A Priority-Driven Adaptive Genetic Algorithm for Time-Sensitive Network Traffic Scheduling

Wei Zheng[1], Zhiqiang Zhu[1,2], Yu Zhang[1], Jin Shao[1]*

[1] School of Software, Northwestern Polytechnical University, China
[2] AVIC Xi'an Aeronautics Computing Technique Research Institute, China
wzheng@nwpu.edu.cn, zhuzq07@163.com, zy02@mail.nwpu.edu.cn, shaojin@mail.nwpu.edu.cn

Abstract

As a network technology specifically designed to meet the demands of real-time data transmission, Time-Sensitive Networking (TSN) plays a crucial role in applications such as vehicular networks, industrial IoT, and 5G ultra-reliable low-latency communication. TSN can efficiently schedule real-time data streams. TSN traffic scheduling requires fast and precise algorithms; however, existing exact solution methods, due to their high computational complexity, struggle to meet the real-time requirements in large-scale joint scheduling scenarios. Genetic algorithms are suitable for addressing scheduling problems in TSN. The challenge lies in designing the genetic algorithm to find the optimal or near-optimal solution for various complex problems, while considering both performance and quality in the scheduling table. This paper proposes a priority-driven adaptive genetic algorithm for TSN traffic scheduling, which jointly considers routing and TSN constraints. The method employs a dual-vector encoding scheme, and prioritizes traffic characteristics by assigning priority scores to initialize the population. By combining a joint sequential crossover operator and a dynamic adaptive mutation strategy, the algorithm enhances both search capability and scheduling performance. Experimental results demonstrate that, compared to other genetic algorithm-based TSN traffic scheduling methods, the proposed method shows significant advantages in terms of convergence speed, task completion time for TSN traffic scheduling, and scheduling stability.

Keywords: Time-Sensitive Networking, Traffic scheduling, Genetic algorithm, Priority-driven

1 Introduction

In fields like industrial control and automotive electronics, the demand for real-time performance and security is increasingly growing. Traditional bus-based proprietary network technologies are gradually unable to meet these requirements in terms of communication bandwidth and efficiency. Against this backdrop, the IEEE Time-Sensitive Networking (TSN) task group [1] was established in 2012, formally initiating research on constructing data transmission services based on standard Ethernet that support high determinism and low latency. TSN employs a range of technical means, such as time synchronization, traffic priority scheduling, bandwidth allocation, and traffic shaping, to provide end-to-end deterministic transmission guarantees for different types of data flows. Unlike traditional best-effort services, TSN offers low latency, low jitter, and high reliability while maintaining compatibility with standard Ethernet, making it a crucial component of next-generation real-time Ethernet communication [2]. The development of TSN originated from the Audio-Video Bridging (AVB) protocol, which was initially aimed at providing real-time transmission assurance for audio and video streams. Based on this, the TSN task group further expanded the application scenarios of AVB, widely applying TSN technology in automotive Ethernet, industrial Internet, and automation control fields.

In the field of TSN research, traffic shaping and scheduling are considered core technologies for achieving deterministic network transmission. Traffic shaping primarily operates at the traffic layer by regulating the data transmission rate to prevent burst traffic from negatively impacting network performance. On the other hand, traffic scheduling is controlled at the packet level, precisely planning the transmission time of packets in switches to achieve deterministic data transmission latency and minimize jitter. However, the traffic scheduling problem is theoretically classified as an NP-complete problem [3], which means it has high algorithmic complexity. This is especially true in industrial Internet of Things (IIoT) scenarios with increasingly complex network structures and significantly different transmission requirements for various traffic types. Inefficient scheduling strategies can lead to wasted system resources and increase transmission latency and jitter, thereby undermining the advantages of TSN in high-precision traffic control.

Currently, the scheduling methods for TT traffic are primarily optimized based on the Time-Aware Shaper (TAS) mechanism. Under the TAS framework, traffic with different priorities is isolated through eight independent queues, and the transmission time slots for each queue are controlled by a Gate Control List (GCL), which ensures time isolation of data flows during transmission. The cyclical operation of the GCL guarantees that TT traffic is transmitted in high-priority channels according

*Corresponding Author: Yu Zhang; Email: zy02@mail.nwpu.edu.cn
DOI: https://doi.org/10.70003/160792642026032702008

to preset time slots, preventing interference from other traffic and thus achieving lower latency and jitter. The key challenge of the TAS mechanism lies in how to generate the Gate Control List. The generation process requires searching for the optimal low-latency solution that satisfies the traffic performance requirements under constraints such as link constraints, isolation requirements, and transmission time slots. However, research shows that no known algorithm can derive the optimal frame scheduling solution in polynomial time [4]. As a result, the scheduling of frame transmission time slots is mostly studied in offline environments. This challenge has led to significant academic interest in TT traffic scheduling methods and has driven further optimization research in this area.

To ensure the latency requirements of TT traffic, the current research focus is on continuously optimizing scheduling algorithms to reduce the end-to-end latency of TT traffic. Integer Linear Programming (ILP), as a mathematical optimization method, is widely used in TSN's frame transmission time slot scheduling problem because it can achieve optimal solutions for integer variables under given linear constraints. For instance, J. Min et al. [5] proposed an ILP-based scheduling method that models the traffic scheduling process as a multi-objective optimization problem, considering both the delay and queue occupancy of TT traffic to obtain optimal time slot configurations. E. Schweissguth et al. [6] introduced a resource-constrained approach, particularly suitable for multicast traffic scenarios. This method reduces the resource consumption of the algorithm model, decreases the running time of the ILP solver, and does so without affecting the quality of the solution.

Satisfiability Modulo Theories (SMT) and Optimization Modulo Theories (OMT) play an important role in solving TSN constraint optimization problems. SMT is used to verify the satisfiability of the constraint conditions, while OMT further optimizes the solution of the objective function based on this. Y. Chi et al. [7] combined task scheduling with TSN and formalized the problem as an SMT problem. Based on the solver's results, they constructed a flow-level scheduling model based on IEEE 802.1 Qbv. By properly decomposing the traffic, this method can effectively reduce constraint inequalities when traffic increases, significantly improving network utilization compared to traditional frame-based programming models. Y. Nakayama et al. [8] proposed a real-time adaptive gating scheduling scheme based on the Time-Aware Shaper (TAS), modeling the scheduling problem as a Boolean satisfiability problem (SAT) and achieving fast runtime computation via an FPGA-based solver.

While mathematical methods can formally describe traffic scheduling problems, they often involve high computational complexity. Heuristic algorithms provide an alternative approach, employing a greedy strategy: starting with an empty Gate Control List (GCL), the algorithm iteratively adds traffic flows by selecting the optimal solution at each step, with each new flow constrained by previously scheduled flows. The process terminates when new flows become unschedulable. Z. Feng et al. [9] developed a fault-tolerant heuristic algorithm enabling online incremental rerouting and rescheduling of interrupted flows while preserving existing flow paths and schedules. Evaluations demonstrate this method outperforms existing approaches in system redundancy and flow acceptance rate.

D. Bujosa et al. [10] proposed the HERMES heuristic algorithm, outperforming traditional constraint programming (CP) methods in runtime and network schedulability, particularly with multiple TT queues, while supporting dual jitter modes. A. Arestova et al. [11] developed a hybrid genetic algorithm integrating NEH, achieving comparable scheduling quality with significantly reduced computation time. H. Liu et al. [12] introduced a flow-sorting genetic algorithm that enhances scheduling capability while maintaining high network utilization. Although existing studies explore TSN scheduling from various perspectives, challenges persist including routing-scheduling decoupling and low computational efficiency.

In our paper, we propose a TSN traffic and routing joint scheduling method based on a priority-driven adaptive genetic algorithm, optimizing the completion time of time-sensitive traffic scheduling tasks. This method first designs a bidirectional encoding scheme tailored to the characteristics of joint scheduling and routing, representing the scheduling order of TT traffic and path allocation separately. The population initialization adopts a priority-layered strategy, dividing TT traffic into high, medium, and low priority flows based on deadlines, periods, and load characteristics, ensuring the quality of the initial population while increasing its diversity. For genetic operations, a two-stage selection strategy is designed, combining tournament selection and roulette wheel selection, balancing the quality and diversity of the population. The crossover operation effectively addresses the coupling problem between flow scheduling order and path allocation using a combined order crossover algorithm, ensuring the feasibility of offspring individuals. The mutation operation introduces a dynamic mutation rate adjustment strategy, adapting the mutation rate to achieve a balance between global search and local optimization, thereby improving the algorithm's convergence speed and global search ability.

2 System Model

This section systematically models the scheduling of TT traffic, including network model and communication model.

2.1 Network Model

The architectural model is an abstract representation of a physical TSN, which includes end systems, switches, and physical links. The TSN is abstractly modeled as an undirected graph $G = (V, E)$, where V represents the set of all nodes in the TSN, including both the sender-receiver terminal node set ES and the TSN switch node set BR. The set E denotes the directed link set, where each link $dp_{ij} \in E$ represents a directed connection from a source node to a destination node.

Figure 1 illustrates an example of a simple network topology, which contains 9 nodes and 9 links, consisting of 4 switches (S) and 5 terminals (H). The routing between two terminal nodes is represented by a set of possible paths. For example, r_i = {[ES_1, BR_1], [BR_1, BR_2], [BR_2, ES_3]} represents a path from terminal node ES_1 to terminal node ES_3. This model serves as a foundational abstraction for studying the TSN flow and routing scheduling, ensuring a clear structure for understanding and optimizing the performance of time-sensitive communications within the network.

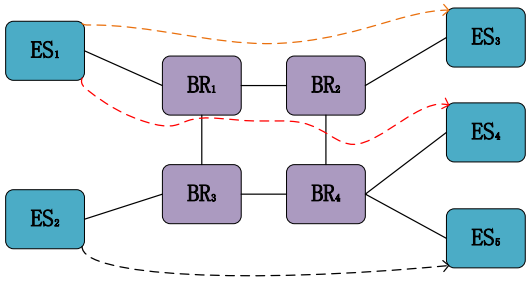

Figure 1. Examples of network topology

2.2 Communication Model

In Time-Sensitive Networking (TSN), the set of all periodic time-sensitive flows is defined as $F = \sum f_i$, where each Time-Triggered (TT) flow, containing multiple data frames, is defined by a tuple f_{TT} = ⟨src, dest, ϕ, S, T, J, D⟩ that encompasses the source node, destination node, transmission offset time of the flow, packet size, period of the flow, maximum jitter constraint, and maximum end-to-end delay constraint (or deadline) of the flow.

TT traffic transmission employs a Time-Aware Shaper (TAS) based on Time Division Multiple Access (TDMA). The Gating Control List (GCL) schedules transmission timing across 8 queues by dividing communication time into periodic slots. Each slot's gate state (0-closed/1-open) is defined by GCL, allocating deterministic transmission windows for high-priority traffic. Precise time offset control ensures end-to-end delays meet real-time requirements. The transmission example is shown in Figure 2.

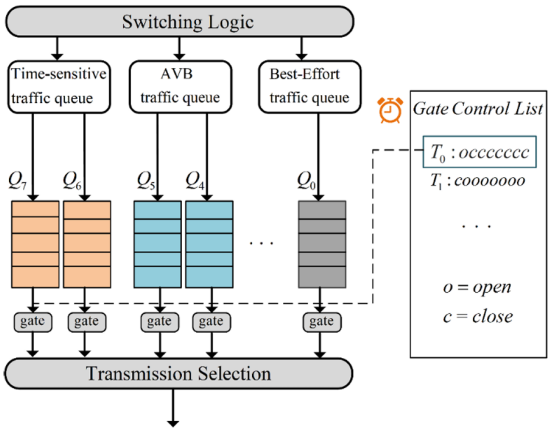

Figure 2. Transmission model

In the TSN network, the transmission of traffic is inevitably influenced by various processes such as link transmission, switch processing, queueing, and frame encapsulation, which result in certain end-to-end delays. These delays consist of propagation delay from the previous node to the current switch, processing delays at priority filtering units, queueing delays under different queues, and the final transmission delay. For TT traffic, once the Gating Control List (GCL) is determined, its specific delay can be accurately calculated based on predefined scheduling tables.

The end-to-end delay d_i for each flow is defined as the sum of the delays across all hops from the source node to the destination node.

$$d_i = \sum (d_{\text{Propagation}}, d_{\text{MaxBridge}}) \quad \textbf{(1)}$$

The propagation delay of a link, denoted as $d_{\text{Propagation}}$, represents the time required for a packet to travel across the link. It is determined by the transmission distance of the packet on the link and the signal transmission speed. For the sake of simplicity, it is assumed that the standard length of the links in the network is 20 meters, resulting in a propagation delay of approximately 0.067 microseconds. This delay contributes minimally to the overall latency.

$d_{\text{MaxBridge}}$ represents the time required by a switch to process and forward a packet in the worst-case scenario. This includes several components, as shown in formula (2).

$$d_{\text{MaxBridge}} = \sum (d_{\text{Transmission}}, d_{\text{Queueing}}, d_{\text{Processing}}) \quad \textbf{(2)}$$

The transmission delay $d_{\text{Transmission}}$ is the time required for data to enter the transmission medium from a node, calculated as shown in formula (3), where L_i is the size of the frame and B_{link} is the link bandwidth. In this paper, B_{link} is uniformly set to 1 Gbps.

$$d_{\text{Transmission}} = \frac{L_i}{B_{\text{link}}} \quad \textbf{(3)}$$

The queuing delay d_{Queueing} is caused by the competition for flow in the switch's output queue and depends on the priority of the frames in the queue and the queue scheduling algorithm.

The processing delay $d_{\text{Processing}}$ is the time required for the switch to parse and process the packet internally. This part is usually related to the hardware performance of the device and can typically be ignored.

3 Scheduling and Routing Constrains

This section defines the relevant constraints for TT traffic scheduling, including time scheduling constraints and routing constraints, to ensure the transmission and routing requirements of time sensitive traffic, respectively.

3.1 Scheduling Constrains

To ensure that the calculated GCL meets the determin-

istic low-latency requirements of TSN, i.e., time-sensitive flows are transmitted with ultra-low latency without link contention within their transmission cycle, the scheduling algorithm must satisfy the following constraints.

The offset of TT traffic determines the specific transmission time for each frame within the scheduling period. To ensure that the transmission of data frames conforms to the requirements of the scheduling period, the offset must satisfy several constraint conditions. Firstly, the offset cannot precede the starting moment of the scheduling period, thereby guaranteeing that the transmission time of data flows does not exceed the scheduling scope, which could otherwise lead to resource scheduling conflicts. Additionally, the offset must be allocated within the scheduling period and sufficient time must be reserved for the frame's transmission, meaning that the frame's offset should not exceed the remaining time in the period minus the transmission duration of the frame. The specific mathematical expression is shown as follows. Here, $f_{ij}^{[v_a,v_b]}.\phi, f_{ij}^{[v_a,v_b]}.T$, and $f_{ij}^{[v_a,v_b]}.L$ represent the offset of the j-th frame of the TT flow f_i on the link $[v_a,v_b]$, the scheduling cycle of the flow, and the transmission delay, respectively.

$$\begin{aligned} &\forall f_i \in f_{TT}, \forall j \in \{1,2,\ldots,\frac{f_i.S}{\mathrm{MTU}}\}: \\ &\left(f_{ij}^{[v_a,v_b]}.\phi \geq 0\right) \wedge \left(f_{ij}^{[v_a,v_b]}.\phi \leq f_{ij}^{[v_a,v_b]}.T - f_{ij}^{[v_a,v_b]}.L\right) \end{aligned} \tag{4}$$

In TSN, the periodic communication requirements of traffic impose strict control over transmission jitter. Jitter reflects the stability of delay during the transmission of data frames. To simplify complexity, jitter constraints are primarily imposed on the sending node of the last hop before the traffic reaches the receiver, thereby reducing the complexity across the entire network. For a data frame j of a TT traffic flow f_i, its arrival time $W_{i,j}$ can be calculated using the frame's offset ϕ, link transmission delay L, and the periodic offset between frames. Based on this, the jitter constraint for the flow must ensure that the transmission time difference $W_{i,x} - W_{i,y}$ between any two frames within the same flow does not exceed the maximum jitter limit $j_{\max}$ of the flow.

$$\begin{aligned} &W_{i,j} = f_{i,j}^{[v_{n-1},v_n]}.\phi + f_{i,j}^{[v_{n-1},v_n]}.L - f_{i,j}^{[v_1,v_2]}.\phi \\ &\forall f_{i,x}, f_{i,y} \in f_i : W_{i,x} - W_{i,y} \leq f_i.j_{\max} \end{aligned} \tag{5}$$

For the same terminal device, all TT traffic must be transmitted according to its defined transmission cycle to ensure the consistency of flow scheduling and transmission determinism. To meet this requirement, the system must ensure that the offset time for each frame within the same flow remains consistent within the cycle. Additionally, for different flows, the transmission time interval between frames should align with their periodic characteristics. This ensures that each data frame of the flow maintains periodic consistency in its transmission time within the scheduling cycle, meaning that the offset time for frames in the same flow remains fixed between adjacent cycles.

$$\begin{aligned} &\forall f_i \in f_{TT}, \forall j \in \{1,2,\ldots,\frac{f_i.S}{MTU}\}: \\ &f_{i,j}^{[v_a,v_b]}.\phi = f_{i,j+\left\lfloor \frac{f_i.S}{MTU} \right\rfloor}^{[v_a,v_b]}.\phi \end{aligned} \tag{6}$$

To meet the requirements for real-time and deterministic transmission, the end-to-end delay of TT traffic must be strictly controlled within a defined range. Specifically, the total time for a data frame to travel from the source node to the receiver node must not exceed its predefined maximum end-to-end delay limit. If the end-to-end delay exceeds this limit, it may result in a decline in real-time service quality or even cause a communication failure within the system.

$$\begin{aligned} &\forall f_i \in f_{TT}, \forall j \in \left\{1,2,\ldots,\frac{f_i.S}{MTU}\right\}: \\ &f_{i,j}^{[v_{n-1},v_n]}.\phi + f_{i,j}^{[v_{n-1},v_n]}.L - f_{i,j}^{[v_1,v_2]}.\phi \leq f_i.D_{\max} - \theta \end{aligned} \tag{7}$$

The transmission of TT traffic across different links must strictly follow the predetermined physical sequence to ensure consistency between the frame transmission logic and the time scheduling. This means that the transmission start time of a data frame on the subsequent link must be greater than or equal to the transmission end time on the previous link. Additionally, the switch node can only process and forward the data to the next link after receiving the data from the previous link. Here, $D_{[v_a,v_x]}$ represents the transmission delay between switches v_a and v_x, including the time required for the data frame to travel through the link from the sender to the receiver.

$$\begin{aligned} &\forall f_i \in f_{TT}, \forall j \in \left\{1,2,\ldots,\frac{f_i.S}{MTU}\right\}: \\ &f_{i,j}^{[v_a,v_x]}.\phi + D_{[v_a,v_x]} + \theta \leq f_{i,j}^{[v_x,v_b]}.\phi \end{aligned} \tag{8}$$

In TSN, to avoid congestion and conflicts when multiple flows arrive simultaneously at the same queue in a switch node, frame isolation control must be applied to the data flow transmission. Frame isolation can be achieved in two ways: temporal isolation and queue isolation. Temporal isolation ensures that only one data flow can be stored in the same queue at any given time by controlling the arrival times of different data flows. Queue isolation allocates separate queues for different data flows to prevent conflicts when sharing the same queue resources. Here, $f_i^{[v_a,v_b]}.queue$ represents the queue allocated for flow f_i on link $[v_a,v_b]$, while $[v_x,v_a]$ and $[v_y,v_a]$ refer to the two non-fixed links before link $[v_a,v_b]$.

$$\forall f_i, f_j \in f_{TT}, \alpha \in \left[0, \frac{\text{hp}}{f_i.T} - 1\right], \beta \in \left[0, \frac{\text{hp}}{f_j.T} - 1\right], i \neq j:$$
$$\left(f_{i,l}^{[v_x, v_a]}.\phi + \alpha \cdot f_i.T + f_{i,j}^{[v_x, v_a]}.L + d_{proc}^{va} + \theta \geq f_{j,k}^{[v_a, v_b]}.\phi + \beta \cdot f_j.T\right)$$
$$\vee\left(f_{j,k}^{[v_y, v_a]}.\phi + \beta \cdot f_j.T + f_{i,j}^{[v_y, v_a]}.L + d_{proc}^{va} + \theta \geq f_{i,l}^{[v_a, v_b]}.\phi + \alpha \cdot f_i.T\right)$$
$$\vee\left(f_i^{[v_a, v_b]}.queue \neq f_j^{[v_a, v_b]}.queue\right) \quad \textbf{(9)}$$

3.2 Routing Constrains

In TSN traffic scheduling, routing constraints refer to the conditions that must be met during the transmission process. These constraints typically include the following aspects.

The transmission path of a data frame must satisfy the constraints of the network topology. Specifically, in the transmission path of each flow f_i, the next-hop target node $v_{i,a}$ of any node v_a must be a node v_b directly connected to v_a.

$$\forall r_i \in R, \forall f_i \in F, \quad [v_a, v_b] \notin r_i \Rightarrow v_{ij,a} \neq v_b \quad \textbf{(10)}$$

To avoid issues such as duplicate transmission and deadlock caused by path loops during network transmission, the routing plan in Time-Sensitive Networks must satisfy the acyclic constraint. Specifically, for the transmission path of each flow, the target node of any two nodes must be distinct to ensure that there are no loops in the transmission path.

$$\forall r_i \in R, \forall [v_a, v_b] \in r_i, \forall f_i \in F, v_a \neq v_b \Rightarrow v_{i,a} \neq v_{i,b} \quad \textbf{(11)}$$

The path integrity constraint ensures that time-triggered flows can reach the target node from the source node in the network and find a complete and valid connection path. Suppose the target node of flow f_i at node v_a is $v_{i,a} = v_b$, and v_b is not a terminal node. In that case, there must exist a node v_x connected to v_b and an associated link $[v_b, v_x]$.

$$\forall r_i \in R, \forall f_i \in F$$
$$(v_{i,a} = v_b) \wedge (v_b \neq v_{\text{dst}}) \Rightarrow (v_{i,b} \neq \text{null}) \wedge ([v_a, v_b] \in r_i) \quad \textbf{(12)}$$

4 Priority-driven Adaptive Genetic Algorithm-based Scheduling for Time-sensitive Traffic

Based on the characteristics of TSN network scheduling, for TT traffic scheduling, we propose a Priority Driven Adaptive Genetic Algorithm (PDAGA) to address the joint optimization problem of TT traffic routing and scheduling in Time-Sensitive Networks. The corresponding encoding method, decoding rules, population initialization, as well as crossover and mutation operators are designed, with the goal of minimizing the total makespan of all traffic flows, i.e., to make the latest completion time of all TT traffic flows as early as possible.

$$\min\left(\max_{\forall f_j \in F}\left(f_j^{[v_{n-1}, v_n]} \cdot \Phi + f_j^{[v_{n-1}, v_n]}.L\right)\right) \quad \textbf{(13)}$$

The individual in PDAGA represents a scheduling solution, which can also be mapped to a Gate Control List (GCL) within a hypercycle. Based on the scheduling constraints, each individual can calculate a fitness value to describe its quality level, which is represented by the makespan. To generate new individuals, PDAGA provides crossover and mutation methods that do not rely on external libraries, which are used to pair and evolve the gene sequences within the population.

4.1 Encoding and Decoding

Genetic encoding maps scheduling solutions into GA-compatible structures by representing flow scheduling sequences and path selections meeting TSN's deterministic requirements. The chromosome comprises two vectors for joint optimization: flow ordering and path assignment.

The flow ordering vector determines the global scheduling order of each flow. For example, the encoding in Figure 3 shows flow 1 is scheduled first, followed by flows 2, 3, 4, and 5. Different orderings generate distinct scheduling solutions, and adjusting flow sequences explores better results.

Each gene in the path assignment vector corresponds to flow path selection. Each TT flow with at least one path has predefined path sets with unique identifiers. The vector combines selected identifiers from each flow's path set, maintaining equal length with the flow ordering vector. For instance in Figure 3, flow1 selects path2 while flow3 chooses path1.

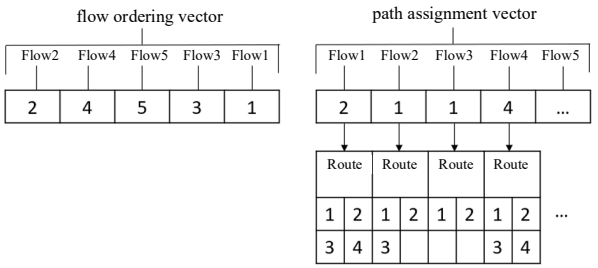

Figure 3. Representation of gene encoding

The genetic algorithm decoding transforms individual gene sequences into TSN scheduling solutions while ensuring TSN constraints. The total TT traffic transmission time is calculated as the end time of the last TT frame minus the start time of the first. The main steps are as follows:

(1) Decode flow order and path assignment. Each chromosome includes a flow ordering vector and path assignment vector. During decoding, genetic encoding maps this data to flow sequences and routes by extracting scheduling orders and assigning paths.

(2) Iterative time window allocation. Each flow is processed in the order specified by the flow ordering vector, with a feasible transmission time window allocated for it. The allocation incorporates routing information and considers constraints such as delay and resource conflicts, ensuring each flow is assigned an available time window within its constraint limits.

(3) Greedy scheduling. A greedy strategy assigns the earliest possible start time for each flow, selecting the first valid time point in the candidate window that meets offset, jitter, and resource constraints. Overlaps on shared links delay transmission, and subsequent flows start immediately after the previous flow ends to ensure compact and deterministic scheduling.

(4) Feasibility verification. After time window allocation, the schedule undergoes validation where each flow and link is checked for constraint compliance. Non-compliant solutions are marked invalid while valid allocations are retained.

(5) Scheduling output. The decoding process generates a complete schedule containing TT traffic flows' start/end times and network paths. This output is used for fitness evaluation to measure solution quality.

4.2 Population Initialization

The global search efficiency and optimization convergence speed of a genetic algorithm largely depend on the method of population initialization [13]. A reasonable population initialization method can improve the quality of the initial population, enhance the diversity of the population, thus accelerating the convergence of the algorithm and leading to better solutions. Considering the characteristics of time-sensitive traffic, this paper proposes a population initialization method based on flow priority hierarchy. This method classifies flows according to priority and applies different initialization strategies for flows with different priorities, ensuring the scheduling quality of critical flows. At the same time, to maintain population diversity, the method combines random generation to initialize the population, balancing the quality and diversity of the initial population.

The steps for generating individuals based on the flow priority hierarchy are as follows.

To accurately assess the potential impact of each flow on completion time in determining priority indicators, we have selected three key parameters as the foundation for priority scoring: deadline D_i, which reflects the importance and urgency of a flow and is a crucial factor in flow scheduling; packet size S_i, a significant influence on scheduling decisions as larger data volumes often necessitate more network resources; and period T_i, where short-period TT traffic signifies higher transmission frequency demands, potentially requiring greater sustained use of network resources.

To comprehensively consider the impact of each flow on scheduling, we define a priority score for flows, taking into account factors such as deadline, load, and periodicity. To eliminate the influence of different dimensions, normalization is required, and in this paper, we adopt the min-max normalization method.

$$P_i = \alpha \times \frac{f_i.D_i - D_{\min}}{D_{\max} - D_{\min}} + \beta \times \frac{f_i.S_i - S_{\min}}{S_{\max} - S_{\min}} + \gamma \times \frac{f_i.T_i - T_{\min}}{T_{\max} - T_{\min}} \tag{14}$$

Subsequently, based on the calculated priority scores, all flows are sorted in ascending order and categorized into different priority levels according to the following proportions: high-priority flows, comprising the top 30% of flows, for which scheduling quality is prioritized to minimize the impact on completion time; medium-priority flows, encompassing the middle 50% of flows, which balance scheduling quality and population diversity; and low-priority flows, representing the bottom 20% of flows, primarily used to enhance population diversity and avoid converging to local optima.

Adopt different optimization strategies for flows of varying priority levels.

(1) Initialization of high-priority flows:

Heuristic methods are commonly employed to enrich the initial population with one or more promising individuals. One of the most frequently mentioned heuristic algorithms in population initialization is the NEH algorithm. The NEH algorithm was developed to address a specific problem within the Job Shop Scheduling Problem (JSP): the permutation flow shop scheduling problem, with the objective of minimizing the makespan. The NEH algorithm has proven effective in numerous studies [14]. However, for TSN traffic scheduling, a drawback of NEH is that each insertion of a flow requires trying all possible positions, leading to a significant increase in computational cost when the number of flows is large. Therefore, combining the characteristics of TT traffic, for high-priority flows, we adopt a heuristic algorithm along with a path selection strategy based on path scores for initialization, to ensure that these flows, which have the greatest impact on completion time, are optimized.

Firstly, sort the flows according to their priority scores to obtain an initial flow sequence. Simultaneously, for each flow, generate a set of candidate paths by considering link load and hop count.

Place the flow with the highest priority into the scheduling sequence, and for the remaining flows, insert them sequentially into the current scheduling sequence. During each insertion, try all possible insertion positions and calculate the scheduling result for each position. For each insertion position, attempt all candidate paths and compute the makespan after insertion, while ensuring that the scheduling after insertion satisfies the constraints. Finally, from all possible insertion positions, select the one that results in the minimum makespan and satisfies the constraints. Update the scheduling sequence and candidate path information.

Repeat the above process until all high-priority flows have been inserted into the scheduling sequence, generating a feasible solution.

During the initialization of high-priority flows, it is crucial to select the optimal path for each flow. Currently, most traffic scheduling mechanisms typically use shortest path routing to determine the routing scheme. However, as

indicated in reference [15], using the shortest path alone cannot achieve optimal results. Therefore, considering link load balancing, a method for generating a candidate path set for flows that combines link load and hop count is proposed. The specific steps are as follows.

Obtain a set of feasible paths. For each high-priority flow, use the network topology and routing algorithm to find all feasible paths that satisfy the routing constraints.

Calculate the score for each path. In this calculation, $L'_{\max}$ stands for the maximum link load value after being adjusted by an introduced amplification coefficient, while M represents the total number of hops on the path, which is equivalent to the number of links traversed by the path. The weight coefficient is designated as H_{total}. Additionally, M is a coefficient introduced to balance the influence of each component in the scoring formula and is set to the average number of hops in the network.

$$\begin{aligned} Score_{path} &= \delta \times L'_{\max} + \epsilon \times H_{total} \\ L'_{\max} &= M \times L_{\max} \end{aligned} \tag{15}$$

$L_{\max}$ represents the maximum link load of a path, which is the maximum value among all link loads on each path. L_{link} refers to the link load. n is the number of links on the path. Among them, $B^{(l)}_{used}$ represents the bandwidth currently occupied by other flows on link l, $B_{current}$ is the bandwidth required by the current high-priority flow, and $B^{(l)}_{total}$ is the total bandwidth capacity of link l.

$$\begin{aligned} L_{\max} &= \max\left\{L^{(1)}_{link}, L^{(2)}_{link}, \ldots, L^{(n)}_{link}\right\} \\ L_{link} &= \frac{B^{(l)}_{used} + B_{current}}{B^{(l)}_{total}} \end{aligned} \tag{16}$$

Finally, compare all feasible paths and select the top K paths with the lowest scores as the transmission path set for the flow. As the link loads change, the set of feasible paths will dynamically change accordingly.

(2) Initialization of Medium-Priority Flows:

For medium-priority flows, three strategies generate scheduling schemes to balance quality and diversity: deadline-based sorting (earliest first), cycle-based sorting (shortest first), and randomized ordering. All generated sequences are retained to enrich the population.

Path allocation dynamically selects conflict-free paths with sufficient resources based on high-priority flow occupation.

(3) Initialization of Low-Priority Flows:

The initialization of low-priority flows aims to increase population diversity and reduce algorithm resource/computation time consumption. Under resource and delay constraints, their scheduling order is randomly generated to maximize initial solution diversity. Path selection uses randomization with heuristics: surplus resource paths are prioritized to avoid conflicts with higher-priority flows. If unavailable, the least occupied alternative path is selected.

After initializing different priority flows, their scheduling schemes are combined into a complete initial individual. Prioritize stable scheduling for high-priority flows, then integrate medium-priority flows without affecting them, followed by low-priority flows. Verify the merged scheme meets global constraints, discarding non-compliant individuals due to resource conflicts or constraint violations.

4.3 Selection

In genetic algorithms, the selection operation refers to picking individuals with higher fitness or better performance from the current population as parent individuals according to specific criteria or evaluation standards, in order to provide genetic information for the generation of offspring. Traditional roulette wheel selection and tournament selection each have their own advantages and disadvantages. To combine the strengths of both, this paper proposes a selection operator based on a combination of roulette wheel and tournament selection. This method maintains the diversity of the population while retaining excellent individuals, thereby improving the performance of the algorithm.

Combining the advantages of tournament selection and roulette wheel selection, a two-phase selection mechanism is adopted. In the first phase, also known as the pre-selection phase, tournament selection is used to screen out a subset of individuals with relatively superior performance from the current population to form a candidate pool. Tournament selection provides moderate selection pressure in this process, enabling the rapid and effective elimination of individuals with poorer fitness and ensuring that the individuals in the candidate pool achieve higher adaptability within a shorter period of time. In the second phase, roulette wheel selection is employed to further select the final parent individuals from the candidate pool. Roulette wheel selection allocates selection probabilities based on the fitness of each individual, such that individuals with higher fitness have a greater probability of being selected, thereby ensuring the inheritance of superior genes. The specific steps are as follows:

(1) Fitness Calculation: For each individual i in the population, calculate its fitness value F_i.

(2) Tournament Pre-selection: Set the tournament size T and the pre-selection pool size N_{pre}. The pre-selection process involves repeating the following steps N_{pre} times: Randomly select T individuals from the population to form a tournament set, select the individual with the highest fitness, and add it to the pre-selection pool.

(3) Calculation of the total fitness in the pre-selection pool, probability of individual selection, and cumulative probability calculation:

$$F_{total} = \sum_{i=1}^{N_{pre}} F_i,\ p_i = \frac{F_i}{F_{total}},\ q_i = \sum_{j=1}^{i} p_j \tag{17}$$

Given the number of parent individuals $N_{parents}$ that need to be selected, repeat the following operations $N_{parents}$

times: Generate a random number $r \in [0,1]$, find the first individual that satisfies condition $q_i \geq r$, and select it into the parent set.

(4) Elite Preservation Strategy. Set the number of elites to be preserved, denoted as E, at 5% to 10% of the population size. Sort the individuals based on their fitness and directly copy the top E optimal individuals to the next generation. Perform crossover and mutation on the parent individuals selected in the previous step. Ultimately, the composition of the new population consists of the directly preserved E elite individuals and the new $N-E$ ordinary offspring individuals generated through crossover and mutation from the parent set.

4.4 Crossover Operators

In genetic algorithms, crossover combines parental genes to create offspring. This paper proposes a Combined Order Crossover (COX) operator for bidirectional vector encodings. COX simultaneously processes flow sequencing and path allocation vectors: applying Order Crossover to generate new sequences while preserving parental gene order, and inheriting corresponding parental paths based on the new sequence to maintain flow-path correspondence.

For the path allocation vector, in offspring 1: for flows inherited from parent 1 in the scheduling sequence, their path selections are inherited from parent 1; for other non-inherited flows, the path with the lowest score from the feasible path set is selected. In offspring 2: for flows inherited from parent 2 in the scheduling sequence, their path selections are inherited from parent 2; for other non-inherited flows, the path with the lowest score from the feasible path set is selected. As shown in Figure 4, flows 3, 1, and 5 in offspring O1 are inherited from parent P1, so their path selections are inherited from P1's path selections—flow 1 selects path 2, flow 3 selects path 1, and flow 5 selects path 2. For other flows, such as flow 2, the path with the lowest score from the feasible path set, which is path 1, is selected.

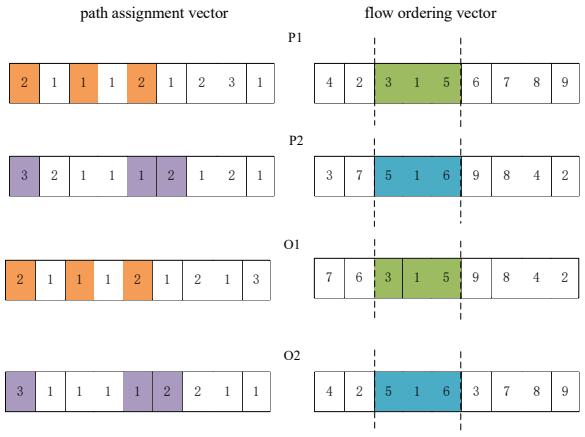

Figure 4. Example of COX

4.5 Mutation

In genetic algorithms, mutation enhances population diversity through random gene modifications, preventing local optima entrapment. Given our dual-vector encoding scheme, we design specialized mutation operators. For the permutation-based flow sequencing vector requiring element uniqueness, we apply swap mutation by exchanging two randomly selected genes. For the path allocation vector where elements represent flow-specific path numbers with varying feasible sets, we employ random reset mutation: randomly selecting a position and replacing it with another valid path from that flow's available set.

While basic mutation operators are crucial for maintaining population diversity and exploring new solution spaces, genetic algorithms still face evolutionary stagnation and local optima issues. This paper proposes a flow priority-stratified adaptive mutation strategy that dynamically adjusts mutation rates, enhancing algorithmic adaptability and robustness while ensuring high-priority flow scheduling quality.

This strategy dynamically adjusts the global mutation rate based on population diversity and fitness growth. Increased mutation rates enhance exploration when diversity is low or fitness stagnates. Moderate rates maintain exploration-exploitation balance at intermediate states. Reduced rates improve exploitation when diversity is high or fitness improves rapidly. This mechanism enhances global search efficiency and convergence performance.

Building on global mutation rate adjustments, this strategy sets differentiated local rates: lower for high-priority flows to stabilize scheduling, moderate for medium-priority to balance efficiency, and higher for low-priority to enhance diversity.

To implement the aforementioned design ideas, this paper proposes the following mutation rate adjustment strategy:

(1) Dynamic Adjustment of Global Mutation Rate. Based on the population diversity index (a) and fitness growth rate (b), the following adaptive mutation rate adjustment strategy is designed:

$$P_m^{global} = \begin{cases} P_m^{\max}, & \sigma < \sigma_{low} \vee \Delta\overline{f} < \Delta\overline{f}_{low} \\ P_m^{init}, & \sigma_{low} \leq \sigma \leq \sigma_{high} \wedge \Delta\overline{f}_{low} \leq \Delta\overline{f} \leq \Delta\overline{f}_{high} \\ P_m^{\min}, & \sigma > \sigma_{high} \vee \Delta\overline{f} > \Delta\overline{f}_{high} \end{cases} \quad \textbf{(18)}$$

The parameters are defined as follows: P_m^{init} represents the initial mutation rate, $P_m^{\max}$ is the maximum mutation rate, and $P_m^{\min}$ is the minimum mutation rate. Additionally, σ_{low} and σ_{high} denote the lower and upper thresholds for population diversity, respectively, while $\Delta\overline{f}_{low}$ and $\Delta\overline{f}_{high}$ represent the lower and upper thresholds for fitness growth rate, respectively.

When population diversity is low or the fitness growth rate is slow, increase the mutation rate to $P_m^{\max}$ to enhance the population's exploration capability. When population diversity and fitness growth rate are at moderate levels, maintain the initial mutation rate P_m^{init} to balance exploration and exploitation. When population diversity is high or the fitness growth rate is high, decrease the mutation rate to $P_m^{\min}$ to improve the algorithm's exploitation capability.

(2) Local Mutation Rate Setting Based on Priority Stratification. After determining the global mutation rate P_m^{global}, streams are classified into three levels based on their priority: high, medium, and low. Corresponding local mutation rates P_m^{high}, P_m^{medium}, and P_m^{low} are then set for streams of different priority levels.

$$\begin{aligned} P_m^{high} &= P_m^{global} \times \alpha \\ P_m^{medium} &= P_m^{global} \times \beta \\ P_m^{low} &= P_m^{global} \times \gamma \end{aligned} \tag{19}$$

Based on the aforementioned strategy, the specific implementation steps of the adaptive mutation rate strategy based on priority stratification proposed in this paper are as follows:

(1) Calculate the population diversity index. Calculate the standard deviation of the fitness values of all individuals in the current population to assess population diversity. Here, N represents the population size, f_i represents the fitness value of the i-th individual, and $\bar{f}$ represents the average fitness value of the population.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f_i - \bar{f})^2} \tag{20}$$

(2) Calculate the fitness growth rate $\Delta\bar{f}$. Calculate the fitness growth rate by comparing the average fitness $\bar{f}_{\text{current}}$ of the current generation's population with the average fitness $\bar{f}_{\text{previous}}$ of the previous generation.

$$\Delta\bar{f} = \bar{f}_{\text{current}} - \bar{f}_{\text{previous}} \tag{21}$$

(3) Adjust the global mutation rate P_m^{global}. Apply the aforementioned global mutation rate adjustment strategy based on the values of σ and $\Delta\bar{f}$ to determine the global mutation rate P_m^{global} for the current generation.

(4) Set local mutation rates based on priority. Using the global mutation rate P_m^{global} and combining the priority stratification of streams, calculate the local mutation rates P_m^{high}, P_m^{medium}, and P_m^{low} for streams of different priority levels.

(5) Apply mutation operations. For an individual's stream sorting vector and path allocation vector, decide whether to apply mutation based on their priority stratification and corresponding mutation rates.

(6) Feasibility check. After the mutation operation, perform a feasibility check on the mutated individual to ensure it meets the TSN scheduling constraints. If it does not meet the constraint requirements, discard the individual's gene.

The aforementioned mutation strategy primarily enhances the flexibility and performance of the algorithm through dynamic adaptability. Specifically, by monitoring the population diversity index and fitness growth rate, the mutation rate can be adaptively adjusted according to the evolutionary state. This mechanism improves the algorithm's performance at different stages and avoids the limitations that may arise from a fixed mutation rate. Additionally, the priority protection strategy is fully utilized.

5 Experiment Validation

This section conducts simulation verification of the performance of the proposed PDAGA algorithm in various network topology environments and a typical environment, and compares it with other algorithms to evaluate the advantages and disadvantages of the algorithm proposed in this paper.

5.1 Experiment Detailed Parameter Setting

This paper employs the OMNeT++ and NeSTiNg frameworks to set up a simulation environment, running on a Windows computer equipped with an Intel(R) Core(TM) i9-14900 2.40 GHz CPU and 32GB of RAM, to validate the TSN traffic scheduling method based on a genetic algorithm.

Current research lacks a standardized dataset for time-sensitive network traffic scheduling. Therefore, this section generates periodic time-sensitive traffic data randomly based on simulation scenarios. Source and destination nodes are selected from distinct terminal nodes to ensure traffic randomness and experimental generality. Packet payload size ranges from 100B to 2000B, with transmission periods of 50-500μs. To ensure scheduling feasibility, each traffic's deadline is set shorter than its transmission period.

To verify the performance of the proposed PDAGA-based time-sensitive traffic scheduling method in terms of runtime and makespan, this section selects the traditional genetic algorithm (TGA) from literature [16], the hybrid genetic algorithm (HGA) combined with the constrained NEH method from literature [11], and the flow sorting genetic algorithm (FSGA) from literature [12] as comparison algorithms. Through comparative analysis with these algorithms, a comprehensive evaluation of the advantages and applicability of PDAGA-based time-sensitive traffic scheduling across different scheduling performance indicators can be conducted.

Due to the stochastic nature of genetic algorithms, there may be some fluctuations in the simulation experimental results. To ensure the accuracy and reliability of the data, each set of experiments was repeated 10 times, and the average values were taken as the final experimental data to reduce the impact of randomness from a single experiment. The specific parameters of the genetic algorithm were set as follows: the population size was 100, the maximum number of iterations was 200, the crossover probability was 0.7, and the mutation probability had an initial mutation rate P_m^{init} of 0.1, a maximum mutation rate $P_m^{\max}$ of 0.3, and a minimum mutation rate P_m^{init} of 0.01. The mutation rate in TGA and HGA was fixed at 0.1. The priority parameters α, β, and γ were set to 0.6, 0.3, and 0.1, respectively.

5.2 Experimental Results under Various Network Topologies

In previous related studies, TSN scheduling experiments were mostly based on typical topologies found in industrial control networks, such as linear, ring, and star topologies [17-18]. To comprehensively evaluate the performance of the proposed method under different network topologies, this paper first draws inspiration from the topology generation approach in literature [19] by randomly generating Barabási-Albert (BA) networks [20] and Erdős–Rényi (ER) networks [21] with different characteristics to construct simulation scenarios. The specific basic parameters of the network topologies are shown in Table 1.

Table 1. Topological parameters

Topology scene	Number of exchanged nodes	Number of terminal nodes	Number of link nodes
ER1	5	5	13
ER2	12	8	25
ER3	20	10	55
BA1	6	4	16
BA2	12	8	36
BA3	20	10	62

To comprehensively evaluate the performance of the algorithms under different traffic loads and network sizes, this experiment conducted scheduling tests for three load conditions: 10 flows, 20 flows, and 40 flows. The makespan data for each algorithm in different scenarios were obtained (as shown in Figure 5, Figure 6, and Figure 7).

Figure 5 displays the makespan results for the scenario with 10 flows. It can be observed that across all network topologies, the method using PDAGA for scheduling time-sensitive traffic consistently exhibits the lowest makespan, significantly outperforming the scheduling methods using TGA, HGA, and FSGA. For example, in the ER3 scenario, the makespan under PDAGA is 73μs, while it is 99μs, 92μs, and 90μs for TGA, HGA, and FSGA, respectively. As the network size increases, the makespan of TGA, HGA, and FSGA grows significantly, whereas the growth rate of PDAGA is more gradual, demonstrating better scalability.

Figure 6 presents the experimental results for the scenario with 20 flows, where scheduling using PDAGA continues to maintain the lowest makespan across all scenarios. For instance, in the BA3 scenario, the makespan of PDAGA is approximately 79μs, significantly lower than that of TGA (151μs), HGA (111μs), and FSGA (117μs). Comparing Figure 6 with Figure 5, it can be observed that as the traffic increases, the performance degradation of the traditional algorithm TGA is more pronounced, while the performance of PDAGA remains relatively stable. In particular, PDAGA demonstrates a more significant advantage in larger-scale networks such as ER3 and BA3.

Figure 7 displays the experimental results for the scenario with 40 flows. Under high traffic conditions, the performance advantage of scheduling using PDAGA becomes even more prominent. In the ER3 and BA3 scenarios, the makespan of TGA, HGA, and FSGA all exceed 160μs, while PDAGA manages to stay around 120μs, reducing the scheduling time by approximately 30%. Especially in the complex topology BA3, PDAGA continues to perform exceptionally well under high loads, indicating that its algorithm design is better suited for complex network structures and high traffic conditions.

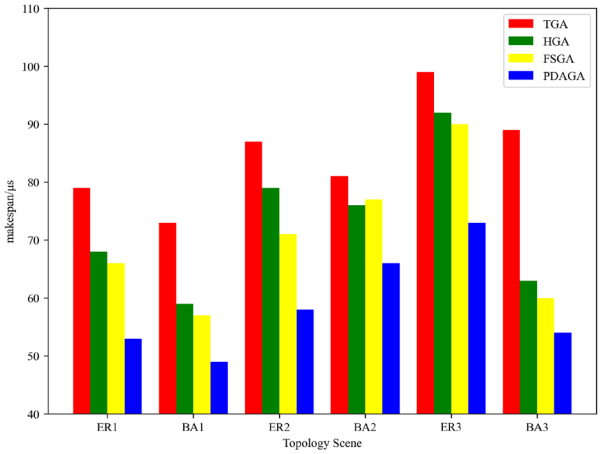

Figure 5. Comparison of makespans under 10 flows

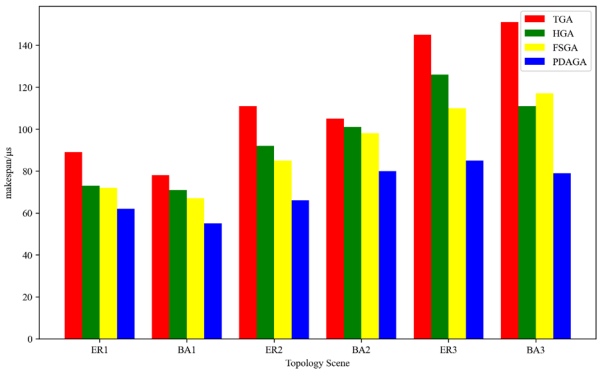

Figure 6. Comparison of makespans under 20 flows

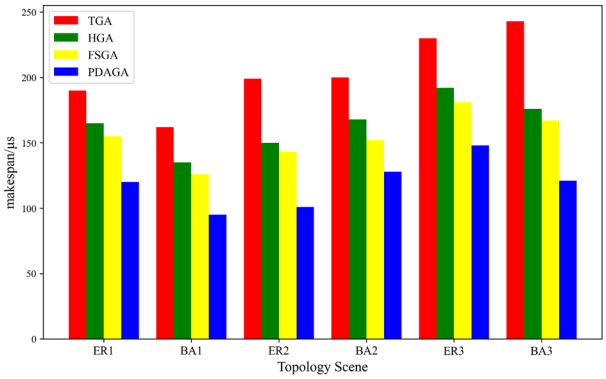

Figure 7. Comparison of makespans under 40 flows

Overall, traffic scheduling using PDAGA demonstrates the lowest makespan across all scenarios, particularly in complex networks and high traffic scenarios, where it outperforms the other three algorithms. This suggests

that PDAGA has advantages in search efficiency and scheduling optimization capabilities. In contrast, when the topology size increases or the traffic load rises, the growth rate of the makespan is more gradual for scheduling using PDAGA. This is attributed to PDAGA's dynamic mutation rate adjustment mechanism and priority hierarchy strategy, which effectively avoid premature convergence and local optimum issues, enhancing its global search capability. By employing a priority hierarchy mechanism and a dynamic mutation rate strategy, PDAGA improves scheduling efficiency and algorithm robustness, validating its superiority in the context of TT traffic scheduling problems.

In the BA3 scenario with 40 flows, to verify the superiority of the PDAGA algorithm, the convergence trends of the makespan during the iteration process for the four algorithms—TGA, HGA, FGA, and PDAGA—were recorded, as shown in Figure 8. This experiment aims to demonstrate the global search capabilities and convergence speeds of the four algorithms in a high-load complex network environment.

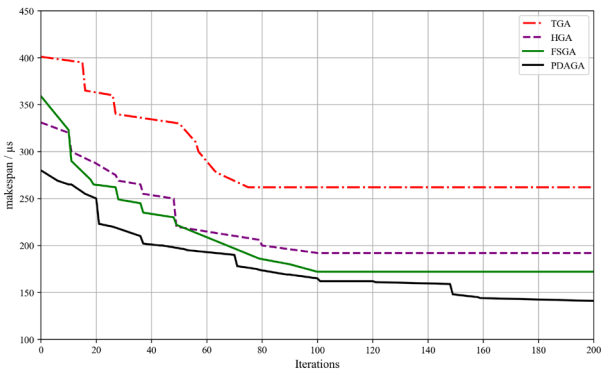

Figure 8. Comparison of convergence results of different algorithms

From the results, it can be observed that during the initial iteration stage (0 to 20 generations), all four algorithms exhibited a rapid decline, with PDAGA showing a more significant decrease. The makespan of PDAGA rapidly decreased from approximately 280μs to around 230μs within the first 20 iterations, demonstrating strong global search capabilities. After 60 generations, the convergence speed of TGA significantly slowed down, with the makespan tending towards stagnation and stabilizing around 270μs. HGA and FSGA continued to decline slowly at this stage, eventually stabilizing at 190μs and 170μs, respectively, by the 100th generation. PDAGA maintained a stable evolutionary trend during this stage, with the makespan dropping to around 160μs. Its dynamic mutation rate adjustment and priority hierarchy mechanism effectively avoided premature convergence issues. In the later stage from 100 to 200 generations, the makespan of HGA and FSGA basically remained unchanged, while PDAGA continued to optimize after the 100th generation, ultimately reducing the makespan to 140μs by the 200th generation, which is lower than the other three algorithms. This indicates that PDAGA has a better balance between local optimization and global exploration capabilities.

5.3 Experimental Results in Typical Environments

To more accurately verify the effectiveness of the proposed method, this section selects the network environment of CEV [22], which is an important application scenario of TSNs, featuring a complex structure and high representativeness. The specific topology is shown in Figure 9. In the experiments, the number of TT (Time-Triggered) flows was gradually increased (from 10, 20, 30, 40, to 50 flows) to evaluate the performance of the algorithms under different flow loads. Meanwhile, to ensure the robustness of the experiments and the statistical reliability of the results, each algorithm was run 10 times separately, and the optimal value distribution of the makespan during the convergence process was recorded. This analysis was conducted to assess the stability and convergence efficiency of the algorithms.

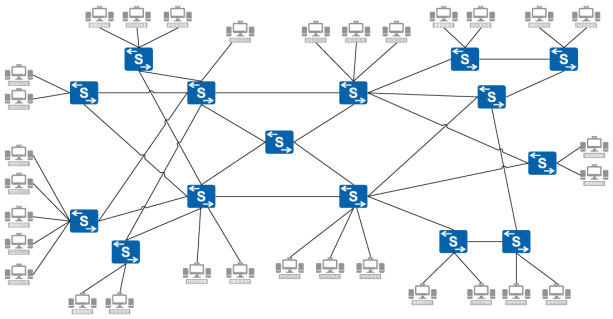

Figure 9. CEV network topology diagram

Figure 10 presents the makespan results for the four algorithms under different numbers of flows. Among them, PDAGA demonstrates the best performance, with its traffic scheduling achieving a lower makespan than TGA, HGA, and FSGA across all flow scales. Even when the number of flows is high, PDAGA retains a significant advantage. HGA and FSGA outperform TGA in terms of makespan optimization, while TGA exhibits the highest makespan across all flow scales, particularly performing poorly as the flow volume increases. This indicates that traditional genetic algorithms struggle to cope with complex scheduling scenarios.

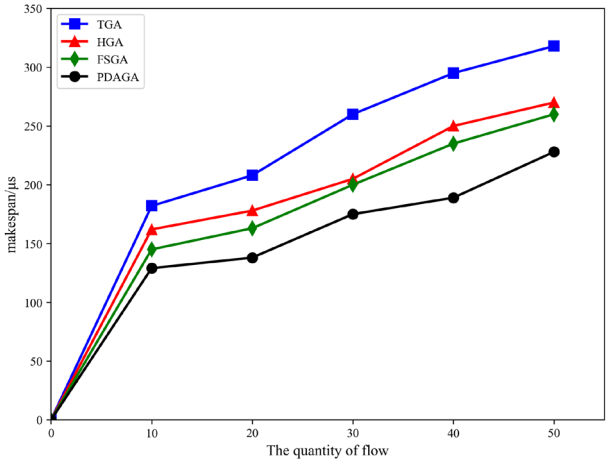

Figure 10. Comparison of makespans under different quantities of streams

Figure 11 displays the distribution of the optimal makespan values for the four algorithms when each was run 10 times under a traffic volume of 50 flows. Among them, PDAGA demonstrates strong convergence stability: the distribution of PDAGA's makespan is concentrated across the 10 runs, with a small range of fluctuation, and the optimal solution stabilizes between approximately 180μs and 200μs. This indicates that the algorithm is highly robust and easily converges to the global optimal solution. The makespan distribution of HGA and FSGA is relatively dispersed, with a fluctuation range of approximately 220μs to 260μs. Some experimental results are close to those of PDAGA, but their overall performance is inferior. In contrast, TGA's makespan exhibits a wide range of fluctuations, suggesting that it is greatly influenced by random factors and struggles to consistently find the optimal solution.

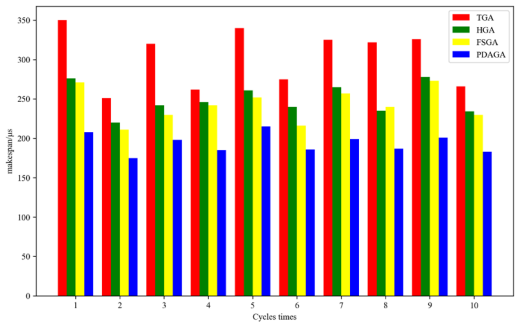

Figure 11. Comparison of optimal makespan distributions under 50 flows across algorithms

The experimental results demonstrate that PDAGA can effectively reduce the scheduling makespan in complex network environments, with particularly significant optimization capabilities under high-load conditions with increased traffic. The makespan distribution of PDAGA is more concentrated, exhibiting higher stability and consistent excellent results across multiple runs. This avoids the local optimum issue that may arise in traditional algorithms. Overall, time-sensitive traffic scheduling based on PDAGA, through the combination of priority strategies, adaptive mutation, and elite preservation strategies, effectively enhances global search capabilities and optimizes the limitations of traditional genetic algorithms in time-sensitive traffic scheduling.

6 Conclusion

In response to the real-time requirements and scheduling complexity of TT traffic in Time-Sensitive Networking (TSN), this paper proposes a joint scheduling method for TSN traffic and routing based on a priority-driven adaptive genetic algorithm. This method optimizes the completion time of time-sensitive traffic scheduling tasks. Firstly, tailored to the characteristics of joint scheduling and routing, a two-vector encoding approach is designed, separately representing the scheduling sequence and path allocation of TT traffic. Population initialization adopts a priority-based hierarchical strategy, categorizing TT traffic into high, medium, and low priority flows according to their deadlines, periods, and load characteristics. This ensures the quality of the initial population while increasing its diversity. In terms of genetic operations, a two-stage selection strategy is designed, combining tournament selection and roulette wheel selection, to balance the quality and diversity of the population. The crossover operation effectively addresses the coupling issue between flow scheduling sequence and path allocation through a joint sequential crossover algorithm, ensuring the feasibility of offspring individuals. The mutation operation introduces a dynamic mutation rate adjustment strategy, adaptively adjusting the mutation rate to strike a balance between global search and local optimization, thereby enhancing the algorithm's convergence speed and global search capability.

Subsequently, extensive experiments were conducted to evaluate our method. Through experiments in various network topologies and with different traffic scales, the results demonstrate that the proposed method exhibits significant advantages in terms of completion time optimization, convergence speed, and stability. This proves its applicability in complex TSN scheduling scenarios.

References

[1] J. L. Messenger, Time-sensitive networking: An introduction, *IEEE Communications Standards Magazine*, Vol. 2, No. 2, pp. 29-33, June, 2018.

[2] N. Finn, Introduction to time-sensitive networking, *IEEE Communications Standards Magazine*, Vol. 2, No. 2, pp. 22-28, June, 2018.

[3] Q. Li, D. Li, X. Jin, Q. Wang, P. Zeng, A simple and efficient time-sensitive networking traffic scheduling method for industrial scenarios, *Electronics*, Vol. 9, No. 12, Article No. 2131, December, 2020.

[4] L. L. Bello, W. Steiner, A perspective on IEEE time-sensitive networking for industrial communication and automation systems, *Proceedings of the IEEE*, Vol. 107, No. 6, pp. 1094-1120, June, 2019.

[5] J. Min, W. Kim, J. Paek, Co-optimization framework for heterogeneous search spaces in time-sensitive network planning, *IEEE Internet of Things Journal*, Vol. 11, No. 7, pp. 11779-11792, April, 2024.

[6] E. Schweissguth, D. Timmermann, H. Parzyjegla, P. Danielis, G. Mühl, ILP-based routing and scheduling of multicast realtime traffic in time-sensitive networks, *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Gangneung, Gangwon, South Korea, 2020, pp. 1-11.

[7] Y. Chi, H. Zhang, Y. Liu, N. Chen, Z. Zheng, H. Zhu, P. Zhang, H. Zhan, Flow-based joint programming of time sensitive task and network, *Electronics*, Vol. 12, No. 19, Article No. 4103, October, 2023.

[8] Y. Nakayama, R. Yaegashi, A. H. N. Nguyen, Y. Hara-Azumi, Real-time reconfiguration of time-aware shaper for ull transmission in dynamic conditions, *IEEE Access*, Vol. 9, pp. 115246-115255, August, 2021.

[9] Z. Feng, Z. Gu, H. Yu, Q. Deng, L. Niu, Online rerouting

and rescheduling of time-triggered flows for fault tolerance in time-sensitive networking, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 41, No. 11, pp. 4253-4264, November, 2022.

[10] D. Bujosa, M. Ashjaei, A. V. Papadopoulos, T. Nolte, J. Proenza, HERMES: Heuristic multi-queue scheduler for TSN time-triggered traffic with zero reception jitter capabilities, *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, Paris, France, 2022, pp. 70-80.

[11] A. Arestova, K. S. J. Hielscher, R. German, Design of a hybrid genetic algorithm for time-sensitive networking, *Measurement, Modelling and Evaluation of Computing Systems: 20th International GI/ITG Conference*, Saarbrücken, Saarland, Germany, 2020, pp. 99-117.

[12] H. Liu, W. Yang, Z. Chang, Joint Optimization for Routing and Scheduling Time-Triggered Flows in Time-Sensitive Networks, *2024 6th International Conference on Natural Language Processing (ICNLP)*, Xi'an, Shaanxi, China, 2024, pp. 687-692.

[13] I. Vlašić, M. Đurasević, D. Jakobović, Improving genetic algorithm performance by population initialisation with dispatching rules, *Computers & Industrial Engineering*, Vol. 137, Article No. 106030, November, 2019.

[14] R. Ruiz, C. Maroto, J. Alcaraz, Two new robust genetic algorithms for the flowshop scheduling problem, *Omega*, Vol. 34, No. 5, pp. 461-476, October, 2006.

[15] M. Kim, D. Hyeon, J. Paek, ETAS: Enhanced time-aware shaper for supporting nonisochronous emergency traffic in time-sensitive networks, *IEEE Internet of Things Journal*, Vol. 9, No. 13, pp. 10480-10491, July, 2022.

[16] M. Pahlevan, R. Obermaisser, Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks, *2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA)*, Torino, Italy, 2018, pp. 337-344.

[17] M. Vlk, K. Brejchová, Z. Hanzálek, S. Tang, Large-scale periodic scheduling in time-sensitive networks, *Computers & Operations Research*, Vol. 137, Article No. 105512, January, 2022.

[18] Y. Zhang, Q. Xu, L. Xu, C. Chen, X. Guan, Efficient flow scheduling for industrial time-sensitive networking: A divisibility theory-based method, *IEEE transactions on industrial informatics*, Vol. 18, No. 12, pp. 9312-9323, December, 2022.

[19] Y. Zhou, S. Samii, P. Eles, Z. Peng, Reliability-aware scheduling and routing for messages in time-sensitive networking, *ACM Transactions on Embedded Computing Systems (TECS)*, Vol. 20, No. 5, pp. 1-24, September, 2021.

[20] X. Jia, W. Xu, Y. Chen, L. Yang, Hybrid self-backhaul and cache assisted millimeter wave two-tier heterogeneous networks with MIMO equipped backhaul access points, *IEEE Access*, Vol. 7, pp. 59963-59983, May, 2019.

[21] X. Jia, P. Deng, L. Yang, H. Zhu, Spectrum and energy efficiencies for multiuser pairs massive MIMO systems with full-duplex amplify-and-forward relay, *IEEE Access*, Vol. 3, pp. 1907-1918, October, 2015.

[22] A. A. Atallah, G. Bany Hamad, O. Ait Mohamed, Multipath routing of mixed-critical traffic in time sensitive networks, *From Theory to Practice: 32nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, Graz, Styria, Austria, 2019, pp. 504-515.

Biographies

Wei Zheng is currently an associate professor with the Department of Software, University of Northwestern Polytechnical. His research is focus on software security and software quality assurance.

Zhiqiang Zhu is currently a student of the Department of Software, University of Northwestern Polytechnical, for the Doctor degree, and the researcher of the AVIC Xi'an Aeronautics Computing Technique Research Institute. His research focus is on aeronautical communication network.

Yu Zhang is currently a student of the Department of Software, University of Northwestern Polytechnical, for the Master degree. His research focus is on software security.

Jin Shao is currently a student of the Department of Software, University of Northwestern Polytechnical, for the Master degree. His research focus is on software security.