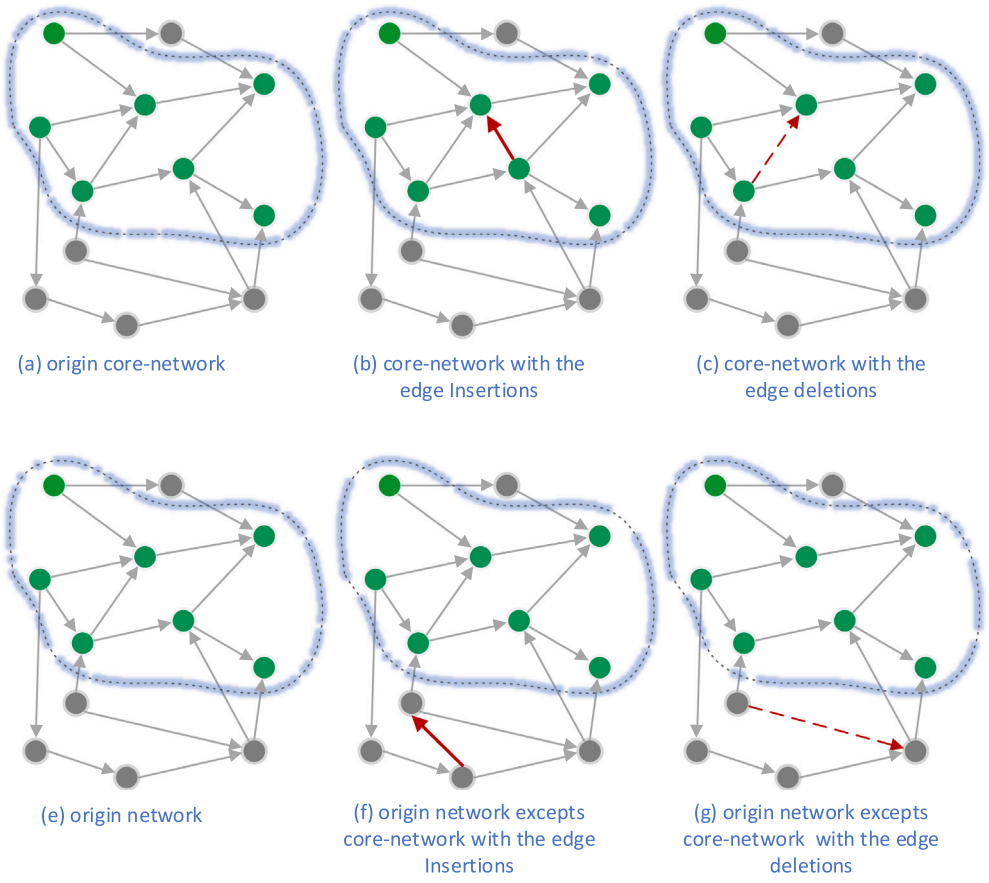


Fig. 6. The edges deletion and insertion operations outside the core network schematic graph.

Algorithm 5: Core-network flow adaptive allocate with edge insertion (CFEI)

Input: Directed graph $G'(V', E', S', T', c, cap)$, Directed graph $G(V, E, S, T, c, cap)$, Commodity demand D on the source node, Maximum flow M, Multi-commodity flow on the core network f, an edge $(a, b) \in E$, a capacity ξ, a cost matrix ψ,

Output: A dictionary of flow flows $dict$ through which commodities can flow on each edge

```
if b ∈ E' and a ∈ E' then
    N = getNeiNod(b);
    for each n ∈ N do
        newPathCost = getPathCost(G,n,a,b,flow);
        oldPathCost = getPathCost(G,n,b,flow);
        if newPathCost < oldPathCost then
            if remainCap(cap(a,b)) ≥ flow(a,b) then
                G'.remove(e(n,b));
                if G'.isNotexist(e(a,b)) then
                    G'.add(e(a,b));
            M = MFD(G');
            dict=MCFD(G', M, D);
            if newPathCost < oldPathCost then
                NEWG'=NPRF(G);
                if NEWG' ≠ G' then
                    F = MFD(NEWG');
                    dict=MCFD(NEWG', M, D);
                    if D > M then
                        dict=CFCD(NEWG', G', D, M)
return dict;
```

Algorithm 6: Core-network flow adaptive update with edge detections (CFED)

Input: Directed graph $G'(V', E', S', T', c, cap)$, Directed graph $G(V, E, S, T, c, cap)$, Commodity demand D on the source node, Maximum flow M, Multi-commodity flow on the core network f, an edge $(a, b) \in E$, a capacity ξ, a cost matrix ψ,

Output: A dictionary of flow flows $dict$ through which commodities can flow on each edge

```
if b ∈ E' and a ∈ E' then
    G.delete(e(a,b));
    NEWG'=NPRF(G);
    if NEWG' ≠ G' then
        F = MFD(NEWG');
        dict=MCFD(NEWG', M, D);
        if D > M then
            dict=CFCD(NEWG', G', D, M)
return dict;
```

and Fig. 6(c) represent the state after the addition and removal of edges to the core network, respectively. These changes reveal how the core network has to adapt to maintain an efficient data flow dynamically.

In addition, Fig. 6(e), Fig. 6(f) and Fig. 6(g) show the addition and deletion of non-core area edges, indicating recently added satellite links or links that are no longer used. In the dynamic upgrading strategy, when the demand for good D exceeds the maximum flow M, the main problem is to upgrade the core network to meet the demand for all goods. Alg. 4 provides a mechanism to adaptively update the traffic distribution without recalculating the entire network when the

Fig. 7. Main satellite network simulation scenario of CSTK software.

demand for a good changes. The ability to react quickly to changes is crucial for satellite networks, especially when they adapt quickly to new situations.

6. Comparison analysis

6.1. Time complexity

The model PAFUDN proposed in this manuscript mainly contains six kinds of algorithms. Alg. 1 (NPRF) mainly calculates the shortest path from all source nodes to sink nodes in the source network and constructs the core network. In the calculation process, the SPFA algorithm is used to solve the shortest path. The time complexity of the algorithm SPFA is $O(VE)$. Therefore, the time complexity of algorithm NPRF is $O(SVE)$. After getting the core network, we need to use Alg. 2 (MFD) to calculate the maximum flow of the core network. The calculation of the maximum flow uses the Ford-Fulkerson algorithm.

The time complexity of the algorithm is $O(Ef)$, where E is the edge in the digital graph, and f is the maximum flow in the graph. This is because each expansion path can be found in $O(E)$ time and increase the flow by at least an integer amount of 1, the upper limit f. Therefore, the time complexity of the algorithm MFD is $O(E)$. After obtaining the maximum flow of the core network, we need to allocate flow to the edges through which the maximum flow passes. As shown in Alg. 3 (MCFD), this algorithm traverses almost all nodes in the core network, so its time complexity is $O(TV')$. If the commodity demand is less than the maximum flow of the current core network, then the problem-solving ends. The time complexity of the PAFUDN algorithm is $O(SVE)$. Meanwhile, the time complexity is the minimum time complexity of the PAFUDN algorithm. If the number of commodity demand is greater than the maximum flow of the core network, it is necessary to use Alg. 4 (CFCD) to extend the path and update the core network. The time complexity of the algorithm CFAA is $O(TV')$. When a new edge is inserted in the network over time, if the insertion of the new edge causes the path in the core network to be the set of shortest paths no longer, then the core network needs to be updated at this time. The time complexity of Alg. 5 (CFEI) is $O(nVE)$. When $num(S) > num(n)$, the time complexity of algorithm PAFUDN is $O(SVE)$. In contrast, if $num(S) < num(n)$, then the time complexity of the algorithm PAFUDN is $O(nVE)$. As time goes by, if the edge in the source network is deleted, then the two nodes that weaken the edge belong to the core network, then the edge is deleted, and the core network is updated. If it does not belong to the core network, it will not impact the final result. Therefore, the time complexity of algorithm 6 CFED is $O(VE)$. In summary, the algorithm PAFUDN proposed in this manuscript has a maximum time complexity of $O(nVE)$, which is bound by constraint $num(S) < num(n)$ after a network update. The minimum time complexity of PAFUDNis $O(SVE)$.

6.2. Datesets details

To address the pivotal challenges associated with resource scheduling and cooperative computing within space-based information networks, we conceived and developed the Common Computing Environment Simulation Toolkit (CSTK). This innovative toolkit, designed specifically for spatial information networks, is built upon authentic datasets derived from existing space-based information networks. Within this framework, we simulate datasets of satellite-related information. This datasets encompasses satellite payload data, orbital trajectories, inter-satellite communication windows, as well as the capacity and availability of satellite resources. The main simulation interface is shown as Fig. 7.

Moreover, the toolkit facilitates the simulation of typical Earth observation scenarios, incorporating variables such as targeted observation regions, temporal requirements for mission execution, and the specific types and quantities of resources needed. To augment the functionality of the CSTK platform, we also engineered a comprehensive suite of computational libraries, tailored to support the diverse computational demands of the toolkit. The scenario management interface and 2d, 3d visualization interfaces for scenarios of CSTK platform are shown in Fig. 8, Fig. 9(a) and Fig. 9(b), respectively.

In this study, the algorithm proposed in Section 4 was evaluated on a pair of distinct datasets tailored for quadratic multi-commodity flow scenarios, particularly within the context of satellite networks. We have generated dynamic network multi-commodity instances by employing the renowned software STK (Satellite Tool Kit)[1] developed by AGI (Analytical Graphics, Inc.) and CSTK platform.

Given the sensitive nature of satellite network simulations, the datasets generated for this research are not publicly available. These

[1] https://www.agi.com/products

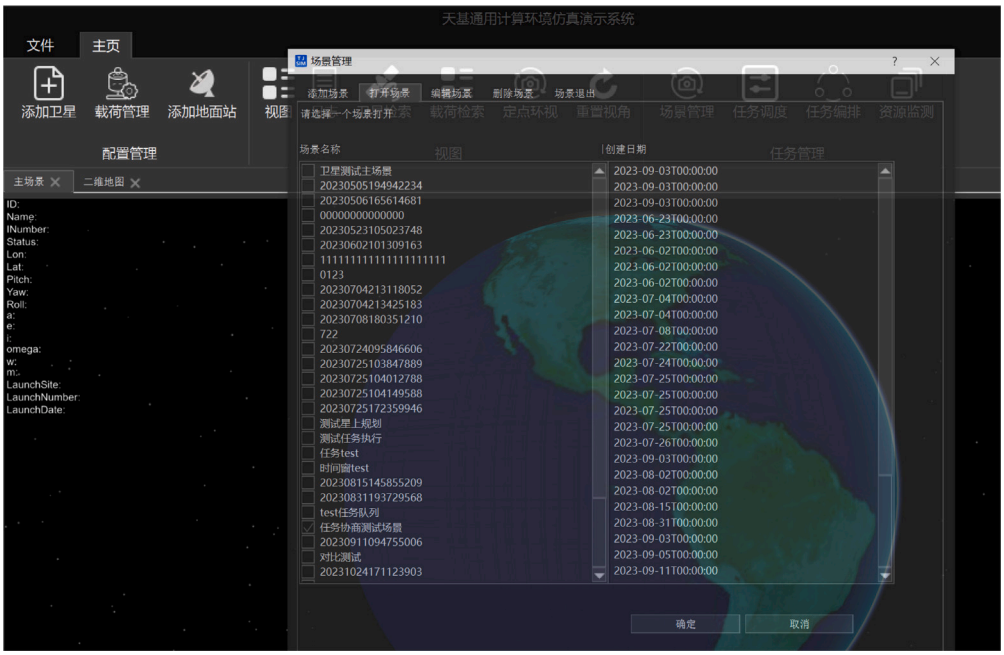
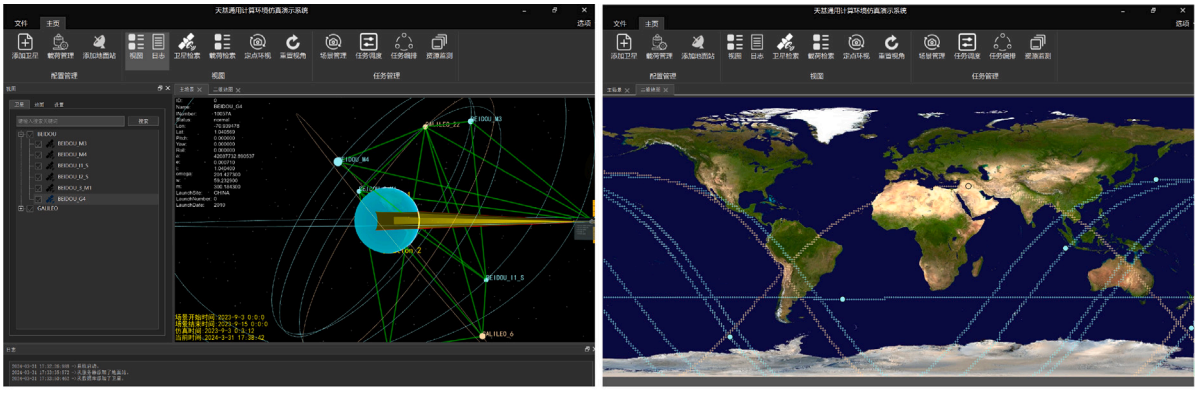


Fig. 8. Scenario management interface of CSTK platform.



(a) 3D scenarios visualization and contact link between satellites (b) 2D scenario visualization and sub-satellite point trajectory maps

Fig. 9. 2D and 3D simulation visualization of certain scenario in CSTK platform.

Table 3
Attribute details of different instances in the experimental data.

Instance	m	n	k
CSTK-RS-1	126	372	11
CSTK-RS-10	1399	4792	11
CSTK-RS-20	2857	10 858	11
CSTK-RS-30	4223	16 148	11
CSTK-RS-40	5652	22 059	11
CSTK-RS-50	7031	27 668	11
CSTK-RS-60	8423	33 388	11
CSTK-RS-70	9750	38 396	11
CSTK-RS-80	10 989	42 472	11
CSTK-RS-90	12 186	46 161	11

datasets, developed on CSTK with 150 actual satellites across 10 simulation scenarios, are tailored for exploring the Multi-Mission Composite Framework (MMCF) problem within satellite networks. Influenced by orbital mechanics and evolving ground communication needs, each scenario represents a unique network configuration, capturing the dynamic nature of satellite networks through variations in edges and commodity demands. This approach simulates real-world challenges of satellite communication management. Details of the initial dataset configurations and the visualization in CSTK platform are described in Table 3 and Fig. 10.

6.3. Parameter settings

Each data instance is updated every 2 seconds, and the updated content is the addition and deletion of edges. The total number of updates is 100. The number of source nodes $0 < s < 10$, the number of sink nodes $0 < d < 20$, the number of each commodity demanded by the source node $n > 1$.

6.4. Experimental results

In addressing the challenges of solving the minimum cost multi-object network flow problem efficiently, it is essential to evaluate the performance of various optimization algorithms in terms of computational efficiency. This study conducts a comparative analysis between the CPLEX Optimizer (CPLEX) [55] and the Interior Point Method Solver (IPM) [37], two widely recognized solutions in the field of optimization. In this experiment, we mainly compare the solution time of the algorithm on the CPU. As shown in Table 4, we mainly focused on

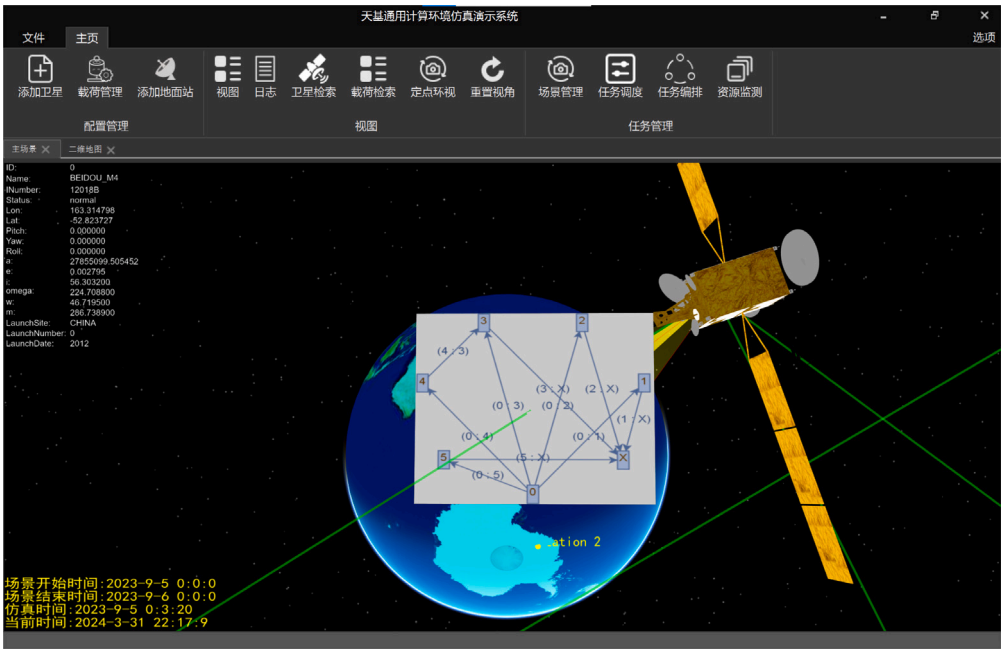


Fig. 10. The visualization of initial MMCF instance in CSTK platform.

Table 4
Comparison of the time consumed by different algorithms for CSTK-RS data update.

Instance	CPLEX	IPM	PAFUDN
	CPU	CPU	CPU
CSTK-RS-1	1.6	1.3	1.2
CSTK-RS-10	234.8	78.6	64.2
CSTK-RS-20	1425.6	271	218.3
CSTK-RS-30	5309.8	938.3	754.1
CSTK-RS-40	10 712.3	1965.2	1367.2
CSTK-RS-50	14 049.7	3163.3	2561.9
CSTK-RS-60	17 133.4	3644.2	2237.6
CSTK-RS-70	25 158.3	5548.7	3282.4
CSTK-RS-80	26 232.1	7029.9	5336.4
CSTK-RS-90	32 412.9	9786.7	7365.2

the experiments on the CSTK-RS data set. We divided the CSTK-RS into 10 data sets, of which 10 data sets contained 100 network updates. It can be seen from Table 4 that the CPLEX solver has the most significant time consumption on different data sets, followed by IPM, and PAFUDN has the most minor time consumption. With the growth of the data set size and the dynamic changes of the network. The time consumption of the three algorithms on the data set is gradually increasing.

As shown in Fig. 11, the CPU runtimes of the PAFUDM algorithm and CPLEX and IPM are compared for processing datasets of different sizes. We can intuitively see that for the small dataset CSTK-RS-1, the time consumption of the three algorithms does not differ much, where CPLEX, IPM and PAFUDN take 1.6, 1.3 and 1.2 s, respectively. However, as the dataset continues to grow, the time consumed by CPLEX increases dramatically, followed by IPM, and PAFUDN takes the least amount of time. For example, for the CSTK-RS-10 dataset, the execution times of CPLEX, IPM and PAFUDN are 234.8 s, 78.6 s and 64.2 s, respectively. On the exceptionally large dataset CSTK-RS-90, the running time of CPLEX, IPM and PAFUDN spikes to 32412.9 s, 9786.7 s and 7365.2 s, respectively. We can clearly see on Fig. 11 that the PAFUDN algorithm shows superior performance compared to the other two algorithms, especially for processing large-scale dynamic network data. This can be attributed to the advanced design of the PAFUDN algorithm, which specifies the dynamic network behavior and actually optimizes the network paths and traffic allocation.

As shown in Fig. 12(a), it shows the time comparison of the three solving methods of CPLEX, IPM and PAFUDN on the data sets CSTK-RS-1, CSTK-RS-10, CSTK-RS-30, and CSTK-RS-40. It can be seen from the figure that as the number of data sets increases, the solving rate of CPLEX increases significantly, especially after CSTK-RS-20, the time consumption of CPLEX increases sharply. The rate of increase of IPM and PAFUDN is the same as before CSTK-RS-20. After CSTK-RS-20, the rate of increase of IPM time consumption is greater than that of PAFUDN.

As shown in Fig. 12(b), it shows the time comparison of the three solving methods of CPLEX, IPM, and PAFUDN on the data sets CSTK-RS-50, CSTK-RS-60, CSTK-RS-70, CSTK-RS-80, and CSTK-RS-90. It can be seen from the figure that with the increase of the number of data sets, the solving rate of CPLEX shows a rapid upward trend, and the time consumption value is much larger than the two algorithms of IPM and PAFUDN. The rate of increase of IPM and PAFUDN is the same, and the rate of increase of IPM time consumption is greater than that of PAFUDN. In summary, As shown in 12, the computational time complexity of CPLEX and IPM solvers is greater than PAFUDN on multiple data sets. Therefore, the algorithm PAFUDN proposed in this manuscript has certain advantages in computing performance. We can see that the fundamental advantage of the PAFUDN algorithm comes from our innovative design above, which addresses the problem of dynamic network systems.PAFUDN, unlike CPLEX and IPM, adapts to network changes in real time and manages the computational resources efficiently. The efficiency of the PAFUDN algorithm not only increases, but also increases with the increase of network complexity, which indicates the robustness of the method in adapting to the demands of big data environments.

The PAFUDN method proposed in this paper has these advantages due to the advanced algorithms used in our design, which are specifically designed for dynamic and large-scale network data to improve computational efficiency and reduce time complexity. In contrast, traditional CPLEX and IPM algorithms are not designed to optimize these dynamic features, resulting in low performance when dealing with a large number of dynamic updates. As shown in Fig. 13, with the two sub-figures Fig. 13(a) Percentage improvement in performance and Fig. 13(b) Performance ratio, we demonstrate the significant advantages of the PAFUDN algorithm over the CPLEX and IPM algorithms on

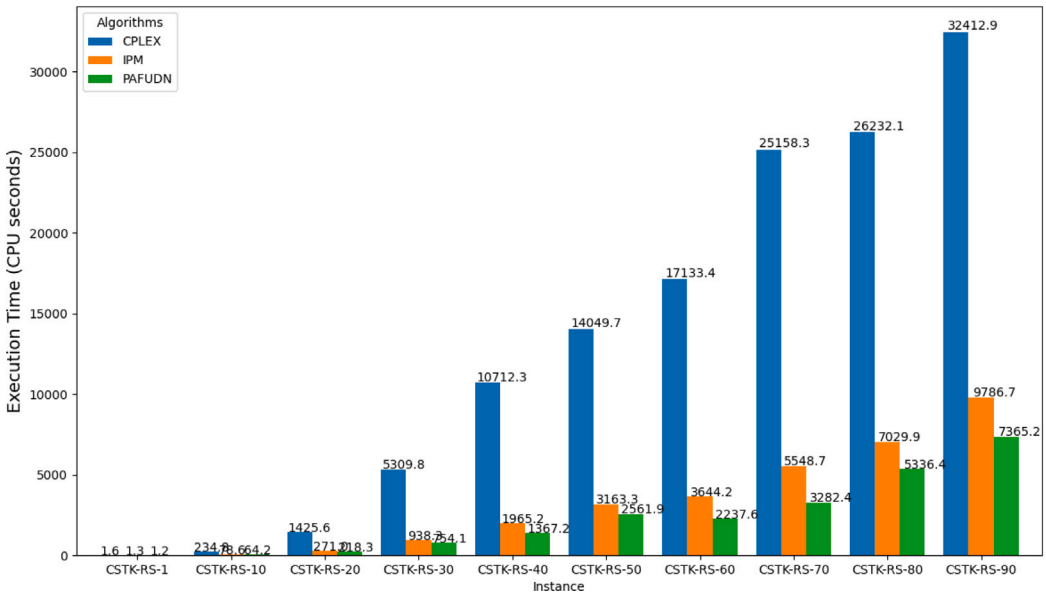


Fig. 11. Comparison of running time of three algorithms CPLEX, IPM and PAFUDN algorithms on different datasets.

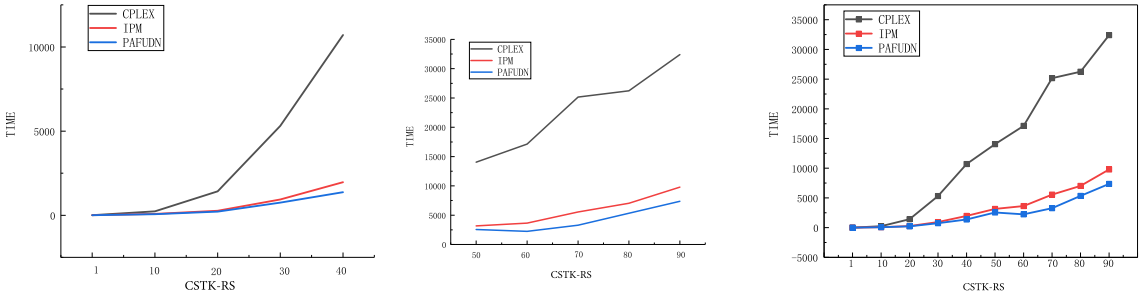
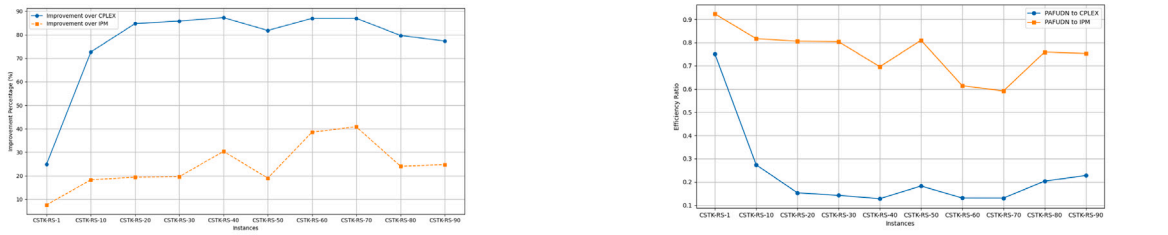


Fig. 12. Performance comparison of three algorithms on datasets.



(a) Performance Improvement Percentage of PAFUDN over CPLEX and IPM

(b) Efficiency Ratio of PAFUDN Compared to CPLEX and IPM

Fig. 13. Comparative Analysis of PAFUDN Algorithm's Performance and Efficiency against CPLEX and IPM.

different example databases. Fig. 13(a) reveals a significant improvement in the performance of the PAFUDN algorithm, especially in the CSTK-RS-10 dataset, which shows an improvement of about 72.7% and 18.3% over the CPLEX and IPM algorithms, respectively. Moreover, in the largest dataset, CSTK-RS-90, the performance improvement of the PAFUDN algorithm still maintains a significant difference of 77.3% and 24.7% over CPLEX and IPM, respectively.

This is further confirmed by the efficiency ratios of PAFUDN with respect to CPLEX and IPM in sub-figure Fig. 13(b). As the dataset size increases, the efficiency ratio of PAFUDN with respect to CPLEX peaks at about 0.8 in CSTK-RS-30, while the efficiency ratio with respect to IPM maintains above 0.8 in most of the instances, suggesting that PAFUDN is more efficient in resource usage.

Thus, the performance and efficiency advantages of PAFUDN reflect its ability to handle complex network problems in real-time dynamic network environments, making it particularly prominent in large-scale dynamic networks.

7. Conclusion

In this paper, we have tackled the challenging problem of the minimum cost flow for large-scale dynamic graphs, an endeavor inspired by the human capability for "prejudgment". We introduced a novel approach that identifies a core network within the dynamic network and dynamically allocates commodity flow to efficiently adapt to the network's continuous evolution. Our model's ability to dynamically update the core network in response to new requirements stands as

a testament to its adaptability and resilience against the fluctuating nature of network dynamics. The primary computational complexity of our approach stems from the application of the shortest path algorithm, which suggests that further improvements in the computational efficiency of our algorithm could be achieved by optimizing the shortest path computation.

Our methodology transforms the daunting task of solving the multi-commodity minimum cost flow (MMCF) problem in a dynamic network into a manageable process. By focusing on the core path for maximum flow distribution and dynamically updating this core path, we have significantly reduced the computational burden typically associated with processing every node and edge within a large network. Our innovative approach to transforming the overall network flow problem into a partial network flow problem has not only simplified the computational process but has also paved the way for more efficient and effective solutions to the MMCF problem.

Our future work will concentrate on enhancing the speed of finding the shortest path within the network. By achieving this, we aim to significantly elevate the computational efficiency of our algorithm. This direction is not only promising but also critical, as it addresses the foundational challenge of optimizing computational resources in the face of complex network dynamics. Additionally, exploring the potential of integrating artificial intelligence and machine learning techniques to predict changes within the network and preemptively adapt the core network could offer a groundbreaking advancement in solving dynamic network flow problems.

CRediT authorship contribution statement

Huilong Fan: Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation. **Chongxiang Sun:** Writing – review & editing, Writing – original draft, Visualization, Validation, Resources, Project administration, Methodology, Funding acquisition, Formal analysis. **Jun Long:** Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Shangpeng Wang:** Writing – original draft, Visualization, Validation, Software, Project administration, Investigation, Funding acquisition. **Fei Zeng:** Visualization, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

This work is supported by the National Key R&D Program of China under Grant 2021YFB3900902, the Theory and Method of Multi-Satellite Collaborative Application under Grant 2019-JCJQZD-342-00.

References

[1] T.N. Dhamala, S.P. Gupta, D.P. Khanal, U. Pyakurel, Quickest multi-commodity flow over time with partial lane reversals, J. Math. Stat. 16 (1) (2020) 198–211.
[2] I.-L. Wang, Multicommodity network flows: A survey, part I: Applications and formulations, Int. J. Oper. Res. (Taichung) 15 (4) (2018) 145–153.
[3] H. Van Hung, T.Q. Chien, Maximal concurrent minimal cost flow problems on extended multi-cost multi-commodity networks, Univ. Danang-J. Sci. Technol. (2021) 29–35.
[4] I. Haasler, A. Ringh, Y. Chen, J. Karlsson, Scalable computation of dynamic flow problems via multi-marginal graph-structured optimal transport, 2021, arXiv preprint arXiv:2106.14485.
[5] S. Khodayifar, Minimum cost multicommodity network flow problem in time-varying networks: By decomposition principle, Optim. Lett. (2019) 1–18.
[6] B. Li, Z. Fei, C. Zhou, Y. Zhang, Physical-layer security in space information networks: A survey, IEEE Internet Things J. 7 (1) (2019) 33–52.
[7] R. Liu, Y. Zhu, Y. Zhang, W. Wu, D. Zhou, K. Chi, Resource mobility aware hybrid task planning in space information networks, J. Commun. Inform. Netw. 4 (4) (2019) 107–116.
[8] R.M. Sundhari, K. Jaikumar, IoT assisted hierarchical computation strategic making (HCSM) and dynamic stochastic optimization technique (DSOT) for energy optimization in wireless sensor networks for smart city monitoring, Comput. Commun. 150 (2020) 226–234.
[9] G. Martinez, S. Li, C. Zhou, Multi-commodity online maximum lifetime utility routing for energy-harvesting wireless sensor networks, in: 2014 IEEE Global Communications Conference, IEEE, 2014, pp. 106–111.
[10] K. Lee, A.S. Alrawahi, D. Toohey, Enabling commodity environmental sensor networks using multi-attribute combinatorial marketplaces, in: 2013 19th Asia-Pacific Conference on Communications, APCC, IEEE, 2013, pp. 115–120.
[11] G.S. Shenoy, J. Tubella, A. González, Exploiting temporal locality in network traffic using commodity multi-cores, in: 2012 IEEE International Symposium on Performance Analysis of Systems & Software, IEEE, 2012, pp. 110–111.
[12] L. Milroy, C. Llamas, Social networks, in: The Handbook of Language Variation and Change, Wiley Online Library, 2013, pp. 407–427.
[13] Y. Shitov, Column subset selection is NP-complete, Linear Algebra Appl. 610 (2021) 52–58.
[14] D. Bienstock, Postential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice, Kluwer Academic Publishers, 2002, pp. 207–212.
[15] D. Robert, Breaking SIDH in polynomial time, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2023, pp. 472–503.
[16] K. Salimifard, S. Bigharaz, The multicommodity network flow problem: State of the art classification, applications, and solution methods, Oper. Res. (2022) 1–47.
[17] A. Trivella, F. Corman, D.F. Koza, D. Pisinger, The multi-commodity network flow problem with soft transit time constraints: Application to liner shipping, Transp. Res. Part E: Logist. Transp. Rev. 150 (2021) 102342.
[18] M. Bodur, S. Ahmed, N. Boland, G.L. Nemhauser, Decomposition of loosely coupled integer programs: A multiobjective perspective, Math. Program. 196 (1) (2022) 427–477.
[19] S.S. Dey, J.R. Luedtke, N.V. Sahinidis, Global solution of integer, stochastic and nonconvex optimization problems, Math. Program. 196 (1) (2022) 1–8.
[20] Q. Guo, C. Zhao, M. Qu, L. Xiong, S.M.H. Hojjatzadeh, L.I. Escano, N.D. Parab, K. Fezzaa, T. Sun, L. Chen, In-situ full-field mapping of melt flow dynamics in laser metal additive manufacturing, Addit. Manuf. 31 (2020) 100939.
[21] M. Skutella, Minimum cost flows over time without intermediate storage, in: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2003, p. 66.
[22] L. Fleischer, M. Skutella, The quickest multicommodity flow problem, in: International Conference on Integer Programming and Combinatorial Optimization, Springer, 2002, pp. 36–53.
[23] M. Fonoberova, D. Lozovanu, Minimum cost multicommodity flows in dynamic networks and algorithms for their finding, Buletinul Academiei de Ştiinţe a Moldovei. Matematica 53 (1) (2007) 107–119.
[24] B. Awerbuch, T. Leighton, Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks, in: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 1994, pp. 487–496.
[25] G. Karakostas, Faster approximation schemes for fractional multicommodity flow problems, ACM Trans. Algorithms (TALG) 4 (1) (2008) 1–17.
[26] M. Chen, Y. Tan, SF-FWA: A self-adaptive fast fireworks algorithm for effective large-scale optimization, Swarm Evol. Comput. 80 (2023) 101–314.
[27] J.C. Poremba, Directed Multicommodity Flows: Cut-Sufficiency and Forbidden Relevant Minors (Ph.D. thesis), University of British Columbia, 2022.
[28] M. Ghatee, S.M. Hashemi, Some concepts of the fuzzy multicommodity flow problem and their application in fuzzy network design, Math. Comput. Modelling 49 (5–6) (2009) 1030–1043.
[29] S. Bavandi, S.H. Nasseri, A hybrid fuzzy stochastic model for fractional multi-commodity network flow problems, Int. J. Math. Oper. Res. 22 (2) (2022) 195–215.
[30] H. Teyeb, A. Balma, N.B.H. Alouane, S. Tata, Optimal virtual machine placement in large-scale cloud systems, in: 2014 IEEE 7th International Conference on Cloud Computing, IEEE, 2014, pp. 424–431.
[31] F. Jiao, S. Dong, Ordered escape routing for grid pin array based on min-cost multi-commodity flow, in: 2016 21st Asia and South Pacific Design Automation Conference, ASP-DAC, IEEE, 2016, pp. 384–389.
[32] S. Bera, S. Misra, A. Jamalipour, Flowstat: Adaptive flow-rule placement for per-flow statistics in SDN, IEEE J. Sel. Areas Commun. 37 (3) (2019) 530–539.

[33] A. Melchiori, A. Sgalambro, A branch and price algorithm to solve the quickest multicommodity k-splittable flow problem, European J. Oper. Res. 282 (3) (2020) 846–857.
[34] N. Farrugia, J.A. Briffa, V. Buttigieg, Solving the multi-commodity flow problem using a multi-objective genetic algorithm, in: 2019 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2019, pp. 2816–2823.
[35] G.R. Waissi, Network Flows: Theory, Algorithms, and Applications, JSTOR, 1994, pp. 110–331.
[36] R. Kia, P. Shahnazari-Shahrezaei, S. Zabihi, Solving a multi-objective mathematical model for a multi-skilled project scheduling problem by CPLEX solver, in: 2016 IEEE International Conference on Industrial Engineering and Engineering Management, IEEM, IEEE, 2016, pp. 1220–1224.
[37] F.A. Potra, S.J. Wright, Interior-point methods, J. Comput. Appl. Math. 124 (1–2) (2000) 281–302.
[38] I. Chabini, Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time, Transp. Res. Rec. 1645 (1) (1998) 170–175.
[39] L. Chen, R. Kyng, Y.P. Liu, R. Peng, M.P. Gutenberg, S. Sachdeva, Maximum flow and minimum-cost flow in almost-linear time, in: 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science, FOCS, IEEE, 2022, pp. 612–623.
[40] H.I. Calvete, L. del Pozo, J.A. Iranzo, The energy-constrained quickest path problem, Optim. Lett. 11 (7) (2017) 1319–1339.
[41] H.S. Fathabadi, S. Khodayifar, M. Raayatpanah, Minimum flow problem on network flows with time-varying bounds, Appl. Math. Model. 36 (9) (2012) 4414–4421.
[42] S.P. Gupta, D.P. Khanal, U. Pyakurel, T.N. Dhamala, Approximate algorithms for continuous-time quickest multi-commodity contraflow problem, Nepali Math. Sci. Rep. 37 (1–2) (2020) 30–46.
[43] S.M. Hashemi, S. Mokarami, E. Nasrabadi, Dynamic shortest path problems with time-varying costs, Optim. Lett. 4 (1) (2010) 147–156.
[44] E. Nasrabadi, R. Koch, Flows over time in time-varying networks, 2014, arXiv preprint arXiv:1401.6053.
[45] R. Karamanis, E. Anastasiadis, M. Stettler, P. Angeloudis, Vehicle redistribution in ride-sourcing markets using convex minimum cost flows, IEEE Trans. Intell. Transp. Syst. 23 (8) (2021) 10287–10298.
[46] T.N. Dhamala, U. Pyakurel, S. Dempe, A Critical Survey on the Network Optimization Algorithms for Evacuation Planning Problems, TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik, 2018.
[47] Y. Yu, D. Calderone, S.H. Li, L.J. Ratliff, B. Açıkmeşe, Variable demand and multi-commodity flow in Markovian network equilibrium, Automatica 140 (2022) 110224.
[48] M. Bisheh-Niasar, R. Azarderakhsh, M. Mozaffari-Kermani, Cryptographic accelerators for digital signature based on Ed25519, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 29 (7) (2021) 1297–1305.
[49] L. Wan, F. Zheng, G. Fan, R. Wei, L. Gao, Y. Wang, J. Lin, J. Dong, A novel high-performance implementation of CRYSTALS-Kyber with AI accelerator, in: European Symposium on Research in Computer Security, Springer, 2022, pp. 514–534.
[50] W.-K. Lee, H. Seo, S.O. Hwang, R. Achar, A. Karmakar, J.M.B. Mera, DPCrypto: Acceleration of post-quantum cryptography using dot-product instructions on GPUs, IEEE Trans. Circuits Syst. I. Regul. Pap. 69 (9) (2022) 3591–3604.
[51] Y. Guo, W. Liu, W. Chen, Q. Yan, Y. Lu, ECLBC: A lightweight block cipher with error detection and correction mechanisms, IEEE Internet Things J. (2024).
[52] J. Kaur, A. Sarker, M. Mozaffari-Kermani, R. Azarderakhsh, Hardware constructions for error detection in WG-29 stream Cipher benchmarked on FPGA, IEEE Trans. Emerg. Top. Comput. PP (99) (2021) 1.
[53] I. Roy, C. Rebeiro, A. Hazra, S. Bhunia, SAFARI: Automatic synthesis of fault-attack resistant block cipher implementations, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (4) (2019) 752–765.
[54] D. Fanding, A faster algorithm for shortest-path-SPFA, J. Southwest Jiaotong Univ. (02) (1994) 207–212.
[55] C.U. Manual, Ibm ilog Cplex optimization studio, Version 12 (1987–2018) (1987) 1.

Huilong Fan received the B.S. degree in network engineering from Nanyang Institute of Technology, Nanyang, China, in 2014, and received M.S. degree in computer science from Guizhou University, Guiyang China, in 2018, He is currently pursuing the Ph.D degree with the school of computer science and engineering, Central South University, Changsha China.

His research interests include Space-based network architecture, coordinated scheduling of space-based network resources.

Chongxiang Sun is currently studying for an undergraduate degree at the school of computer science and engineering, Central South University, Changsha China.

His research interests include Space-based network architecture, coordinated scheduling of space-based network resources.

Jun Long received the B.S. degree in computer software from Changsha Railway Institute, Changsha, China, in 1996, received M.S. degree in computer application technology from Central South University, Changsha, China in 2004, and received Ph.D. degree with the school of computer application technology, Central South University, Changsha, China, in 2011.

He is currently a Professor with the Big Data Institute and the Director of the Network Resources Management and Trust Evaluation KeyLaboratory of Hunan Province, Central South University, China. His main research interests include machine learning, computer vision and natural language processing.

Shangpeng Wang received his B.S. degree in software engineering from Shanxi University, Taiyuan, China, in 2018. He went on to obtain his M.S. degree in computer science from Central China Normal University, Wuhan, China, in 2020. Currently, he is pursuing a Ph.D. degree in the School of Computer Science and Engineering at Central South University, Changsha, China.

His research interests focus on space-based network architecture and coordinated computing of space-based network resources.

Fei Zeng received a bachelor's degree in software engineering from Changsha University in 2015 and a master's degree in software engineering from Guizhou University in 2019. Currently pursuing a Ph.D. at Central South University, his main research interests include computer networks, machine learning and operating systems.