

Scheduling Method for Time-Sensitive Network with No-Waiting Flow

Zeya Li*, Zexiang Liu, Yang Liu, Yutao Zeng, Jinfeng Tang, and Danni Xu

Xi'an Microelectronics Technology Institute, Xi'an 710054, Shaanxi, China

18392600309@163.com, lzexiang@163.com, liuyangailj@163.com,
zengyutao1020@126.com, jinfengtang@126.com, anne_xu2011@163.com

Received 21 November 2024; Revised 1 April 2025; Accepted 8 May 2025

Abstract. With the rapid advancement of communication technology, many time-sensitive applications have emerged, necessitating real-time networks for efficient data transmission. Time-Sensitive Network (TSN) is pivotal in supporting time-critical operations and is widely acknowledged as the optimal solution for meeting real-time network requirements. However, TSN requires effective routing and scheduling strategies to achieve robust real-time performance and bounded low latency. Therefore, this paper proposes a joint routing and scheduling (JRS) method based on a greedy algorithm to address the scheduling challenges associated with time-sensitive traffic in TSN. Specifically, we introduce a bandwidth-balanced routing algorithm to enhance problem-solving success rates while simultaneously reducing node load within the network. Additionally, we present a greedy-based scheduling algorithm that rapidly generates feasible solutions for network scheduling within short time frames. Integrating these routing and scheduling methods expands the solution space for this problem domain. Finally, experimental results demonstrate that our proposed method effectively caters to simple and complex environments encountered in practical use cases with broad applicability. This approach exhibits minimal execution time while maintaining high feasibility in typical topologies; furthermore, it holds the potential to adapt to multiple optimization algorithms.

Keywords: TSN, no-waiting traffic flow, joint routing and scheduling, bandwidth balancing

1 Introduction

In the era of modern technological advancements, the evolution of network technology has emerged as a pivotal driving force behind industry transformation. From facilitating automated control in large-scale industrial production to enabling seamless equipment collaboration in complex medical surgeries, from ensuring precise flight control in aerospace applications to managing emerging intelligent city infrastructure, real-time networks play an indispensable role. Time-Sensitive Network (TSN), renowned for its exceptional attributes such as low latency, minimal jitter, and high reliability, has garnered significant attention within network technology as the optimal solution for addressing real-time application scenarios.

TSN has three types of traffic: Time-Triggered (TT) traffic, Audio Video Bridging (AVB) traffic, and Best Effort (BE) traffic [1, 2]. Real-time performance primarily relies on the scheduling of TT traffic; thus, the scheduling of TT traffic is the core issue in TSN. This problem has been proven to be an NP problem [3]. There are two research aspects concerning TT traffic scheduling: Fixed Routing and Scheduling (FRS) and Joint Routing and Scheduling (JRS). FRS involves planning after specifying the TT path, while JRS integrates routing and scheduling for optimal solutions. Compared to FRS, JRS offers distinct advantages as coupling routing with scheduling expands the solution space, facilitating feasible and optimal solutions.

Therefore, the JRS method represents the most suitable solution for TSN scheduling. However, in addressing the JRS problem, most studies concentrate on achieving improved solution outcomes within predefined network configurations while neglecting the development of a unified approach adaptable to diverse network topologies. To tackle this issue, we propose establishing scalable mathematical models to support future algorithmic extensions. Second, we opt for optimization based on the greedy algorithm, enabling faster acquisition of scheduling results across various scenarios. Additionally, the algorithm presented in this paper can serve directly as the fitness function for subsequent metaheuristic Algorithms.

In summary, because of the complexity of different network topologies, to find a scheduling algorithm that applies to most network environments with low latency and subsequent development, this paper designs a joint

* Corresponding Author

routing and scheduling algorithm for the JRS problem based on greedy strategy while considering bandwidth balancing. Maintaining balanced utilization of each link's bandwidth achieves effective routing and scheduling for TT traffic. The main contributions of this paper are as follows:

1) We developed a mathematical model to depict the TSN joint routing and scheduling problem. The model is well-suited for many use cases and demonstrates excellent extensibility.

2) We propose a balanced traffic joint routing and scheduling method based on a greedy algorithm, a fundamental step toward optimizing the JRS problem. This method applies to a wide range of network topology environments and serves as a scalable underlying algorithm for subsequent algorithm optimization.

3) During testing, Simulation verification encompassing linear, tree, ring, and mixed typical topologies was performed. This provides valuable insights for applying scheduling algorithms in engineering applications.

This paper is structured as follows: Section 2 provides an overview of the related work, Section 3 outlines the algorithm design, Section 4 presents simulation analysis, and finally, Section 5 offers a comprehensive summary.

2 Related Works

The present research primarily investigates FRS research, which currently encompasses two primary methodologies: precise methods and heuristic methods.

For precise methods, Özkan et al. [4] define the no-wait communication constraints for TSN scheduling and propose a novel iterated scheduling approach with a backtracking mechanism and partial solution support extending the conventional IIS procedure. Zhu et al. [5] for the application of TSN technology in train communication networks, a single frame simple scheduling method-based SMT is proposed to effectively reduce the entire network's worst end-to-end delay, improving the real-time performance of TSN-based TCN. These studies suggest that precise solutions are suitable for achieving high-precision solutions in small-scale networks.

In contrast, heuristic methods do not guarantee the discovery of the optimal solution but can efficiently find reasonable solutions within a short timeframe. Guo et al. [6] design a schedulability-aware routing algorithm based on the improved ant colony algorithm to enhance the schedulability of time-triggered flows under the no-wait scheduling problem. Yang et al. [7] presented an industrial application scheduling strategy based on taboo search, which effectively reduces execution time and increases the number of scheduled time-sensitive flows by 26%, surpassing SMT-based algorithms. Xu et al. [8] propose an online scalable scheduling and routing scheme for TSN. A novel slot occupancy representation model based on divisibility theory is established to transform the clique-based to a three-level tree-based mode in complex IIoT scenarios. Yang et al. [9] proposed a deep learning-based joint optimization framework.

Therefore, we can conclude that for TT's JRS scheduling problem, it is recommended to use precise methods in scenarios with simple applications or low traffic volumes in the network. The result will be the unique optimal solution under specified constraints [10-13]. However, implementing this method also relies on the Mathematical solver. In such cases, heuristic algorithms can be employed to search for local optima. Heuristic algorithms are problem-solving techniques that prioritize finding reasonable solutions quickly rather than guaranteeing optimality [14, 15]. Metaheuristic algorithms can be utilized if conditions permit further optimization beyond local optima [16].

However, current studies predominantly utilize meta-heuristic algorithms to optimize networks under specific conditions. In practical usage scenarios, there is a demand for an algorithm that is convenient for subsequent development and applicable to diverse situations. It also consumes minimal computing resources while providing rapid scheduling outcomes. Consequently, the research objective of this paper is to address the issues above by designing a TSN joint routing and scheduling algorithm that is universal, computationally efficient, development-friendly, and provides valuable references for practical engineering applications.

3 Algorithm Design

This paper proposes a bandwidth-balanced joint routing and scheduling algorithm based on a greedy approach. The mathematical model for scheduling and the system model for the algorithm is defined. This section introduces the designed routing and scheduling algorithm and explains the joint routing and scheduling algorithm process.

3.1 System Model

The system model contains the network model and application model. The network model contains fundamental elements, delay models, and minimum time units. The application model delineates hardware scenarios applicable to the algorithm.

Network Model. The TSN network contains three fundamental components: hardware devices, network connections, and communication traffic. It can be represented as $G = \langle N, E, F \rangle$, where N represents all the nodes in the network, $N = \{SW, ES\}$, consisting of two types: n switches and m end stations. $SW = \{SW_1, \dots, SW_n\}$, and $ES = \{ES_1, \dots, ES_m\}$. The E represents all k undirected edges in the network denoted as $E = (E_1, \dots, E_k)$. Since each undirected edge operates in full-duplex mode during working, it is represented by two directed edges $E_k = (E_{k,1}, E_{k,2})$.

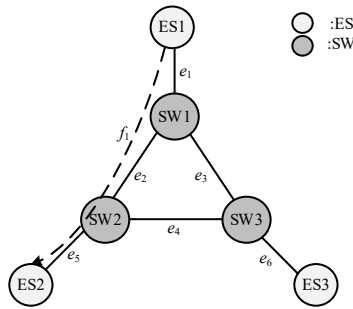


Fig. 1. Network model

In practical scenarios, the length of network connections affects transmission delay, which is reflected in modeling network latency. F represents l communication flow in the network expressed as $F = \{f_1, \dots, f_l\}$. Each flow has five essential attributes: $f_i = (f_{i,s}, f_{i,d}, f_{i,p}, f_{i,l}, f_{i,e})$, representing the sending node of the flow, receiving node of the flow, sending period, fixed size, and deadline, respectively. Fig. 1 illustrates a simple ring network structure comprising 3 switches, 3 end stations, 6 undirected edges, and one scheduled TT flow.

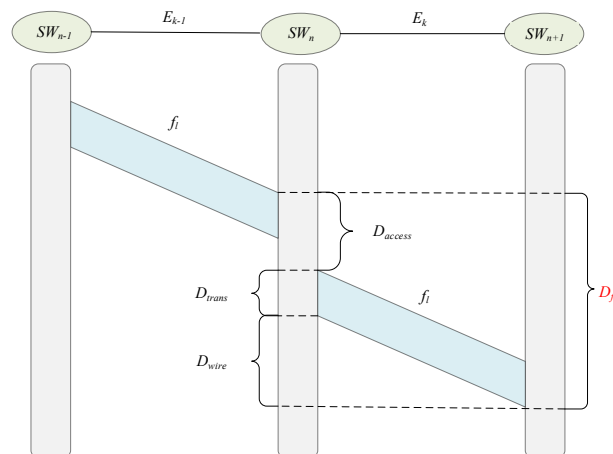


Fig. 2. Delay model

The delay D_f of flow f in TSN networks, as depicted in Fig. 2, can be expressed as $D_f = D_{trans} + D_{wire} + D_{access}$. Here, D_{trans} represents the forwarding delay, D_{wire} represents the wire delay, and D_{access} represents the processing delay. These three parameters exhibit a linear relationship with the size of the processed flow. Hence, the delay can be formulated as $D_f = a \times f_{l,s}$, where a denotes the linear adjustment parameter for delays. Furthermore, considering that different communication rates correspond to distinct clock frequencies in practical usage, we introduce *microtick* to represent the configurable minimum operational unit. All flows' sizes in algorithm are multiples of *microtick*, and time offsets in scheduling results are also quantified by their respective number of *microtick*.

Application Model. TSN contains a comprehensive set of protocols. Data shaping mechanism is the primary protocol influencing TT scheduling. Here, we present an application model suitable for the algorithm discussed in this paper. If additional shaping protocols are employed, modifying the algorithm appropriately based on actual additions becomes imperative.

The algorithm described in this paper implements a no-waiting data flow for scheduling traffic, which is suited for the application model of Time Aware Shaper (TAS) + Frame Preemption (FP). No-waiting data flow refers to the exclusive storage of only one TT data stream within each port's TSN internal storage buffer at any given time, ensuring no contention between individual port data.

TAS is the core mechanism for implementing real-time scheduling of TT traffic, as shown in Fig. 3. Data shaping primarily occurs at the egress ports of switch devices. Each egress port consists of eight queues that store data with priorities ranging from 0 to 7. Queues 0 to 3 hold BE data, queues 4 to 5 hold AVB data, and queue 7 holds TT data. Gate Control Lists (GCL) are set up for each queue to control their opening/close, enabling spatial isolation for high-priority queues. Through the GCL mechanism, temporal isolation is achieved for high-priority queues, ensuring absolute real-time performance in both time and space while keeping jitter within a computable range for TT data. On the other hand, the FP mechanism serves as a shaping method for lower-priority AVB and BE data by allowing fragmented transmission without interrupting TT traffic transmission. Adding the FP mechanism does not affect the algorithm designed in this paper; therefore, it is an optional shaping mechanism applicable to our proposed algorithm.

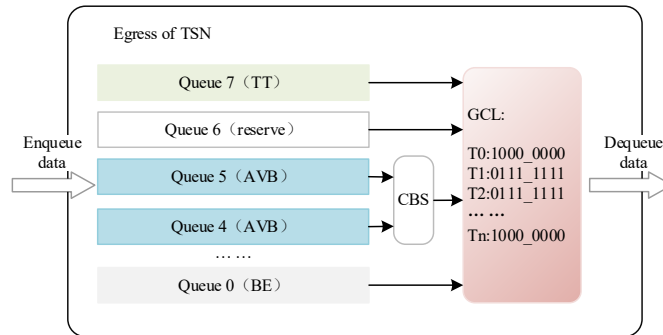


Fig. 3. Application model

Table 1. Model parameter

Parameters	Meaning	Parameters	Meaning
G	TSN collection	$f_{l,e}$	Deadline of flow l
N	Set of network nodes	D_f	Delay of flow l
E	Set of network edges	D_{trans}	Forwarding delay
F	Set of traffic flows	D_{wire}	Wire delay
SW	Set of switches	D_{access}	Processing delay
ES	Set of end stations	a	Linear delay adjustment
$E_{k,1/2}$	Number of Edge E_k	<i>microtick</i>	Configurable minimum operational unit
$f_{l,s}$	Sending node of f_l	SR	Set of decision variables
$f_{l,d}$	Receiving node of f_l	O	Time offsets of scheduling flows
$f_{l,p}$	Period of f_l	R	Route result
$f_{l,l}$	Size of f_l		

The parameter variables involved in the model have been represented in Section 3.1.1. Additionally, a set of decision variables, $SR = \{O, R\}$, is necessary to represent the result of the network. O represents the time offsets of scheduling flows on all network edges, expressed as multiples of *microtick*. R represents binary matrices ranging from 0 to 1, indicating the routing results for each flow. Table 1 presents all parameters included in the system model.

3.2 Joint Routing Scheduling Algorithm

The JRS problem contains two components: routing and scheduling. Routing involves determining the optimal sequence of edges for all scheduled flows. Scheduling planning the initial transmission time for each flow (as subsequent time offsets for all subsequent edges can be calculated based on the initial offset). In Fig. 4, where three flows await scheduling on an edge: f_1 with a period of 10 *microtick* and length of 1 *microtick*; f_2 with a period of 20 *microtick* and length of 3 *microtick*; and f_3 with a period of 10 *microtick* and length of 2 *microtick*. The objective is to achieve conflict-free scheduling within a timeline. Here, we introduce the concept of *hyperperiod* as the least common multiple among all flows awaiting scheduling. A feasible and reasonable schedule only necessitates no conflicts in the scheduling results for all flows within one *hyperperiod*.

The scheduling result in the example: *hyperperiod* length is 20 *microtick*. The f_1 window offset ranges from 0 to 1 *microtick* and 10 to 11 *microtick*, while the f_2 window offset spans from 6 to 9 *microtick*. For f_3 , the window offset extends from 3 to 5 *microtick* and 13 to 15 *microtick*. This completes traffic scheduling for one edge, and successful overall network scheduling requires satisfaction of all edge schedulings.

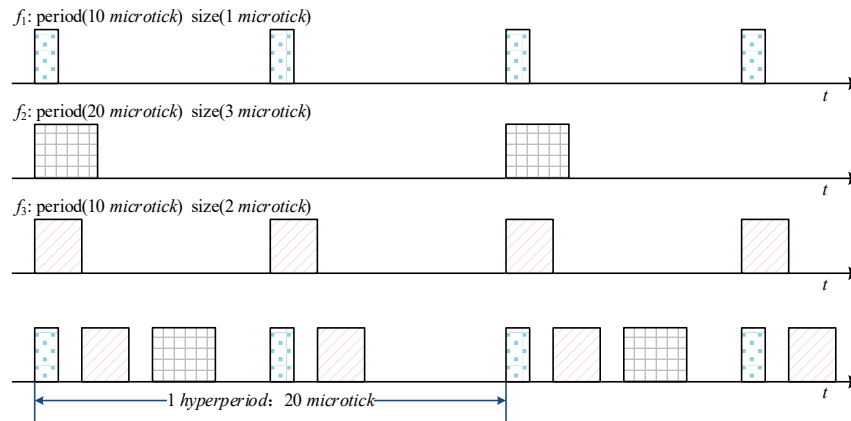


Fig. 4. Example diagram of GCL calculation

We present a joint routing and scheduling method based on the greedy algorithm incorporating bandwidth balancing. This method comprises Algorithm 1, which integrates bandwidth balancing into the routing algorithm, and Algorithm 2, which builds upon the greedy scheduling algorithm. The inputs of this method include fundamental network information G , a delay linear adjustment parameter a , and a configurable minimum operation unit *microtick*. The output of this method is a set of joint routing and scheduling results SR . The specific steps involved in this method are as follows.

Algorithm 1 incorporates bandwidth balancing into the routing process. Its inputs include the fundamental network information G , the parameter a for linear delay adjustment, and the configurable minimum operation unit *microtick*. As an output, this algorithm provides the routing result R .

Algorithm 1. Routing algorithms incorporating bandwidth balancing:Input: $G = \langle N, E, F \rangle$, a , *microtick*.Output: R .

```

1   for all_node
2     Initial  $M$ . // Calculate the adjacency matrix for the entire network.
3      $max\_band = lcm(F)$ . // Calculate the least common multiple of all traffic in the network.
4      $initial\_flow\_band$  // Calculate the bandwidth utilization of all links.
5     for all  $F$ 
6       recalculate  $flow\_band$ . // Update the bandwidth utilization of all links.
7       for all  $E$  // Update the adjacency matrix  $M$  based on the bandwidth occupancy.
8         If  $flow\_band \in (0,0.1)$   $M.E = M.E + 1$ .
9         else if  $flow\_band \in (0.1,0.2)$   $M.E = M.E + 2$ .
10        else if  $flow\_band \in (0.2,0.5)$   $M.E = M.E + 3$ .
11        else  $M.E = inf$ 
12      end
13    end
14    get path by Dijkstra. // Calculate the shortest path by the Dijkstra algorithm.
15    get route result. // Get routing result.
16  end
17  get  $R$ . // Get the final routing results.

```

based on the information in network G , the network's adjacency matrix is calculated in lines 1) ~2). Here, adjacent nodes are assigned an essential distance of 1 and adjusted according to line length in actual usage. The least common multiple of all flows in the network is computed in line 3). Line 4) utilizes the result from line 3) to calculate the $flow_band$ matrix for bandwidth utilization on each edge. Routing results for each link are computed in lines 5) ~16). Specifically, lines 6) ~13) calculate the $flow_band$ matrix based on the current routing result and update corresponding positions in adjacency matrix M accordingly. Line 14) computes the routing path for the current flow using the updated adjacency matrix M . Line 15) represents the current routing result. Finally, line 17) outputs the final routing results R after completing routing for all flows.

Algorithm 2 incorporates greed into the scheduling process. It inputs the fundamental network information G , the delay linear adjustment parameter a , and the configurable minimum operation unit *microtick*. The output of this algorithm is the scheduling time offset O .

Algorithm 2. Time table algorithm based on greed:Input: $G = \langle N, E, F \rangle$, a , *microtick*.Output: O .

```

1    $time\_slot = lcm(F)$ . // Calculate the length of the rearrangeable time slots.
2    $schedule\_flag = 1$ . //Schedule flag.
3   for all  $F$ 
4     for all  $time\_slot$ 
5       update offset result. //The scheduling results are updated According to the current time slot.
6        $E\_slot = lcm(E, F)$ . // Calculate the minimum common multiple of the period of current traffic
       on each edge.
7       for all  $E$ 
8         check deadline. // Check compliance with deadlines.
9         check conflict. // Check conflict.
10        if deadline_ok & conflict_ok
11           $schedule\_flag = 1$ .
12          break; // The current scheduling is optimal, so proceed with scheduling the next flow.
13        end
14      end
15    end
16  end
17  get  $O$ . // Update all scheduling results  $O$ .

```

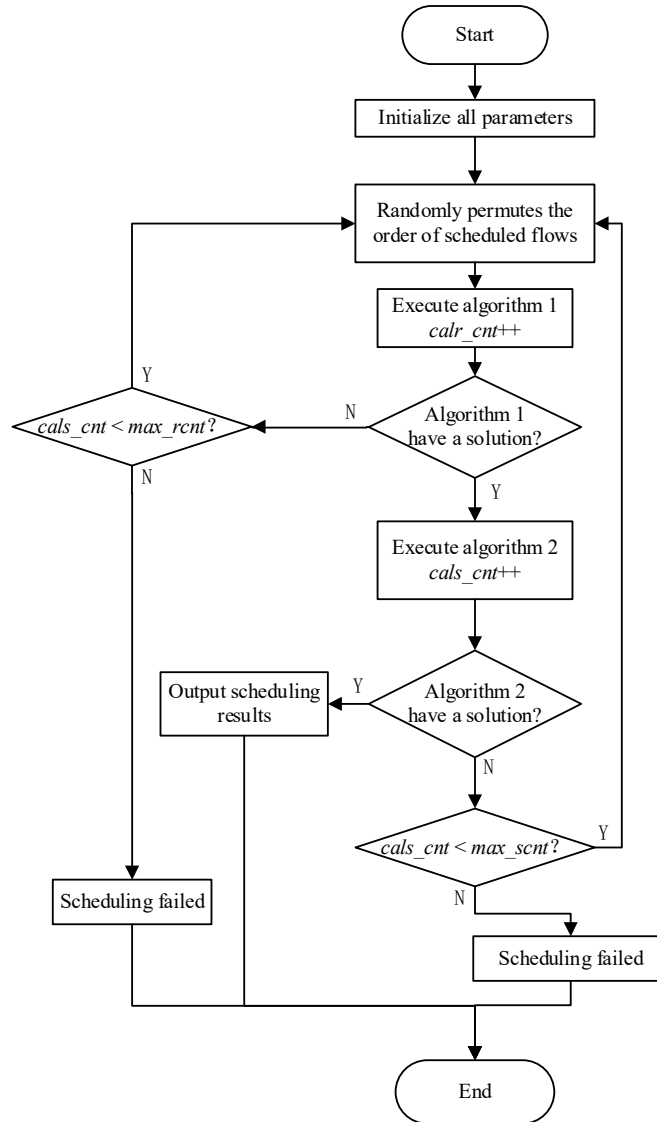


Fig. 5. Workflow of the joint routing and scheduling algorithm

Line 1) calculates the length of the time slot that can be arranged, and the least common multiple of all traffic periods is taken as the maximum time slot that can be arranged, denoted as $time_slot$. Line 2) initializes the scheduling flag $schedule_flag$ to 1. Lines 3) to 16) complete the scheduling of all flows, whereas lines 3) and 4) indicate arranging each flow sequentially within the $time_slot$. Line 5) represents updating the scheduling result $offset_result$ based on the current arrangement. Line 6) calculates the least common multiple E_slot for scheduling traffic on each link in the network. Lines 7) to 13) represent checking deadlines and conflicts for current scheduling. While they all meet requirements, exit from this loop and proceed with scheduling for the next flow. After completing scheduling for all flows, line 17) outputs schedule result R .

Combining algorithm 1 and algorithm 2, a feasible solution for route scheduling can be obtained, denoted as $SR = \{O, R\}$. The implementation steps of the algorithm are shown in Fig. 5. Outlined as follows:

Step 1: Configure the algorithm parameters, scheduling flows, network topology, fixed adjustment parameter values, number of algorithm iterations, and upper and lower bounds.

Step 2: Randomly generate the sequence for scheduled flows.

Step 3: Execute Algorithm 1 and record the current count of routing calculations $calr_cnt$.

Step 4: Determine if Algorithm 1 has completed its calculation. If successful, proceed to Step 6; otherwise, move to Step 5.

Step 5: Check if $calr_cnt$ has reached the maximum preset count for routing calculations max_rcnt . If it has, proceed to Step 10; otherwise, return to Step 2.

Step 6: Execute Algorithm 2 and record the current count of routing calculations $cals_cnt$.

Step 7: Determine if Algorithm2 has completed its calculation. If yes, proceed to step 9; otherwise, go to step 8.

Step 8: Check if $cals_cnt$ has reached the maximum preset count for routing calculations max_scnt . If it has been reached, go to step10; otherwise, return to step2

Step 9: Output scheduling results

Step 10: Scheduling failed.

4 Evaluation

To validate the design algorithm's efficacy, we compare the algorithm's time consumption, Bandwidth utilization, and scheduling success ratio within different flows and network Scales using commonly encountered topology structures. A bandwidth-unaware greedy algorithm (contrast algorithm 1) and an FRS algorithm with separation of routing and scheduling (contrast algorithm 2) are employed as a contrast group.

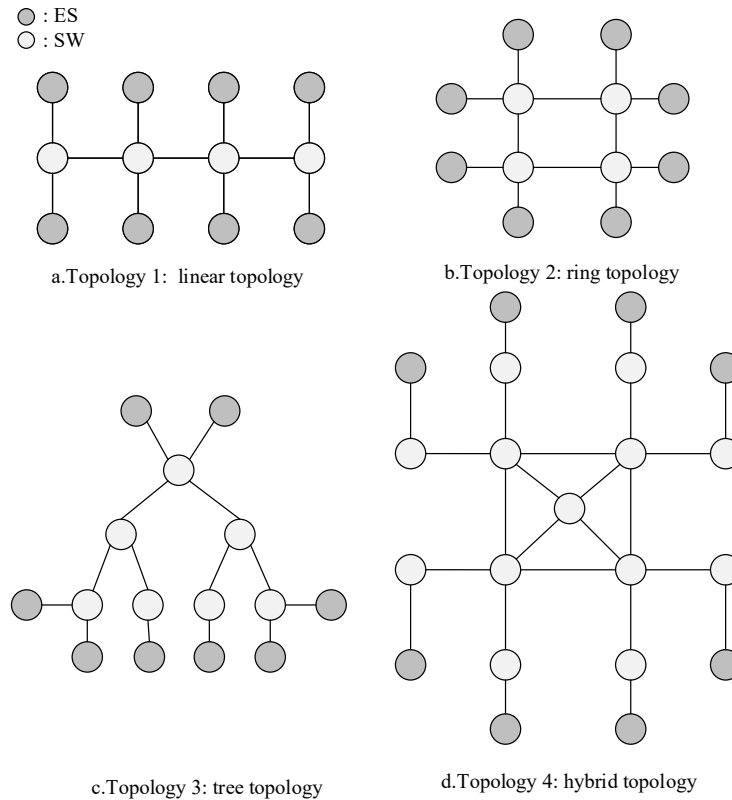


Fig. 6. The simulation verifies the network topology

4.1 Hardware Condition and Network Description

The experimental software employed was Matlab 2018b, and the hardware configuration for the experiment comprised an AMD Ryzen 5 4500U processor operating at a clock frequency of 2.5GHz, accompanied by 8GB RAM, and running on the Windows 11 operating system. Four distinct network topologies were utilized in the

experiment, as illustrated in Fig. 6, encompassing linear topology, ring topology, tree topology, and hybrid topology. To meet standard usage scenario requirements, the maximum number of hops for the path switch within the experimental network was constrained to 7. The length of flow ranged from 1 to 20 *microtick*. Period sizes between 100 and 2000 *microtick*. The deadline time corresponded with the period size.

4.2 Simulation Analysis

Fig. 7 to Fig. 10 indicate the variation of algorithm execution time with the number of scheduling flows under four different topology connections. Experiments were conducted on 20/ 40/ 100 /200 /300 /400 /500 /600 /700 /800 /900 /1000 scheduling flows for contrast. Compared to contract algorithm 1, our proposed algorithm reduces average execution time by 34.18% in Topology 1, 36.62% in Topology 2, 73.24% in Topology 3, and 38.83% in Topology 4. Compared to contract algorithm 2, our proposed algorithm reduces average execution time by 27.68% in Topology 1, 28.34% in Topology 2, 65.67% in Topology 3, and 32.46% in Topology 4. The trend of execution time curves varies with an increase in the number of scheduling flows across different topologies; our algorithm demonstrates superior performance as it effectively balances bandwidth utilization on each link, thereby expanding the solution space range for each link and reducing scheduling difficulty when there is a larger volume of scheduling traffic and feasible solutions exist.

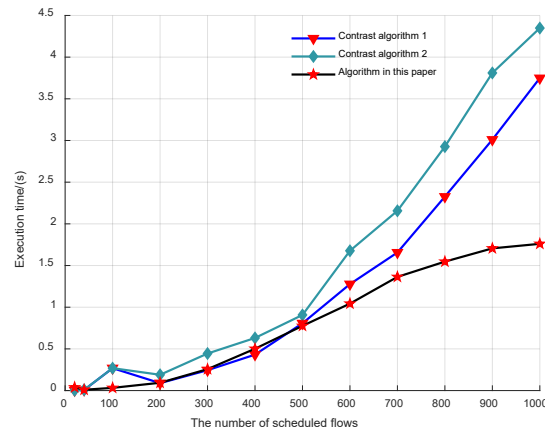


Fig. 7. Comparison of algorithm time consumption under different numbers of scheduled flows for Topology 1

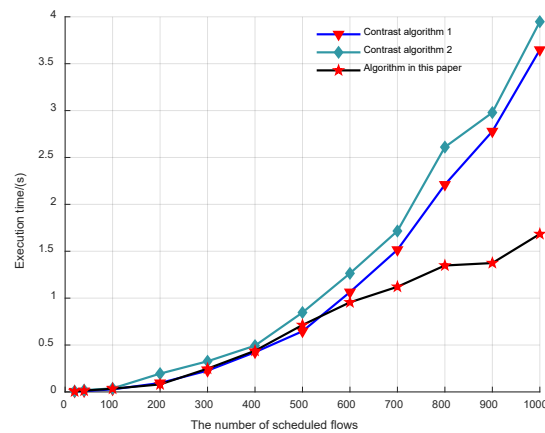


Fig. 8. Comparison of algorithm time consumption under different numbers of scheduled flows for Topology 2

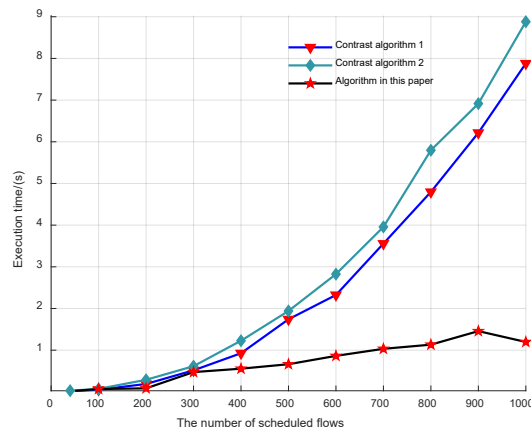


Fig. 9. Comparison of algorithm time consumption under different numbers of scheduled flows for Topology 3

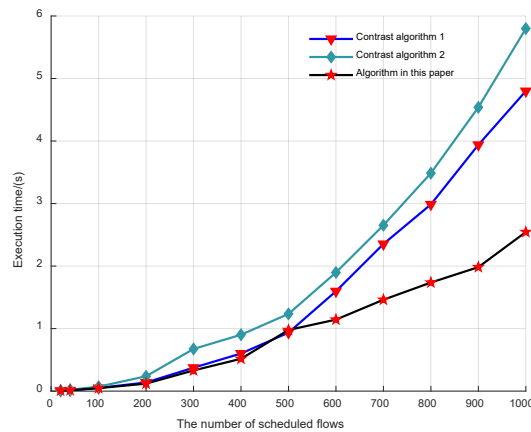


Fig. 10. Comparison of algorithm time consumption under different numbers of scheduled flows for Topology 4

The box plots in Fig. 11 to Fig. 13 illustrate the bandwidth utilization on each edge after scheduling. In this experiment, each end station is exclusively connected to one switch, ensuring consistent bandwidth utilization across our and comparative algorithms. To more effectively demonstrate the superiority of our scheduling algorithm, we present only the box plots of bandwidth utilization on links between switches. Taking four different topologies with 1000 scheduled flows as an example, it can be observed that our algorithm consistently achieves higher bandwidth utilization than the comparative algorithms across all four topologies.

Meanwhile, we also analyzed the scheduling success ratio of the algorithm, as depicted in Fig. 14. The primary factors influencing the scheduling success ratio are the solution space's size and the scheduling strategy's quality. Under identical conditions, Topology 4 was selected as the experimental network. To more intuitively illustrate the differences in the algorithm's scheduling success ratio, the frame length of all flows was set to exceed 200 Bytes during the network environment configuration. This ensured that scheduling failures occurred more rapidly. For each group of experiments, the number of trials was fixed at 100. Based on the configuration above, a trend graph illustrating the scheduling success rate of the algorithm as the number of scheduling flows increased was obtained. As shown in Fig. 14, the scheduling success ratio of all algorithms decreases with an increase in the number of scheduled flows. However, the decline in the scheduling success ratio of our algorithm is significantly less pronounced than that of the comparison algorithm. This demonstrates that our algorithm exhibits clear advantages in achieving results within a fixed solution space.

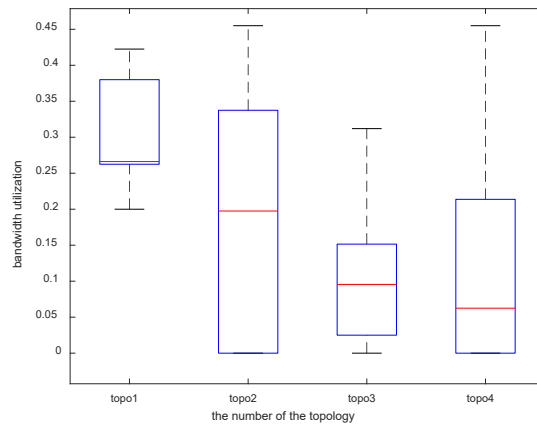


Fig. 11. The box diagram of the bandwidth utilization of contract algorithm 1

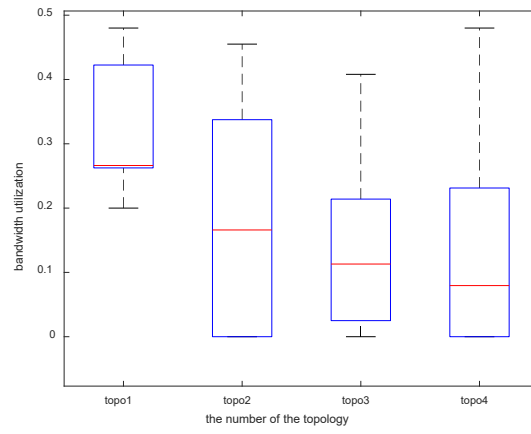


Fig. 12. The box diagram of the bandwidth utilization of contract algorithm 2

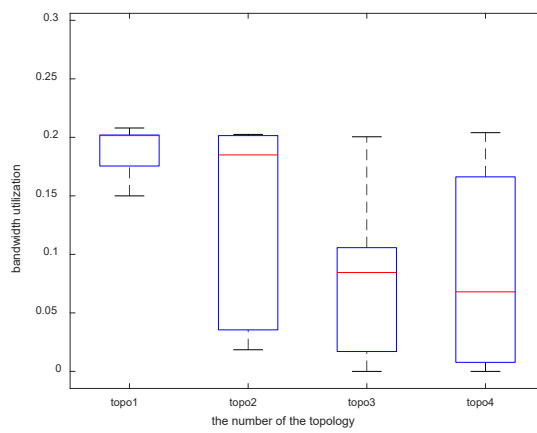


Fig. 13. The box diagram of the bandwidth utilization of algorithm in this paper

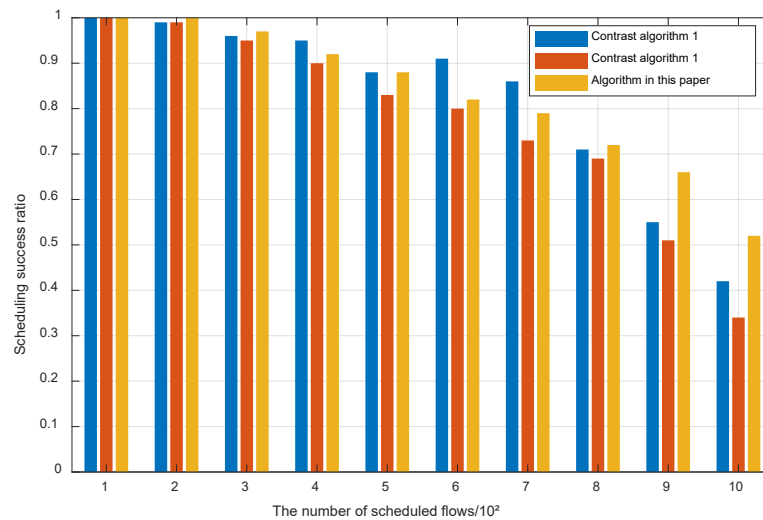


Fig. 14. The trend of algorithm scheduling success ratio with the number of scheduled flows

5 Conclusion

This paper proposes a greedy-based bandwidth-balanced time-sensitive network joint routing and scheduling algorithm to solve the scheduling problem in TSN and ensure no waiting for real-time communication of flows. The algorithm is validated in four typical topological networks: linear, tree, ring, and hybrid. Experimental results demonstrate that the designed algorithm exhibits high efficiency across different network topologies and can serve as a fundamental component of metaheuristic algorithms, laying a solid foundation for future optimization scheduling research.

The algorithm presented in this paper, however, has certain limitations. Instead of providing the optimal solution, it generates a set of feasible solutions that only meet the basic lower limit indicator requirements without achieving the theoretically optimal network state. Therefore, further research on network optimization will be conducted based on this foundation to attain an optimal network configuration state.

References

- [1] Y. Liu, Z. Li, L. Gong, D. Xu, J. Tang, The state-of-the-art research for time-sensitive network and future research interests, *Microelectronics & Computer* 39(6)(2022) 1-11.
<https://dx.doi.org/10.19304/J.ISSN1000-7180.2022.0070>
- [2] Z. Feng, L. Gong, D. Xu, Y. Liu, ILP-based dynamic flow balancing scheduling algorithm in time sensitive network, *Microelectronics & Computer* 38(6)(2021) 33-37.
<https://doi.org/10.19304/j.cnki.issn1000-7180.2021.06.006>
- [3] L. Xu, Q. Xu, C. Chen, Y. Zhang, S. Wang, X. Guan, Efficient Task-Network Scheduling With Task Conflict Metric in Time-Sensitive Networking, *IEEE Transactions on Industrial Informatics* 20(2)(2024) 1528-1538.
<https://doi.org/10.1109/TII.2023.3278883>
- [4] A.K. Özkan, S. Cevher, Enhanced conflict-aware iterated integer linear programming for IEEE 802.1 Qbv Time-Sensitive Network scheduling, *International Journal of Communication Systems* 37(17)(2024) e5920.
<https://doi.org/10.1002/dac.5920>
- [5] G. Zhu, X. Nie, Y. Li, K. Ma, Y. Yang, An Efficient Time-Sensitive Networking Traffic Scheduling Method for Train Communication Network, *IEEE Transactions on Electrical and Electronic Engineering* 20(3)(2025) 394-404.
<https://doi.org/10.1002/tee.24210>
- [6] Y. Guo, F. Luo, Z. Wang, Y. Tong, Y. Ren, A schedulability-aware routing algorithm for time sensitive network based on improved ant colony algorithm, *Ad Hoc Networks* 169(2025) 103741.
<https://doi.org/10.1016/j.adhoc.2024.103741>

- [7] D. Yang, K. Gong, J. Ren, W. Zhang, W. Wu, H. Zhang, TC-flow: Chain flow scheduling for advanced industrial applications in time-sensitive networks, *IEEE Network* 36(2)(2022) 16-24.
<https://doi.org/10.1109/MNET.007.2100444>
- [8] L. Xu, Q. Xu, Y. Zhang, S. Wang, C. Chen, X. Guan, OSSR: Online Scalable Scheduling and Routing for Industrial Time-Sensitive Networking, *IEEE Transactions on Network Science and Engineering* 12(3)(2025) 1761-1775.
<https://doi.org/10.1109/TNSE.2025.3538792>
- [9] L. Yang, Y. Wei, F. R. Yu, Z. Han, Joint Routing and Scheduling Optimization in Time-Sensitive Networks Using Graph-Convolutional-Network-Based Deep Reinforcement Learning, *IEEE Internet of Things Journal* 9(23)(2022) 23981-23994.
<https://doi.org/10.1109/JIOT.2022.3188826>
- [10] X. Hong, Y. Xi, P. Liu, Resource-Aware Online Traffic Scheduling for Time-Sensitive Networking, *IEEE Transactions on Industrial Informatics* 20(12)(2024) 14267-14276.
<https://doi.org/10.1109/TII.2024.3449988>
- [11] IEEE Standard for Local and metropolitan area networks-- Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams, *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)* (2010) 1-84.
<https://doi.org/10.1109/IEEESTD.2009.5375704>
- [12] IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks Amendment 34: Asynchronous Traffic Shaping, *IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020)* (2020) 1-151.
<https://doi.org/10.1109/IEEESTD.2020.9253013>
- [13] IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 29: Cyclic Queuing and Forwarding, *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017)* (2017) 1-30.
<https://doi.org/10.1109/IEEESTD.2017.7961303>
- [14] T. Stüber, L. Osswald, S. Lindner, M. Menth, A Survey of Scheduling Algorithms for the Time-Aware Shaper in Time-Sensitive Networking (TSN), *IEEE Access* 11(2023) 61192-61233.
<https://doi.org/10.1109/ACCESS.2023.3286370>
- [15] G. Patti, L.L. Bello, L. Leonardi, Deadline-Aware Online Scheduling of TSN Flows for Automotive Applications, *IEEE Transactions on Industrial Informatics* 19(4)(2023) 5774-5784.
<https://doi.org/10.1109/TII.2022.3184069>
- [16] H. Nie, S. Li, Y. Liu, An Enhanced Routing and Scheduling Mechanism for Time-Triggered Traffic with Large Period Differences in Time-Sensitive Networking, *Applied Sciences* 12(9)(2022) 4448.
<https://doi.org/10.3390/app12094448>