






Article

Heuristic Routing Algorithms for Time-Sensitive Networks in Smart Factories

Yue Li ^{1,2,3,†} , Zhenyu Yin ^{1,2,3,*,†} , Yue Ma ^{1,2,3}, Fulong Xu ^{1,2,3} , Haoyu Yu ^{1,2} , Guangjie Han ^{4,5} 
and Yuanguo Bi ^{6,7}

- ¹ School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China; liyue161@mails.ucas.ac.cn (Y.L.); mayue@sict.ac.cn (Y.M.); xufulong16@mails.ucas.ac.cn (F.X.); yuhaoyu@sict.ac.cn (H.Y.)
 - ² Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China
 - ³ Liaoning Key Laboratory of Domestic Industrial Control Platform Technology on Basic Hardware and Software, Shenyang 110168, China
 - ⁴ College of Internet of Things Engineering, Hohai University, Changzhou 213022, China; hanguangjie@gmail.com
 - ⁵ Changzhou Key Laboratory of Internet of Things Technology for Intelligent River and Lake, Changzhou 213022, China
 - ⁶ School of Computer Science and Engineering, Northeastern University, Shenyang 110167, China; biyuanguo@mail.neu.edu.cn
 - ⁷ Engineering Research Center of Security Technology of Complex Network System, Ministry of Education, Shenyang 110167, China
- * Correspondence: congmy@163.com
† These authors contributed equally to this work.

Abstract: Over recent years, traditional manufacturing factories have been accelerating their transformation and upgrade toward smart factories, which are an important concept within Industry 4.0. As a key communication technology in the industrial internet architecture, time-sensitive networks (TSNs) can break through communication barriers between subsystems within smart factories and form a common network for various network flows. Traditional routing algorithms are not applicable for this novel type of network, as they cause unnecessary congestion and latency. Therefore, this study examined the classification of TSN flows in smart factories, converted the routing problem into two graphical problems, and proposed two heuristic optimization algorithms, namely GATTRP and AACO, to find the optimal solution. The experiments showed that the algorithms proposed in this paper could provide a more reasonable routing arrangement for various TSN flows with different time sensitivities. The algorithms could effectively reduce the overall delay by up to 74% and 41%, respectively, with promising operating performances.

Keywords: time-sensitive network; smart factory; industrial internet; routing; heuristic algorithm



Citation: Li, Y.; Yin, Z.; Ma, Y.; Xu, F.; Yu, H.; Han, G.; Bi, Y. Heuristic Routing Algorithms for Time-Sensitive Networks in Smart Factories. *Sensors* **2022**, *22*, 4153. <https://doi.org/10.3390/s22114153>

Academic Editor: Jose Manuel Molina López

Received: 12 April 2022

Accepted: 26 May 2022

Published: 30 May 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the rapid development of the industrial internet, real-time communication technologies with deterministic low latency have become a critical requirement in many industrial sectors. For example, most industrial automation networks require end-to-end latency to be strictly controlled at no more than 1 millisecond [1]. In addition to the latency requirements, most application scenarios also have diverse demands on transmission jitter, packet loss rate, etc., while traditional Ethernet communication can only provide best-effort and soft real-time transmission services. In response to the growing demand for industrial real-time communication, industrial enterprises all over the world have developed various industrial control network protocols based on standard Ethernet communication, such as Real-Time TTEthernet, EtherCAT, PROFINET, SERCOIII, etc. These deterministic industrial networks connect manufacturing equipment and controllers, constituting operation technology (OT) networks [2] that are now widely used.

However, incompatible network protocols lead to problems such as incompatible applications, a lack of interoperability, difficulty in portability, and expensive costs for development, deployment, and maintenance. With the relentless efforts of the AVnu Industry Alliance and the IEEE TSN Working Group (formerly the AVB Working Group), TSNs [3] have emerged as a brand new industrial communication technology that are now being actively promoted by industrial communities. TSNs allow both periodic and non-periodic data to be transmitted within the same network, giving standard Ethernet communication the ability for deterministic transmission. TSNs are constructed on the standard 802.1 Ethernet protocol stacks, which naturally have the advantage of interconnection and can achieve open Layer 2 Forwarding while ensuring deterministic latency bounds and bandwidth guarantees [4]. Therefore, TSNs are able to interconnect mutually isolated information technology (IT) networks and OT networks to achieve the co-networked converged transmission of multiple data flows with varying time sensitivities [5]. In recent years, TSNs have received continuous attention from both academia and industry and have been identified as a key technology for the next generation of industrial communication systems [6–8]. With this continuous attention and the efforts of standardization organizations, including the IEEE and IEC, a series of amendments and standards to improve TSN protocols [9–11] have been released.

The continuous improvement of TSN-related standardization has become a research trend in the building of a comprehensive industrial internet communication system that is based on a TSN with deep integration of IT and OT [12]. Among many application directions, smart factories, a key component of Industry 4.0, comprise an important application scenario for TSN communication technology. A smart factory is a comprehensive production system with multiple intelligent subsystems at different levels [13], with each subsystem having different needs for industrial communication. While ensuring real-time control data communication for smart factories, only TSNs can realize barrier-free communication with other the subsystems mentioned above within the smart factory architecture [14,15]. For TSNs in smart factories, a novel routing mechanism needs to be established that considers their unique characteristics.

Currently, research on TSNs is still in the development stage and existing data flow route planning mechanisms are relatively simple. In our previous studies [16–18], we contributed to the traffic shaping mechanisms along with real-time secure communication methods for joint OPC UA–TSN IoT-based intelligent industrial production lines; however, the depth of discussion around routing for TSNs has much room for improvement. In both mainstream studies and our previous studies, the network topologies were not complicated, routes of all kinds of traffic were determined using spanning tree protocols and shortest path routing algorithms, and port queuing and time slot allocation in frame transmission were performed. However, the composite routing problem for multiple flows within large-scale networks is difficult to solve in polynomial time, so online routing algorithms can hardly meet the growing needs of TSNs [19]. Researchers have turned to developing offline routing methods, which aim to generate a reasonable network routing list in an offline manner through exploiting the characteristics of TSN deterministic communication. During actual communication, data flows are scheduled according to the preset routing list. Compared to online algorithms, this approach avoids the strict computing time limits and can perform more iterations to obtain a better approximate optimal solution. Therefore, we propose an offline TSN routing method that covers both TT flow and non-TT flow routing problems, based on heuristic algorithms. The main contributions of our study are as follows:

1. We proposed an improved genetic algorithm to solve the TT flow routing problem (GATTRP). We modeled the TT flow routing problem in the TSN systems of smart factories and transformed the problem into a multiple traveling salesmen problem (MTSP) to solve. Based on the existing genetic algorithms, we optimized the design of the genetic evolution operators and algorithm processes and finally, formed an improved genetic algorithm with faster convergence speed and better results;

2. We proposed an adversarial ant colony optimization (AACO) algorithm to solve the non-TT flow routing problem. We modeled the non-TT flow routing problem in the TSN systems of smart factories and transformed the problem into a load balancing pathfinding problem for multiple priority flows within a directed graph. Based on existing ACO algorithms, we designed a novel pheromone update rule to balance the impacts of higher priority tasks and path length on pathfinding. The algorithm could effectively equalize the non-TT network load and reduce the network latency;
3. We established a simulation experiment platform for the smart factory TSN communication system and evaluated the performance of the proposed algorithm through experiments. The results showed that both algorithms produced a certain improvement in the corresponding evaluation indicators, which matched our expectations.

The remainder of this paper is organized as follows. In Section 2, we list the results of our survey on related works. In Section 3, we parse and mathematically model the actual problems. In Section 4, we propose the TT routing algorithm, named GATTRP. In Section 5, we propose the non-TT routing algorithm, named AACO. In Section 6, we establish an experimental environment and discuss performance evaluation. Finally, we conclude this paper in Section 7.

2. Related Works

Our study combined several fields, such as time-sensitive network routing, task/volume-balanced MTSPs, load balancing routing, etc. We reviewed many studies from the related fields, which are listed below.

2.1. Time-Sensitive Network Routing

TSNs are composed of series of IEEE technical standards, including precise time synchronization, network traffic shaping, network configuration, and other aspects. The IEEE TSN Standardization Working Group issued a standard amendment [20] to address the problem of path controlling and proposed shortest path bridging (SPB) as the basis for establishing network bridging in Chapter 12. Based on this, a management information base (MIB) was specified in Chapter 17, which formed the IEEE 8021-SPB-MIB standardization framework. Furthermore, the amendment proposed a path control and reservation (PCR) method in Chapter 45, which was based on the use of SPB and spanning tree protocols to achieve the path control and traffic reservation of TSN flows. Following on from IEEE standards, Schweissguth et al. and Falk et al. [21,22] proposed solutions for the joint traffic scheduling and route planning problem, both of which were based on integer linear programming (ILP). Schweissguth et al. [21] used two performance metrics (i.e., end-to-end delay and scheduling capability) to evaluate their experimental results for different traffic patterns and network topologies. Falk et al. [22] adopted an ILP solver for instances with large parameter variations and evaluated the performance of their algorithm based on the solution time. Mahfouzi et al. [23] proposed an iterative algorithm for joint routing and scheduling based on SMT, but the performance of their algorithm was too sensitive to the degree of transmission path conflicts between flows, which led to an unsatisfactory success rate. Nayak et al. [24,25] proposed the concept of a time-sensitive software-defined network (TSSDN), which forms a logically centralized control plane of SDN to compute global routing schemes. The above studies provided feasible routing mechanisms for TT traffic, but it is difficult to meet the requirements of the solution time for large-scale routing scenarios and no relevant studies have been found that route for non-TT traffic. Therefore, the existing methods can hardly cope with the routing challenges brought by future large-scale TSN communication systems.

2.2. Heuristic Optimization Algorithms

Heuristic algorithms are intuitively or empirically constructed algorithms that search for a feasible solution to each instance of an optimization problem at a limited cost (in terms of computational time and space). Since heuristic algorithms can usually find

promising solutions in a reasonable amount of time when dealing with many practical NP-hard problems, they have become a research hotspot in recent years. Inspired by various phenomena, such as animal behavior and natural laws, researchers have proposed many novel and effective optimization algorithms and have proven their value in practical applications. For example, based on the gravitational search algorithm (GSA), which was inspired by the law of gravity and interactions between mass entities, Precup et al. [26] proposed the tuning of a class of fuzzy control systems to obtain a reduced sensitivity. Li et al. [27] proposed an effective rule classifying method, namely the heuristic algorithm to reduce memory demand (HARD), for heterogeneous bit-split string matching architectures. Based on the gray wolf optimization (GWO) algorithm, which was inspired by the action of a gray wolf preying on its prey, Zamfirache et al. [28] proposed an RL-based control approach to train neural networks. Pozna et al. [29] proposed a hybrid metaheuristic optimization algorithm called the particle filter–particle swarm optimization (PF–PSO) algorithm, which can effectively optimize the position control of a family of integral-type servo systems. The above works have been proven to be successful in various applications and thus, are valuable for the further improvement of heuristic algorithms. In order to solve the multi-objective task scheduling problem of intelligent production lines, we proposed a hybrid algorithm called the improved hybrid monarch butterfly optimization and improved ant colony optimization algorithm (HMA) [30] to combine the advantages of cloud computing and fog computing. Based on our previous research, we started trying to solve the routing problems in TSN transmission using heuristic algorithms.

2.3. Task/Volume-Balanced MTSPs

The traveling salesman problem (TSP) is a typical NP-hard combinatorial optimization problem, which comprises finding the best traversal route at the lowest cost (time, distance, etc.) through a given number of cities, in which all cities are visited only once by a single traveler, except for the starting city [31]. The MTSP, on the other hand, comprises M travelers visiting a portion of cities separately and each city (except the starting city) is only visited by any traveler once and, eventually, finding the minimum cost to finish traversing all of the cities [32]. When $M = 1$, MTSP is transformed into classical TSP, so TSP is a special case of MTSP [33].

The genetic algorithm (GA) has definite advantages for solving the task-balanced MTSP problem. Carter et al. [34] proposed a two-stage chromosome encoding method based on classical GA and designed corresponding genetic operators to solve the MTSP both in terms of the shortest total distance and the shortest "longest distance", which could effectively reduce the solution space and eliminate redundant solutions. Zhou et al. [35] successfully proved the advantages of the improved uniparental GA for solving the MTSP, as well as proposing three algorithms to solve the MTSP with multiple starting points and closed loops. Lu et al. [36] combined the K-means clustering algorithm and the GA to solve the multi-objective MTSP, which avoided travelers crossing paths and also reduced computing time. However, the correctness of the results and the convergence performance of the above algorithms still need to be improved as retaining good individuals while maintaining the diversity of the population within the GA for the MTSP remains a challenging problem.

2.4. Load Balancing Routing Assignment Problem

When routing in large-scale network systems, the load balancing problem needs to be fully considered to avoid partial network congestion. To solve the load balancing routing problem, some existing clustering protocols for wireless sensor networks (WSNs) have appreciable reference value. The LEACH (low-energy adaptive clustering hierarchy) algorithm [37] was the first proposed hierarchical routing algorithm, whose core idea is to divide the network nodes into clusters and randomly select nodes in turn to be the cluster head nodes. The other nodes forward their collected data to the cluster head node and, eventually, the cluster head node consolidates the data and forward them to the sink

node. Younis and Fahmy [38] proposed a HEED clustering approach. Its major difference from the basic LEACH protocol is that HEED uses a multi-hop method to communicate with the sink node, while LEACH uses a single-hop method. Inspired by these two important clustering protocols, Tarhani et al. [39] proposed SEECH, Bhushan et al. [40] proposed FLEAC, and Sert et al. [41] proposed MOFCA, forming a rich variety of clustering routing methods for WSNs that have a better performance.

There are researchers continuously trying to apply ant colony optimization (ACO) algorithms to solving the load balancing routing assignment problem. ACO is an intelligent optimization algorithm that optimizes practical problems by imitating the foraging behavior of ants in nature, which was first proposed by Italian scholar Marco Dorigo in the 1990s [42]. Ramamoorthy et al. [43] proposed an enhanced hybrid ant colony optimization routing protocol (EHACORP) to improve the efficiency of the routing process using the shortest path. Belgaum et al. [44] explored two artificial intelligence optimization techniques, including ACO and PSO, for load balancing in SDN. Govardhan and Srinivasan [45] proposed a modified evolutionary computing-driven dynamic load balancing model, named intrinsically modified ant colony system (IMACS), for mega-cloud infrastructures. The above algorithms have different degrees of optimization for the route assignment problem of load balancing. However, to the best of our knowledge, there is still a lack of well-performing load balancing route assignment methods for TSN multi-priority scheduling characteristics.

3. Problem Modeling

To study the TSN routing problem within a complex network topology, the network was abstracted as a directed graph $G(V, E)$. The nodes in this network included two main types: switch (SW) nodes and end system (ES) nodes, as shown in Figure 1. All of the nodes formed a node set V and each SW had multiple incoming and outgoing ports, which were responsible for forwarding the data frames received from the incoming ports to the corresponding outgoing ports, according to the routing list. The ESs could be hard real-time control units, such as servo drivers, or soft real-time or non-real-time units, such as sensors, cameras, mobile operating terminals, etc. $E \subseteq V \times V$ was the set of edges, where each element represents a unidirectional link from one node to another. The TSN connections supported full-duplex, so the physical links between node v_i and v_j corresponded to two directed edges $[v_i, v_j]$ and $[v_j, v_i]$ in the model. Each link $[v_i, v_j]$ was defined by a triplet $\langle b_{ij}, c_{ij}, q_{ij} \rangle$, which denoted the bandwidth capacity, propagation delay, and queuing delay of that link, respectively.

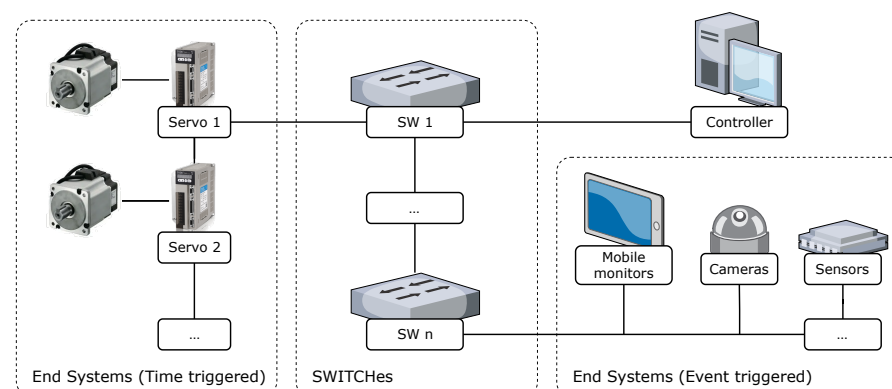


Figure 1. The architecture of a TSN in a smart factory and the composition of the TSN nodes.

Based on the common TSN traffic classification method [7,46,47], we classified the industrial data transmitted by TSN industrial communication systems in smart factories into three main types. For high-precision servo motors in key manufacturing equipment, such as computer numerical control (CNC) machine tools and six-axis robotic arms, the master unit needs to periodically send control data, along with time-synchronized data, as de-

defined in [9,48], which are collectively called time-triggered (TT) data. Meanwhile, some upper-level industrial applications that rely on computer vision, such as object recognition inspection systems for workpiece shapes, require access to audio and video surveillance streams throughout the manufacturing process. These data, collectively called audio/video bridging (AVB) data, are less time-sensitive than TT data and, therefore, need to be scheduled for transmission after the highest priority TT data. In addition, TSNs also provide non-real-time transmission services for upper industrial integrated management systems, such as ERP and MES, in smart factories. These communication systems are not considered within the QoS of real-time industrial networks, collectively called best-effort (BE) communication. A summary of the above three types of data is depicted in Table 1.

Table 1. TSN data classification.

Category	Sample Data	Description	Priority
TT	Control Data Frames	Communicate with industrial control slave devices, such as servo motors, on a strictly time cycle basis to control their actions and collect encoder feedback	7
	Time-Synchronized Frames	Traverse each network node following the rules of the optimal master clock algorithm to complete precise time synchronization	
AVB	Audio Bridging Data	Sensing signals, such as vibration, sound, etc., typically requiring latency to be less than 5 ms	5~6
	Video Bridging Data	Continuous image signals captured by industrial surveillance cameras with large bandwidth consumption: allowable time delay range is 0~100 ms	
	Key Sensor Data	Event-triggered multi-source heterogeneous sensor signal data, which is an important data source for realizing intelligent manufacturing management	
BE	ERP, MES, etc. System Data	Generic Ethernet data with no particular real-time QoS requirements	0~4
	Background Stream Data	Deliver as much as possible	

AVB and BE data cannot be transmitted in hard real-time, as with TT data, so we defined them as the same type of data, i.e., non-TT data, in our study. In the following sections, we define the two different routing problems, TT and non-TT, and propose two different optimization algorithms to solve these routing problems.

4. Improved Genetic Algorithm to Solve TT Flow Routing Problem

When the controller conducts time-triggered communication with servo motors within TT subnetworks, the communication method is the master–slave method, in which the controller acts as the master station to send control-type frames to each slave station and slave stations return the processed frames to the master station. In the IT–OT converged industrial control network considered in this paper, the technical idea of aggregated forwarding frames was adopted, in which the whole subsystem has only one frame that runs in a loop. For the master, all devices with I/O information are considered as “logical” devices and the address of the field device corresponds to the physical location in the control frame, according to the protocol. When a message passes through a slave device, the slave only needs to read the command data from the corresponding mapped address and simultaneously resend the feedback data to the same place, so the effective utilization of the message can reach up to more than 90%. The frame format for the control data and communication mechanism design are shown in Figure 2.

In industrial control networks, the number of slaves that need to be traversed in single time cycle increases as the network size continues to grow and the time cycle that is required to traverse the nodes increases linearly. In this case, we considered a TT control network based on the idea of distributed control, as depicted in Figure 3. In smart factories, due to the high degree of integration of IT and OT networks, the central control server can combine the requirements from upper-level industrial applications and the cloud platform to simultaneously control CNC machine tools, industrial robots, and other key

manufacturing equipment on multiple smart production lines in real time. Each CNC system, six-axis robot arm, etc., forms a set of subnetworks with its own hard real-time requirements within the system. The central control server accesses the submasters of each subnetwork sequentially via SWs, controls them with TT frames, and collects feedback. By optimizing the traversal route, we can reduce the traversal period as much as possible, so that the access time interval from the central control server to each subnetwork can be significantly reduced. The more frequent the access to the manufacturing equipment in the same time slice, the faster the speed of command response and the higher the manufacturing accuracy and flexibility of the smart production line. Therefore, the optimization objective for TT flows in this paper was to reduce the traversal cycle from the central control server to each real-time subnetwork.

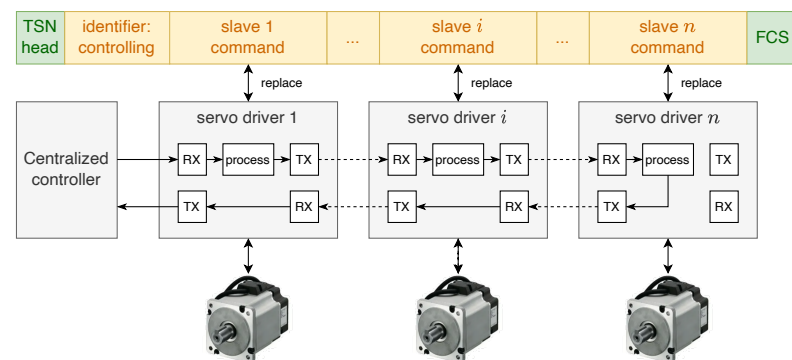


Figure 2. Aggregated forwarding frame-based TT communication.

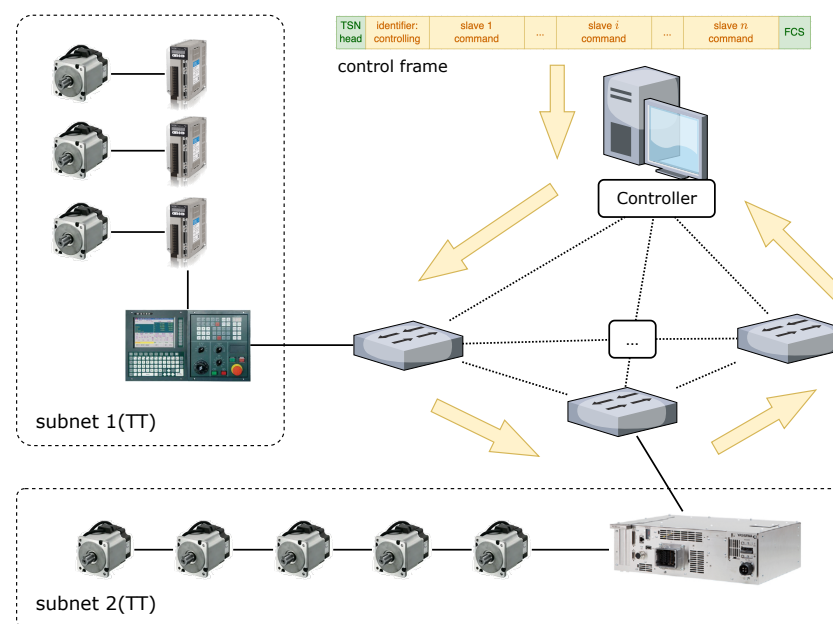


Figure 3. Schematic of TT control flow network traversal.

4.1. Definition of the Optimization Problem in TT Routing

Based on the distributed traversal approach for TT communication that was proposed above, the TT routing problem in TSNs could be transferred into the following: the central control node v_1 was directly connected to m SW nodes and periodically sent m control-type data frames (noted as a_1, a_2, \dots, a_m) to n destination subnetworks at the same time. After each subcontrol system finished receiving and processing, it overwrote the control data field at the corresponding position within the frame with feedback data containing operating status and then forwarded it downward. Finally, all frames converged at v_1 after the traversal. Every subnetwork that was connected to a SW node could be merged into

the model. Considered as a single node, the processing delay of the node equaled the total traversal time inside the node. We let the number of ES nodes inside subnetwork v_i be o_i and the time granularity of traversing a single ES node be δ , then the total time delay d_{ij} between node v_i and v_j included the propagation delay between the two nodes, the queuing delay, and the processing delay required to traverse within the v_i node:

$$d_{ij} = c_{ij} + q_{ij} + \delta \cdot o_i. \quad (1)$$

The minimum time that was required to complete a cycle equaled the time required for the longest of all sent frames to complete forwarding and return to the source node, i.e., the minimum value of cycle T was:

$$T = \min \left(\max \sum_{i=1}^n \sum_{j=1}^n d_{ij} \cdot \rho_{ij}^k \right), \forall k \in \{1, 2, \dots, m\}, \quad (2)$$

where ρ_{ij}^k is the transmission direction of the TT frame between nodes v_i and v_j , which is defined as follows:

$$\rho_{ij}^k = \begin{cases} 1, & \text{TT frame } k \text{ goes from } i \text{ to } j \\ 0, & \text{TT frame } k \text{ goes from } j \text{ to } i \end{cases} \quad (3)$$

For each k th TT frame, we planned a loop route starting from v_1 . Assuming the route was $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \dots \rightarrow v_6 \rightarrow v_1$, there had to be $\rho_{1,2}^k = \rho_{2,3}^k = \dots = \rho_{6,1}^k = 1$, while any other $\rho_{ij}^k = 0$. Then, we obtained the $1/m$ part of the solution: route $\omega_k = (v_2, v_3, \dots, v_6)$.

In addition, we defined y_i^k to mark whether the k th TT frame had visited node v_i as follows:

$$y_i^k = \begin{cases} 1, & \text{TT frame transmitted through node } v_i \text{ once} \\ 0, & \text{others} \end{cases} \quad (4)$$

Each node v_i was marked one time per TT frame arrival; therefore, for every node v_i , there was:

$$y_i^k = \sum_{j=0}^n \rho_{ji}^k, \forall i \in \{1, 2, \dots, n\}, k \in \{1, 2, \dots, m\}. \quad (5)$$

By treating the node v_i as a city, the links between nodes e_{ij} as inter-city paths, and the m TT frames transmitted in parallel as m traveling salesmen, we could further transform the TT routing problem into an MTSP problem.

4.2. Optimization Goal and Constraints of TT Routing

Through the description of the above formula, our goal formula became clear: comparing the completion times of all m salesmen to find the longest route. Our goal was to minimize the completion time of the salesman who traveled the longest route by adjusting the routing plans for all of the salesmen, namely:

$$\begin{aligned} & \text{Minimize}\{T\}, \\ \text{s.t. } & (c1) : \sum_{k=1}^m y_1^k = m, \\ & (c2) : \sum_{k=1}^m y_i^k = 1, \forall i \in \{2, 3, \dots, n\}, \\ & (c3) : \sum_{i=1}^n y_i^k \geq 2, \forall k \in \{1, 2, \dots, m\}, \\ & (c4) : T \leq \text{MaxTime}. \end{aligned} \quad (6)$$

In Equation (6), (c1)–(c4) are the constraints that needed to be obeyed by any solution. Constraint (c1) ensured that m TT frames all returned to the central control node within

a single common traversal cycle. Constraint (c2) ensured that every node v_i , except for v_1 , was visited only one time because a node being accessed two or more times within one cycle would break the transmission periodicity and thus, cause chaos in the whole network. Constraint (c3) ensured that each route included at least one node in addition to the central control node. Constraint (c4) ensured that the final result T was less than the preset threshold $MaxTime$, otherwise its real-time performance could not be guaranteed.

4.3. Description of GATTRP

In this section, we propose an improved GA named GATTRP. Through GATTRP, we could encode the routes of the TT flows and seek the optimal solution for this problem in a genetic evolutionary way. In the following content, we describe how the improved GA works, starting with the rules by which the chromosomes are encoded.

4.3.1. Chromosome Encoding Rules

In order to reduce the search space and eliminate redundant solutions, this paper adopted a two-segment chromosome encoding method. The first segment represented the order of salesmen traversing each node and the second segment represented the breakpoints between each salesman. When there were n nodes, the first node v_1 was fixed as the starting and destination node for salesmen to traverse; the other $n - 1$ nodes were randomly arranged to be visited by m salesmen. The fixed starting node was not encoded in the chromosome and the length of the first part of the chromosome was $n - 1$, indicating the random arrangement of $n - 1$ nodes. The length of the second part was $m - 1$, indicating that when the nodes needed to be divided into m salesmen's paths, $m - 1$ breakpoints were needed. The breakpoints in the second part were stored in increasing order.

In the following example, we made $n = 10$, $m = 3$, and the number of the fixed starting node be 1. We let the randomly generated breakpoints be 4 and 6. Then, we could encode the chromosome as is depicted in Figure 4. The traversal route of the first salesman was $1 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 4 \rightarrow 1$, the traversal route of the second salesman was $1 \rightarrow 2 \rightarrow 8 \rightarrow 1$, and the traversal route of the third salesman was $1 \rightarrow 10 \rightarrow 3 \rightarrow 7 \rightarrow 1$.

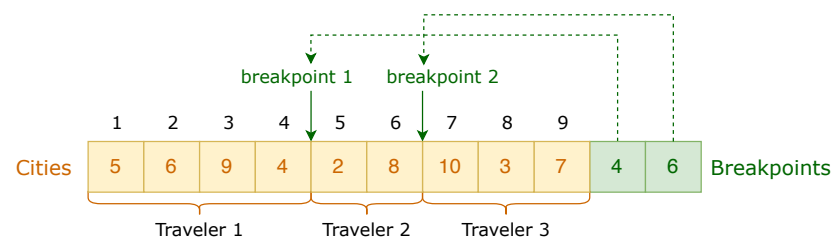


Figure 4. Encoding method for an example chromosome.

The above chromosome corresponds to the route of TT flows traversing in the actual network as depicted in Figure 5.

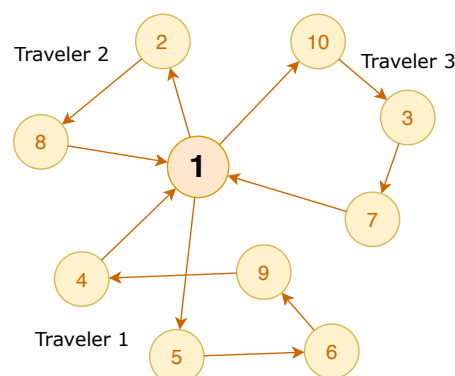


Figure 5. The traversal route of the chromosome in Figure 4.

4.3.2. Population Initialization

We initialized a population with an initial chromosome number of U_0 and we set the maximum population size allowed throughout the iteration as U_{max} and the maximum number of iterations as I_{max} . The gene segments of U_0 chromosomes were randomly initialized and the fitness of individuals on each chromosome was calculated.

4.3.3. Genetic Evolution Operator Design

We combined simple operators, such as the *flip*, *slide*, *swap* mutation, to design relatively complex operators, which improved the diversity of the mutation process, accelerated the evolution process, and, eventually, enhanced the efficiency of the GA.

In existing mutation operators, the route length of each traveling salesman is constant, which is not enough to effectively improve the population diversity and it is detrimental to locally search for new possible solutions. Therefore, for the second segment of the chromosome, the operation of $+n$, $-n$ or $+0$ was randomly applied to the breakpoint gene segments on the premise that the breakpoints were not equal to each other and were arranged in ascending order. The effect of the above operations was a change in the length of the salesmen's routes. The mutation process is depicted in Figure 6.

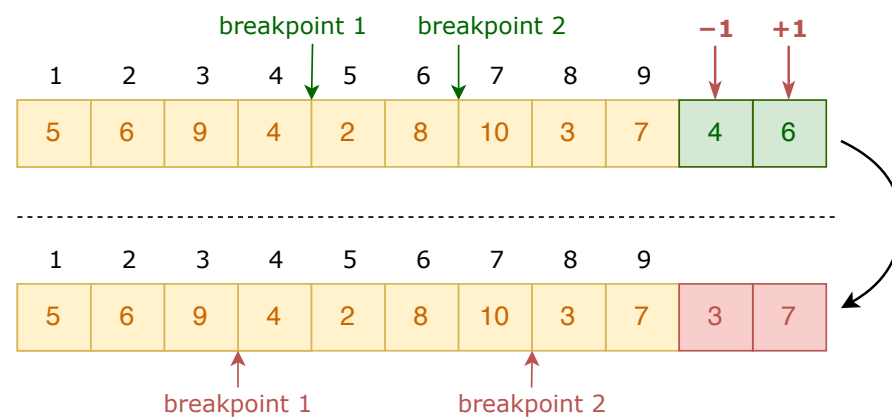


Figure 6. Example process of drift mutation.

4.3.4. Offspring Breeding

The optimization goal of the improved GA proposed in this section was to find the traversal route with the earliest "latest arrival" time of multiple traveling salesmen. Therefore, Equation (2) was determined as the fitness function for this section. We defined fitness fit_i as the evaluation criteria when breeding the i th chromosome, where $fit_i = 1/\min(T)$. After generating the primary population according to the above chromosome encoding method, chromosomes with better fitness were selected to breed more offspring, so that they could keep their genetic advantages in the evolutionary iteration process. The specific number of reproductions of each chromosome was:

$$O_i = \left\lfloor (O_{max} - O_{min}) \frac{fit_i - fit_{min}}{fit_{max} - fit_{min}} \right\rfloor + O_{min}, \quad (7)$$

where O_i represents the number of offspring of the i th chromosome, O_{max} and O_{min} are the maximum and minimum allowable numbers of breeding offspring, respectively, and fit_{max} and fit_{min} are the maximum and minimum fitness values in the formed population, respectively.

In addition, we proposed an adaptive regeneration strategy to improve the searching ability and robustness of the GATTRP. We recorded the elite chromosomes with the top fitness rankings in each round of iterations. When the elite chromosomes were not evolved within 20% of I_{max} rounds in a row, the chromosomes in the bottom 10% of the genetic population were selected to perform a swap crossover with the elite chromosomes. By

adopting an adaptive regeneration strategy to replace evolutionarily stagnant chromosomes, the diversity of the population was increased to improve the evolutionary ability of the algorithm.

4.4. Algorithm Flow

Based on the above theory, the pseudo-code of the GATTRP proposed in this paper is described in Algorithm 1.

Algorithm 1: GATTRP.

Input: $m, n, \{d_{ij}\}, G(V, E)$
Output: $\omega_1, \omega_2, \dots, \omega_m, 1/fit_{max}$

- 1 Initialize parameters $U_0, U_{max}, O_{min}, O_{max}, I_{max}$;
- 2 Randomly initialize U_0 chromosomes of length $n + m - 2$;
- 3 **for** iterator from 1 to I_{max} **do**
- 4 Calculate the fitness $\{fit_i\}$ of each chromosome according to Equation (2);
- 5 Sort chromosomes according to fitness;
- 6 **if** population size $> U_{max}$ **then**
- 7 | Eliminate the last chromosomes until the population size = U_{max} ;
- 8 **end**
- 9 **if** There is no elite chromosome records **then**
- 10 | Mark the top 20% of chromosomes as elite chromosomes and record them;
- 11 **else**
- 12 **if** The top 20% of chromosomes are consistent with the elite chromosome records **then**
- 13 | The number of iterations without valid evolution +1;
- 14 **if** The number of iterations exceeds the allowable range **then**
- 15 | Take postposition chromosomes to swap with elite ones;
- 16 **end**
- 17 **else**
- 18 | Update elite chromosome records;
- 19 | Reset the number of iterations without valid evolution;
- 20 **end**
- 21 **end**
- 22 **forall** chromosomes in the current population **do**
- 23 Calculate the number of child breeding O_i according to Equation (7);
- 24 **for** iterator from 1 to O_i **do**
- 25 | Select other chromosome by roulette to swap and obtain a child;
- 26 | The child have P_{mut} probability of mutation. Mutation of flip, slide, and drift occurs in the proportion of $P_{flip}, P_{slide},$ and P_{drift} respectively;
- 27 **end**
- 28 **end**
- 29 Mix offspring chromosomes into current population;
- 30 **end**
- 31 Decompose the chromosome with the highest fitness into m sub-segments;
- 32 **for** k from 1 to m **do**
- 33 | In the k th sub-segment, sequentially obtain the values of the connected segments i, j , make $\rho_{ij}^k = 1$, generate $\omega_1, \omega_2, \dots, \omega_m$;
- 34 **end**

4.5. Working Pattern of GATTRP

We added a TT routing program based on GATTRP into the central control server of the smart factory TSN to optimize the TT routing. When a new TT subnetwork sent a registration message to apply to join the current TSN, the transmission continued normally

while the GATTRP process was created to perform offline route planning. The newly joined subnetwork was treated as a node in V and the delay cost between it and other nodes was calculated to generate the new $\{d_{ij}\}$. The updated $G(V, E)$ and $\{d_{ij}\}$ were input to the GATTRP process to calculate m new routes. Starting from the next time period, the central control server traversed the entire TT network with m new paths to complete the registration of the new subnetwork.

4.6. Convergence of GATTRP

GATTRP is based on a heuristic algorithm and the average time complexity of its convergence is complex and can be influenced by various factors, including population size, number of iterations, and randomness. Thus, it was difficult to produce an accurate expression. For its convergence, we could only estimate its time complexity as $O(U_{max} \times O_{max} \times I_{max})$.

4.7. Key Innovations and Contributions of GATTRP

To the best of our knowledge, GATTRP transforms, for the first time, the TT transmission problem in a smart factory TSN into an MTSP problem for optimization, which is the key innovation of this paper in terms of the TT routing problem. In addition, in contrast to existing genetic algorithms, GATTRP adds an elite chromosome mechanism that alleviates the problem of the GA tending to fall into premature and difficult-to-search-for solutions. With this optimization, GATTRP has a higher probability of obtaining better results after the iterations.

5. Adversarial Ant Colony Optimization Algorithm for Solving Non-TT Routing Problem

Non-TT flows in TSNs are used for soft real-time or non-real-time applications, providing bounded worst-case end-to-end delay (WCD) but with a looser delay constraint than TT flows. Multiple types of data flows, including TT, AVB, and BE, are sent from multiple ESs, as well as presequential SWs, which are connected to ingress ports in a single SW. To solve the composite traffic scheduling problem, the IEEE 802.1 Qbv standard [49] defined the time-aware shaper (TAS) mechanism to achieve traffic scheduling for different priority queues in time windows by establishing gate control lists (GCLs). Under the condition of network clock synchronization, the GCL in TAS periodically controls the opening and closing of the egress gates of the corresponding priority queue. With the adoption of GCLs, the transmission rate can match the egress bandwidth while segregating traffic of different priority levels, thereby reducing the interference of low-priority traffic on high-priority traffic and avoiding the starvation of low-priority traffic as much as possible.

Typically, the SWs keep a total of eight priority queues, so the sequence of gates is Gate 7 to Gate 0. In the examples in this paper, "o" means gate open, "C" means gate closed, and each action is based on the time window. Correspondingly, the GCL can tell the time sequence of each type of traffic that is allowed to be sent to the output ports in a scheduling cycle, as shown in Figure 7.

In this scheduling mechanism, the number of time windows allocated to each priority queue within a common time period is limited. Therefore, as the number of queueing flows increases, it inevitably leads to relative congestion, which eventually causes an increase in the time needed to complete the transmission of each single task. For non-TT flows, ESs connected to each SW may send multiple types of non-TT data with different priorities, such as audio, video or sensor signals. The destination of non-TT flows is basically concentrated on one centralized data server. In this case, when the routing mechanism assigns too many time-consuming non-TT flows to the same data link, it is difficult to meet the soft deadlines of all flows due to severe time slot contention, resulting in a waste of bandwidth resources. When planning routes for low-priority non-TT flows, we considered changing their routing scheme from simply taking the shortest routing approach. When there were already higher

priority flows causing queue congestion, we considered “bypassing” the congested SWs in exchange for a relative balance of SW loads at the cost of a partial loss of route length.

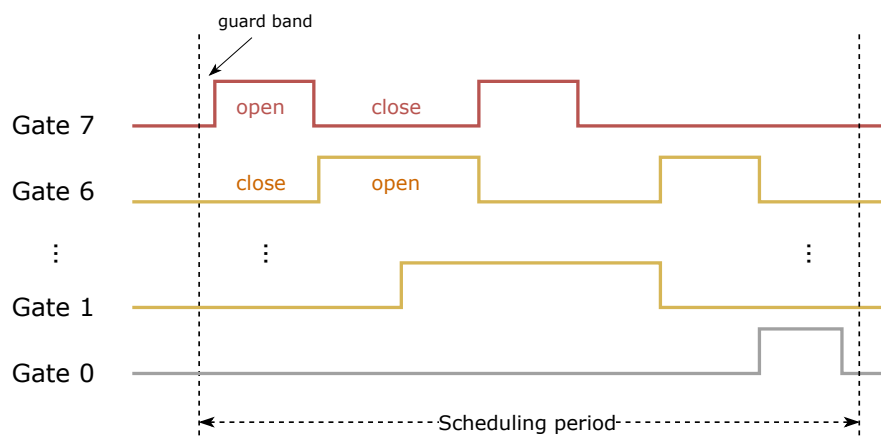


Figure 7. An example of TAS scheduling timing in a TSN (corresponds to Figure 8).

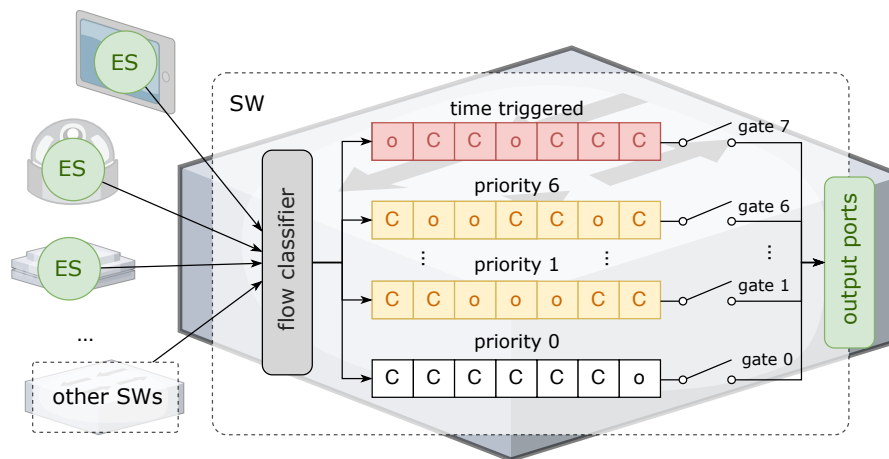


Figure 8. TAS scheduling mechanism with GCLs in a TSN.

5.1. Definition of the Optimization Problem in Non-TT Routing

Based on the above conditions, the non-TT routing problem in TSNs could be defined as follows: in a given $G(V, E)$ with n nodes, the planning of m routes for m non-TT flows with the sets of source nodes of these routes was defined as $SRC = \{v_{src_1}, v_{src_2}, \dots, v_{src_m}\}$, where the nodes are all repeatable. The destination of all non-TT flows was a centralized data server node v_{dst} . The priority of the k th non-TT flow was p_k , which was also repeatable, and the set of its packet lengths was $PCK = \{pck_1, pck_2, \dots, pck_m\}$. The route of the k th non-TT flow was denoted as $\omega_k = (v_{src_k}, v_2, v_3, \dots, v_{dst})$, which represented a path passing through n nodes of $v_{src_k} \rightarrow v_2 \rightarrow v_3 \rightarrow \dots \rightarrow v_{dst}$.

We retained the description of the time delay from Section 4.1, with the difference that for this problem, the non-TT flow did not need to traverse each ES node inside the subnetworks, so the time delay of the k th non-TT flow between v_i, v_j nodes in the routes ω_k was:

$$d_{ij}^k = c_{ij} + q_{ij}^k, \tag{8}$$

where q_{ij}^k is the queuing delay of the k th non-TT flow between v_i, v_j . The higher the number of non-TT flows involved in queuing on any single SW node, the larger the WCD of the flows with a higher priority than the current k th flow, thereby making q_{ij}^k increase correspondingly. The correspondence between the number of non-TT flows in a queue

and q_{ij}^k was determined by the GCL-based scheduling mechanism in the TSN, which was obtained through the simulation experiments detailed in later sections of this paper.

Since the traffic scheduling principle of a TSN requires planning higher priority non-TT flows first, the ω_i with the highest p_i had to be prioritized, followed by planning the rest of flows in order of priority. The minimum value of the overall time delay T was:

$$T = \sum_{k=1}^m \sum_{i=1}^n \sum_{j=1}^n d_{ij} \cdot \rho_{ij}^k, \quad (9)$$

where ρ_{ij}^k is the transmission direction of the non-TT frame between nodes v_i and v_j , which was defined as follows:

$$\rho_{ij}^k = \begin{cases} 1, & \text{non-TT frame } k \text{ goes from } i \text{ to } j \\ 0, & \text{non-TT frame } k \text{ goes from } j \text{ to } i \end{cases} \quad (10)$$

In addition, we defined y_i^k to mark whether the k th non-TT frame had visited the node v_i . Similarly, each node v_i was marked one time per non-TT frame arrival; therefore, for every node v_i , there was:

$$y_i^k = \sum_{j=0}^n \rho_{ji}^k, \quad \forall i \in \{1, 2, \dots, n\}, k \in \{1, 2, \dots, m\}. \quad (11)$$

5.2. Optimization Goal and Constraints of Non-TT Routing

When non-TT flows with multiple priorities participate in queuing within the same SW, TAS schedules all priority queues in units of time windows. In a single SW, as the number of flows participating in the priority queue increases, the WCD of each data flow gradually increases, which is brought about by the deterioration of the queuing situation. The optimization goal of the AACO algorithm was to avoid unnecessary queuing as much as possible and, therefore, reduce the overall WCD for all flows, realize the relative balance of the TSN data link loads, and improve the response speed and bandwidth utilization of the non-TT networks, namely:

$$\begin{aligned} & \text{Minimize}\{T\}, \\ \text{s.t. } & \text{(c1): } T_i < T_j, \forall p_i > p_j \\ & \text{(c2): } y_i^k \leq 1, \forall i \neq \text{dst}, k \in \{1, 2, \dots, m\}, \\ & \text{(c3): } \sum_{k=1}^m y_{\text{dst}}^k = m, \\ & \text{(c4): } T \leq \text{MaxTime}. \end{aligned} \quad (12)$$

In Equation (12), (c1)–(c4) are the constraints that needed to be obeyed by any solution. Constraint (c1) ensured that non-TT flows with higher priority always reached the destination earlier, where T_i and T_j are time delay for two different routes ω_i and ω_j . Constraint (c2) ensured that every node v_i , except for v_{dst} , was visited by the k th non-TT flow no more than one time, so that there were no loops in the routes. Constraint (c3) ensured that the m non-TT flows all converged at the destination node v_{dst} . Constraint (c4) ensured that the final result T was less than the preset threshold MaxTime , otherwise its soft real-time performance was unsatisfied.

5.3. Description of AACO

To solve the problem described above, we proposed a novel AACO algorithm to achieve load balancing routing among multi-priority non-TT flows. We set up m ant colonies, each of which represented a non-TT flow. Unlike classical ACO algorithms, there were strong and weak colonies among these ant colonies and the strength of the i th ant colony was equal to the priority p_i of the i th non-TT flow. When the pheromone was

updated, the pheromone left by the ant colonies with higher priorities greatly suppressed the pheromone increment of the weak colonies and reduced the probability of picking identical routes. Weak ant colonies would rather choose detouring than repeating the route of stronger colonies. Meanwhile, the distance to be detoured was also taken into account. When the extra cost of detouring was too large, weak ant colonies would choose to participate in the queuing process with stronger ant colonies, after weighing up the costs. Taking a small network consisting of six SWs with $m = 3$ as an example, a sample procedure of the AACO algorithm solving the non-TT route assignment problem is shown in Figure 9.

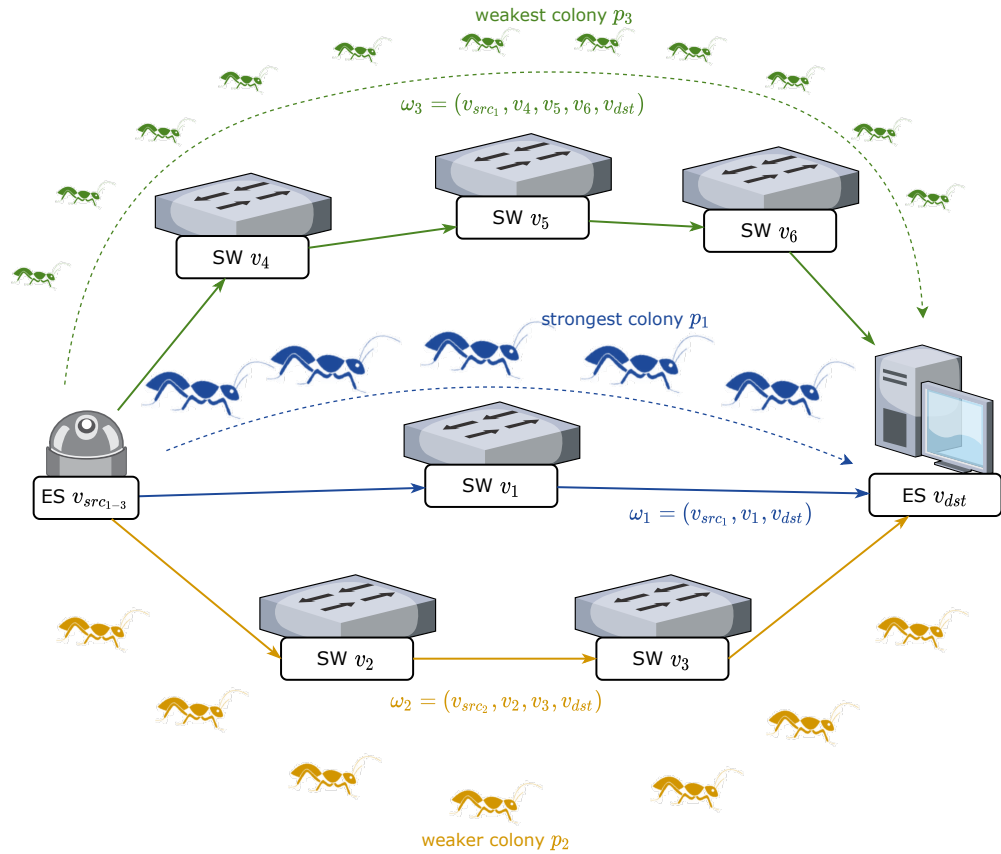


Figure 9. A sample procedure of the AACO algorithm solving the non-TT route assignment problem.

5.3.1. Ant Colony Initialization

For the route assigning task of m non-TT flows, we initialized m ant colonies with Z ants per colony. All ants from the i th colony were located at the corresponding source node src_i . We also initialized m different pheromone concentration values for each edge in E , denoted as τ_{ij}^k :

$$\tau_{ij}^k = C_0, \forall i, j \in \{1, \dots, n\}, \forall k \in \{1, \dots, m\}. \quad (13)$$

5.3.2. State Transition Probability

We then calculated the state transition probability of the ants and selected the next node to visit by roulette, based on the state transition probability. The state transition probability was calculated as follows:

$$P_{ij}^{kl}(t) = \begin{cases} \frac{[\tau_{ij}^k(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{s \in allowed_i} [\tau_{is}^k(t)]^\alpha [\eta_{is}(t)]^\beta}, & j \in allowed_i \\ 0, & otherwise. \end{cases} \quad (14)$$

where $P_{ij}^k(t)$ is the probability that the l th ant in the k th ant colony chooses to visit v_j from the current node v_i at the time t , $allowed_l$ is the set of nodes that are directly accessible and have not been visited yet, $\tau_{ij}^k(t)$ is the pheromone concentration left on the edge e_{ij} by the k th ant colony, and α is the pheromone influence factor and β is the cost function influence factor, both of which are preset constants. The cost function was defined as:

$$\eta_{ij}(t) = \frac{1}{c_{ij} + \delta h_i} \quad (15)$$

where h_i is SW hops between v_i and v_j .

5.3.3. Pheromone Update Rules

In each time window, the pheromone $\tau_{ij}^k(t)$ was volatilized in a fixed ratio. At the same time, the pheromone concentration was increased on the edges, according to the route traveled. For each ant in the k th colony:

$$\tau_{ij}^k(t+1) = (1 - \rho)\tau_{ij}^k(t) + \Delta\tau_{ij}^k(t), \quad (16)$$

where ρ is the volatility coefficient (preset between 0 and 1), $\Delta\tau_{ij}^k(t)$ is the sum of the pheromone increments in the k th population, which is accumulated by the pheromone increment of each ant $\Delta\tau_{ij}^{kl}(t)$, and Q is a preset pheromone increment constant. $\Delta\tau_{ij}^{kl}(t)$ was calculated by:

$$\Delta\tau_{ij}^{kl}(t) = \begin{cases} Q/c_{src_k, dst} & \text{tour}(i, j) \in \text{tour } \omega_l(v_{src_k}, \dots, v_{dst}) \\ 0, & \text{otherwise} \end{cases} \quad (17)$$

Compared to existing ACO algorithms, the main improvement of the ACO in this paper was the calculation of pheromone increment $\Delta\tau_{ij}^k(t)$. The pheromone increment of each ant colony was calculated in decreasing order of priority. When calculating the k th pheromone increment, the pheromone increments of the previous $k - 1$ ant colonies were taken into account. Ant colonies with higher priorities had a stronger suppression effect on the current colony. Therefore, when we calculated $\Delta\tau_{ij}^k(t)$, the sum of the pheromone increments of stronger ant colonies on the path to v_j had a proportional negative impact on the current pheromone increment. $\Delta\tau_{ij}^k(t)$ was calculated by:

$$\Delta\tau_{ij}^k(t) = \begin{cases} \sum_{l=1}^m \Delta\tau_{ij}^{kl}(t) - \gamma \sum_{s=p_k+1}^{max(p)} \Delta\tau_{*j}^s \frac{s \cdot pck_s}{\sum_{h=1}^{max(p)} h \cdot pck_h}, & \text{if } > 0 \\ 0, & \text{otherwise} \end{cases} \quad (18)$$

where γ is a preset constant whose purpose is to adjust the influence of stronger colonies on the current colony, $\Delta\tau_{*j}^s$ is the sum of pheromone increments on the edges of all paths whose priority is s and whose destination is v_j , and $max(p)$ is the maximum value of priority p . The colonies with higher priorities than the k th colony were $(p_k + 1, \dots, max(p))$.

5.4. Algorithm Flow

Based on the above theory, the pseudo-code of the AACO algorithm proposed in this paper is described in Algorithm 2.

Algorithm 2: AACO algorithm for non-TT flow route planning.

```

Input:  $m, \{d_{ij}^k\}, p_{1,\dots,m}, PCK, SRC, G(V, E)$ 
Output:  $\omega_1, \omega_2, \dots, \omega_m$ 
1 Initialization parameters  $Z, Q, C_0, \alpha, \beta, \gamma$ ;
2 Establish  $m$  two-dimensional pheromone matrices and combine them into a
  three-dimensional pheromone array  $[V \times V, m] = C_0$ ;
3 Initialize  $k$  colonies with  $Z$  ants;
4 for iterator from 1 to  $I_{max}$  do
5   forall  $m$  ant populations do
6     foreach ant in colony do
7       if The node is not the destination node  $dst$  then
8         Add the current node to the tabu list;
9         Get the set of adjacent nodes, exclude nodes in the tabu list;
10        Calculate state transition probabilities according to Equation (14);
11        Visit next node by roulette, record the path;
12      else
13        Return to the source node  $src$  and collect path set;
14        Calculate the total delay  $c_{src,dst}$  of the path ;
15      end
16    end
17    foreach edge  $e \in E$  do
18      Calculate the pheromone increments according to Equation (17,18);
19      Update pheromone according to Equation (16);
20    end
21  end
22 end
23 for  $i$  from 1 to  $m$  do
24   Add the nodes in the path set to  $\omega_i$ ;
25 end

```

5.5. Working Pattern of AACO

We added a non-TT routing program based on the AACO into the central control server of the smart factory TSN to optimize non-TT routing. When a new non-TT application sent a registration message to apply to set up a data flow with the data server, the transmission continued normally while the AACO process was created to perform offline route planning. The ES node where the application was running was treated as the source node v_{src_k} and the delay cost between it and other nodes was calculated to generate the new $\{d_{ij}^k\}$. The updated $G(V, E)$, $\{d_{ij}^k\}$, and the priority of this data flow p_k were input to the AACO process to calculate m new routes. Starting from the next time period, all non-TT flows were transmitted with m new routes to complete the registration of the new non-TT application.

5.6. Convergence of AACO

AACO is also based on a heuristic algorithm and the average time complexity of its convergence is complex and can be influenced by various factors, including the number of non-TT flows, number of ants, number of iterations, and randomness. Thus, it was difficult to produce an accurate expression. For its convergence, we could only estimate its time complexity as $O(m \times Z \times n \times I_{max})$.

5.7. Key Innovations and Contributions of AACO

For the non-TT routing problem in TSNs, one of the contributions of this paper is the selection of ACO as the basis of algorithm optimization. By reasonably adjusting the impact factor of the pheromones of stronger ant colonies on the update of the pheromones

of other ant colonies, the improved ACO could effectively balance the relationship between the greedy principle and load balancing in order to better adapt to the TSN traffic shaping mechanisms, which is harder or even impossible for other algorithms to handle.

6. Performance Evaluation

In this section, we present simulation experiment environment that was used to evaluate the performances of the algorithms proposed in this paper when adopted in the routing scenarios of large-scale TSN networks in smart factories.

6.1. Single SW Traffic Scheduling Experiment Based on NeSTiNg

Currently, there are two mainstream experimental frameworks in the field of TSN research, namely NeSTiNg [50] and CORE4INET [51]. NeSTiNg is an open-source project on the GitLab website [52], which was released in 2019 specifically for TSN simulation. Using the NeSTiNg simulation framework, Luxi Zhao et al. [53–56] accomplished a series of research works on network calculus-based TSN network latency analysis and optimization. In order to prepare the raw data on the time delay from a single SW for the routing algorithm proposed in this paper, we built a TSN simulation experimental environment based on NeSTiNg and set up the network topology depicted in Figure 10.

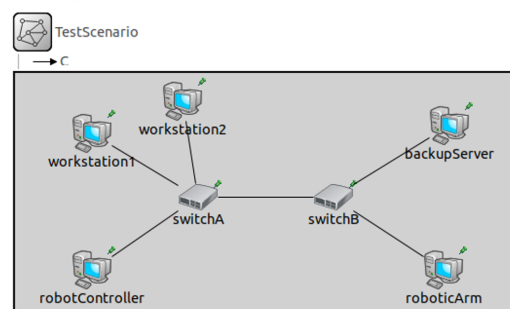


Figure 10. TSN topology of the simulation experiment that was built based on NeSTiNg.

On the experimental network, we deployed the end-to-end latency optimization methods proposed in [53,54]. For all flows that were transmitted through switchA, we changed the quantity of TT and non-TT flows participating in queuing and performed several sets of comparative experiments to quantify the correspondence between the WCD and queuing congestion, as depicted in Figure 11. Based on these data, we were able to calculate the correspondence between the queuing delay q_{ij}^k and the quantity of flows to be transmitted to v_j when the routes of non-TT flows overlapped, which was previously discussed in Section 5.1.

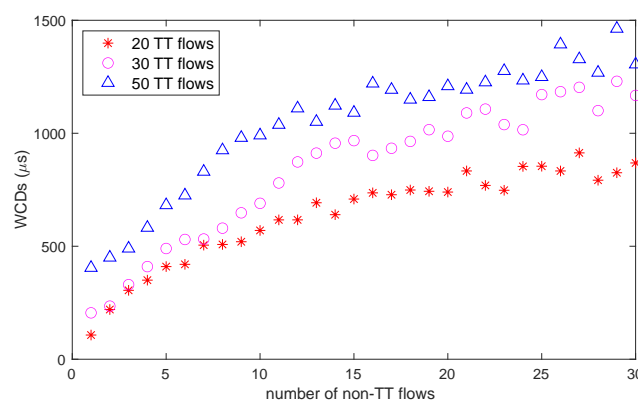


Figure 11. WCD changes with the quantity of flows participating in the queue.

6.2. TT Flow Routing Experiment Based on GATTRP

In order to verify the effectiveness of the GATTRP proposed in this paper, we built a simulation experiment environment for the TT routing problem on the MATLAB® R2021b software platform and developed a test program based on Algorithm 1. The simulation program is available online at [57]. We simulated a factory with 30 TT subnetworks, in which each subnetwork was abstracted as a node. When visualizing the output results, the central control server was marked with a red pentagram, the traversal routes of m TT frames were represented by line segments, and different routes were labeled with different colors. Finally, on the longest route, all propagation and processing delays that were incurred while passing through each subnetwork were summed to calculate the final output T . The preset simulation parameters are shown in Table 2. We ran the experiments 30 times for each value of m and took the average value as the final result for each statistic in order to avoid misguidance by randomness. Figure 12a–c present some of the routing results obtained during the experiments.

Table 2. TT flow routing simulation parameters.

Symbol	Value	Description	Remarks
n	30	The number of nodes to be traversed	Can be selected by the user
U_0	100	The size of the initial population	Can be selected by the user
U_{max}	150	The maximum size of the chromosome population	Can be selected by the user
I_{max}	1500	The maximum number of iterations	Parameter of the algorithm
P_{mut}	40%	The probability of mutation	Parameter of the algorithm
P_{flip}	25%	The probability of flip mutation	Parameter of the algorithm
P_{slide}	25%	The probability of slide mutation	Parameter of the algorithm
P_{drift}	50%	The probability of drift mutation	Parameter of the algorithm

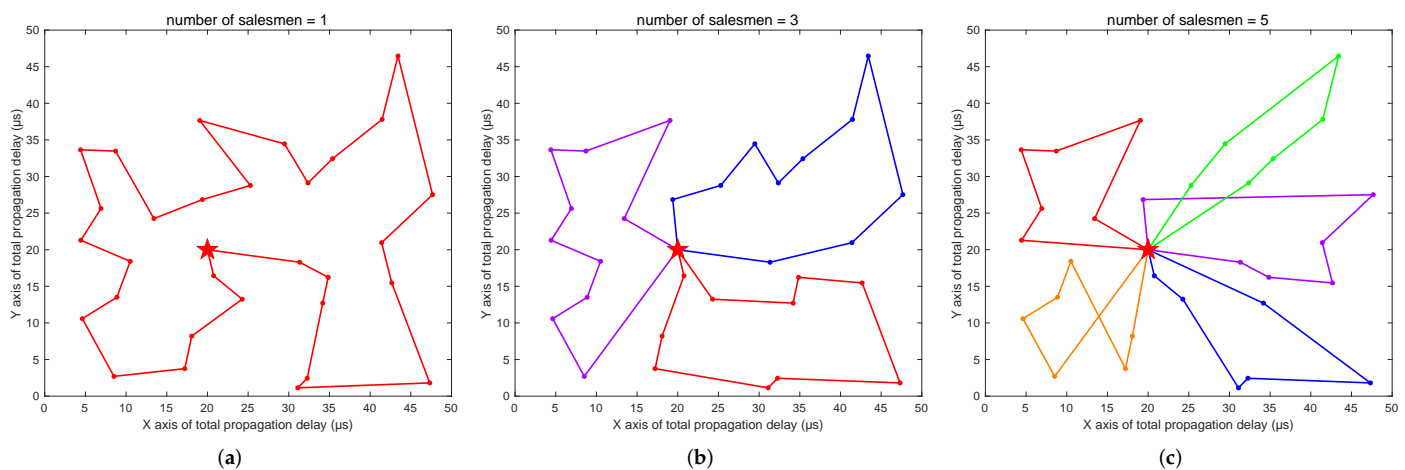


Figure 12. Comparison of traversal routes when the number of salesmen changed: (a) traversal route of one salesman; (b) traversal routes of three salesmen; (c) traversal routes of five salesmen.

As the TT flows traversed through each SW, they consumed a certain amount of processing delay. We set a random processing delay for each SW with maximum values of $10 \mu\text{s}$, $15 \mu\text{s}$, and $20 \mu\text{s}$ to simulate the change of the optimization target $\min(T)$ using Algorithm 1 when the number of salesmen m changed. The results are shown in Figure 13.

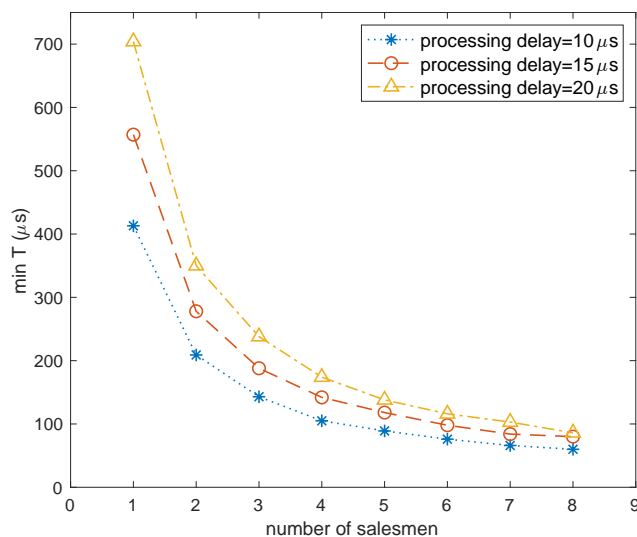


Figure 13. The effect of changing the number of salesmen on $min(T)$ when using Algorithm 1.

From the experimental results, we could confirm that it was feasible and effective to solve the routing problem of TT nodes based on the idea of solving the MTSP. With an increase in the number of salesmen, SW hops on the routes decreased and the propagation and processing delays on the longest route were correspondingly reduced. Considering that the number of parallel ingress and egress ports of the central control server is limited and that network wiring has comprehensive limitations from environment and financial costs, the value of m had to be properly set within a reasonable range. In addition, we observed that the higher the processing delay of the subnetworks, the better the optimization effect. Therefore, the larger the overall network size, the more obvious the optimization effect of the GATTRP.

To illustrate the operational efficiency of Algorithm 1, we conducted comparative experiments using several other algorithms. Figure 14 shows GVNS, which is the general variable neighborhood search algorithm proposed by Soylu [58], GA2PC, which is the two-segment GA proposed by Carter et al. [34], TCX, which is the improved GA with a two-part chromosome crossover operator proposed by Yuan et al. [59], and GATTRP, which is Algorithm 1 from this paper. All four algorithms were tested using the same $G(V, E)$ with $m = 5$ and a processing delay of up to $20\mu s$. The convergence curves are shown in Figure 14. The results showed that GATTRP converged faster and the final result after convergence was better.

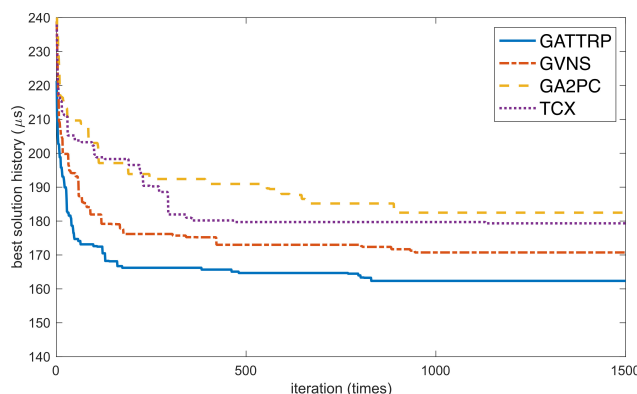


Figure 14. Convergence curves of the four algorithms when solving the same MTSP problem.

6.3. Non-TT Flow Routing Experiment

In order to verify the effectiveness of Algorithm 2, as proposed in this paper, we developed a non-TT routing test program, which is available online at [57]. The simulation

parameters are shown in Table 3. Likewise, we performed each experiment 30 times and took the average result as the final result to minimize misguidance caused by randomness.

We simulated a case of sending m non-TT flows from industrial sensors to a data server and compared the AACO to two other routing algorithms, as depicted in Figure 15. To make the visualization more intuitive, we chose the smaller m value of 5 and set the starting and end nodes as the two farthest nodes within the network. The starting node was labeled with a diamond and the end node with a pentagram. The route of each non-TT flow was represented by a line segment, which were distinguished from each other by color.

Table 3. The non-TT routing simulation parameters.

Symbol	Value	Description	Remarks
n	100	The number of SWs in the non-TT network	Can be selected by the user
Z	80	The number of ants in each colony	Parameter of the algorithm
Q	2	The pheromone increment constant	Parameter of the algorithm
C_0	2	The initial pheromone constant	Parameter of the algorithm
I_{max}	800	The maximum number of iterations	Parameter of the algorithm
α	1	The pheromone impact factor	Parameter of the algorithm
β	5	The heuristic impact factor	Parameter of the algorithm
ρ	0.5	The pheromone volatile factor	Parameter of the algorithm
γ	0.05	The stronger ant colony impact factor	Parameter of the algorithm

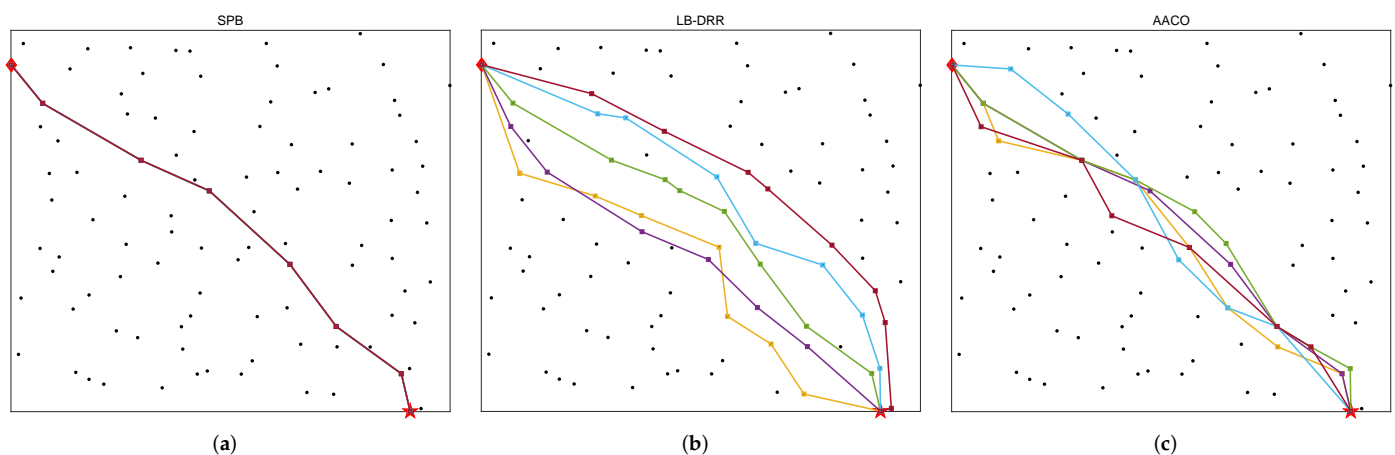


Figure 15. Comparison of the three algorithms for non-TT routing: (a) routes assigned by the SPB; (b) routes assigned by the LB-DRR; (c) routes assigned by the AACO.

We developed a routing program based on the SPB algorithm proposed by [20] and the route obtained is depicted in Figure 15a. Since the routes planned by SPB for each non-TT flow are always the shortest solution, when the source node and the destination node were the same for all m flows, the obtained multiple paths completely coincided. In addition, we made appropriate modifications for our problem model based on the LB-DRR method proposed by Ojewale et al. [19]. The results of the LB-DRR algorithm are depicted in Figure 15b. For LB-DRR, the routes never overlapped with other routes when there were other options. In contrast, the route scheme generated by Algorithm 2 from this paper was more balanced and reasonable.

When a flow chose to detour, the more SWs the route passes through, the longer the processing and propagation delays. We defined the sum of processing and propagation delays as the detouring delay. In addition, from the simulation results presented in Section 6.1, we obtained the correspondence between queuing delay and the number of queuing flows, so we could estimate the queuing delay. We counted the average total delay from multiple experiments to evaluate the algorithm performances, as depicted in Figure 16. The results showed that the SPB, which pays attention to the shortest path, had

the best detouring delay but the queuing delay was too long, resulting in the longest total delay. The LB-DRR, which focuses on load balancing, had the best queuing delay but due to too many detours, results were also unsatisfactory. The AACO algorithm proposed in this paper comprehensively considered a balanced distribution along with the optimization of routes; thus, the total delay was the shortest and the optimization effect was the best.

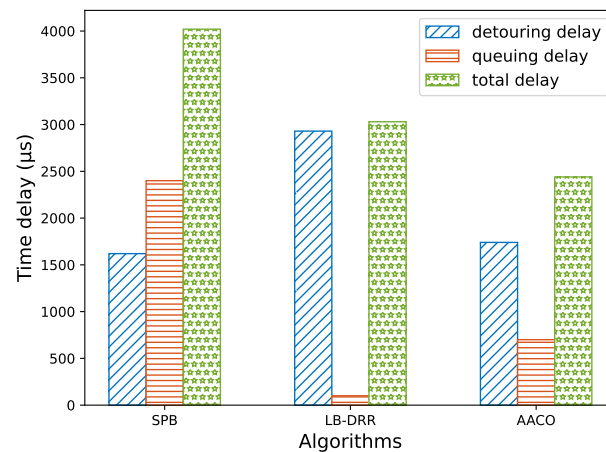


Figure 16. Comparison of the three algorithms when solving the same non-TT routing problem.

7. Conclusions

This paper highlighted the routing problems in TSNs. Different from traditional bus-based real-time communication networks, TSNs have two main routing problems: those for TT flows and those for non-TT flows. To address the specific communication needs of smart factories, we classified network data from smart factories and analyzed the communication requirements and transmission mechanisms of various types of traffic. By mathematically modeling the problem, we transformed the routing problem into an MTSP problem and a load-balanced multi-priority route assignment problem. For the MTSP problem, we proposed an improved algorithm named GATTRP. For the multi-priority route assignment problem, we proposed an improved algorithm named AACO. The simulation results showed that transforming the traversal problem of TT flows into an MTSP can effectively reduce the traversal cycle time and that the GATTRP proposed in this paper has a strong convergence performance. For non-TT flows, the AACO proposed in this paper is more comprehensive and exhibits an excellent performance and better results. However, there were also some limitations in this paper. For upper-layer industrial applications, the transmission mechanisms and requirements of AVB flows and BE flows, which belong to the same non-TT flow type, are not exactly the same. If they can be distinguished for refined routing, the routing performance could be further improved. In the future, we will focus on improving the routing algorithms for AVB flows.

Author Contributions: Conceptualization, Z.Y. and Y.L.; methodology, Z.Y. and Y.L.; software, Y.L.; validation, Z.Y., Y.L., and Y.M.; investigation, Y.L. and H.Y.; data curation, Y.M. and F.X.; writing—original draft preparation, Y.L.; writing—review and editing, Z.Y., Y.L., G.H., and Y.B.; visualization, Y.M. and F.X.; supervision, Z.Y., Y.L., and H.Y.; project administration, Z.Y., G.H., and Y.B.; funding acquisition, Z.Y. and G.H. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the National Key R&D Program of China under grant number 2017YFE0125300.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author. They are restricted to experimental results.

Acknowledgments: This work was supported by the National Key R&D Program of China under grant number 2017YFE0125300.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Frank, A.G.; Dalenogare, L.S.; Ayala, N.F. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *Int. J. Prod. Econ.* **2019**, *210*, 15–26. [[CrossRef](#)]
2. Patera, L.; Garbugli, A.; Bujari, A.; Scotece, D.; Corradi, A. A Layered Middleware for OT/IT Convergence to Empower Industry 5.0 Applications. *Sensors* **2022**, *22*, 190. [[CrossRef](#)] [[PubMed](#)]
3. Messenger, J.L. Time-Sensitive Networking: An Introduction. *IEEE Commun. Stand. Mag.* **2018**, *2*, 29–33. [[CrossRef](#)]
4. Fedullo, T.; Morato, A.; Tramarin, F.; Rovati, L.; Vitturi, S. A Comprehensive Review on Time Sensitive Networks with a Special Focus on Its Applicability to Industrial Smart and Distributed Measurement Systems. *Sensors* **2022**, *22*, 1638. [[CrossRef](#)]
5. Larrañaga, A.; Lucas-Estañ, M.C.; Martínez, I.; Val, I.; Gozalvez, J. Analysis of 5G-TSN Integration to Support Industry 4.0. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; Volume 1, pp. 1111–1114. [[CrossRef](#)]
6. Bruckner, D.; Stanica, M.P.; Blair, R.; Schriegel, S.; Kehrer, S.; Seewald, M.; Sauter, T. An Introduction to OPC UA TSN for Industrial Communication Systems. *Proc. IEEE* **2019**, *107*, 1121–1131. [[CrossRef](#)]
7. Nasrallah, A.; Thyagaturu, A.S.; Alharbi, Z.; Wang, C.; Shao, X.; Reisslein, M.; ElBakoury, H. Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 88–145. [[CrossRef](#)]
8. Li, Y.; Jiang, J.; Lee, C.; Hong, S.H. Practical Implementation of an OPC UA TSN Communication Architecture for a Manufacturing System. *IEEE Access* **2020**, *8*, 200100–200111. [[CrossRef](#)]
9. ISO/IEC/IEEE 8802-1AS:2021(E); IEEE/ISO/IEC International Standard for Information Technology–Telecommunications and Information Exchange between Systems–Local and Metropolitan Area Networks–Part 1AS:Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks. IEEE: New York, NY, USA, 2021; pp. 1–422. [[CrossRef](#)]
10. IEEE Std 802.1BA-2021; IEEE Standard for Local and Metropolitan Area Networks–Audio Video Bridging (AVB) Systems. IEEE: New York, NY, USA, 2021; pp. 1–45. [[CrossRef](#)]
11. IEEE P1722.1/D13; IEEE Approved Draft Standard for Device Discovery, Connection Management, and Control Protocol for Time-Sensitive Networking Systems. IEEE: New York, NY, USA, 2021; pp. 1–474.
12. Wang, Y.M.; Yang, S.S.; Ren, X.B.; Zhao, P.; Zhao, C.; Yang, X.Y. IndustEdge: A Time-Sensitive Networking Enabled Edge-Cloud Collaborative Intelligent Platform for Smart Industry. *IEEE Trans. Ind. Inform.* **2022**, *18*, 2386–2398. [[CrossRef](#)]
13. Yu, W.; Liu, Y.; Dillon, T.; Rahayu, W.; Mostafa, F. An Integrated Framework for Health State Monitoring in a Smart Factory Employing IoT and Big Data Techniques. *IEEE Internet Things J.* **2022**, *9*, 2443–2454. [[CrossRef](#)]
14. Seijo, Ó.; Iturbe, X.; Val, I. SHARP: Implementation of a Hybrid Wired-Wireless TSN Network to Enable Flexible Smart Factories. In Proceedings of the 2021 17th IEEE International Conference on Factory Communication Systems (WFCS), Linz, Austria, 9–11 June 2021; pp. 95–98. [[CrossRef](#)]
15. Hellmanns, D.; Glavackij, A.; Falk, J.; Hummen, R.; Kehrer, S.; Dürr, F. Scaling TSN Scheduling for Factory Automation Networks. In Proceedings of the 2020 16th IEEE International Conference on Factory Communication Systems (WFCS), Porto, Portugal, 27–29 April 2020; pp. 1–8. [[CrossRef](#)]
16. Li, Y.; Ma, Y.; Yin, Z.; Gu, A.; Xu, F. A Communication Model to Enhance Industrial Wireless Networks based on Time-Sensitive Networks. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 363–367. [[CrossRef](#)]
17. Li, M.; Yin, Z.; Ma, Y.; Wang, C.; Chai, A.; Lian, M. Design and verification of secure communication scheme for industrial IoT intelligent production line system with multi-path redundancy and collaboration. *Neural Comput. Appl.* **2021**. [[CrossRef](#)]
18. Chai, A.; Ma, Y.; Yin, Z.; Li, M. Real-Time Communication Model Based on OPC UA Wireless Network for Intelligent Production Line. *IEEE Access* **2021**, *9*, 102312–102326. [[CrossRef](#)]
19. Ojewale, M.A.; Yomsi, P.M. Routing heuristics for load-balanced transmission in TSN-based networks. *SIGBED Rev.* **2020**, *16*, 20–25. [[CrossRef](#)]
20. IEEE Std 802.1Qca-2015 (Amendment to IEEE Std 802.1Q-2014 as Amended by IEEE Std 802.1Qcd-2015 and IEEE Std 802.1Q-2014/Cor 1-2015); IEEE Standard for Local and Metropolitan Area Networks— Bridges and Bridged Networks—Amendment 24: Path Control and Reservation. IEEE: New York, NY, USA, 2016; pp. 1–120. [[CrossRef](#)]
21. Schweissguth, E.; Danielis, P.; Timmermann, D.; Parzyjegl, H.; Mühl, G. ILP-Based Joint Routing and Scheduling for Time-Triggered Networks. In Proceedings of the 25th International Conference on Real-Time Networks and Systems (RTNS '17), Grenoble, France, 4–6 October 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 8–17. [[CrossRef](#)]
22. Falk, J.; Dürr, F.; Rothermel, K. Exploring Practical Limitations of Joint Routing and Scheduling for TSN with ILP. In Proceedings of the 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hakodate, Japan, 28–31 August 2018; pp. 136–146. [[CrossRef](#)]

23. Mahfouzi, R.; Aminifar, A.; Samii, S.; Rezine, A.; Eles, P.; Peng, Z. Stability-aware integrated routing and scheduling for control applications in Ethernet networks. In Proceedings of the 2018 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 682–687. [\[CrossRef\]](#)
24. Nayak, N.G.; Dürr, F.; Rothermel, K. Time-Sensitive Software-Defined Network (TSSDN) for Real-Time Applications. In Proceedings of the 24th International Conference on Real-Time Networks and Systems (RTNS '16), Brest, France, 19–21 October 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 193–202. [\[CrossRef\]](#)
25. Nayak, N.G.; Dürr, F.; Rothermel, K. Incremental Flow Scheduling and Routing in Time-Sensitive Software-Defined Networks. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2066–2075. [\[CrossRef\]](#)
26. Precup, R.E.; David, R.C.; Petriu, E.M.; Preitl, S.; Paul, A.S. Gravitational Search Algorithm-Based Tuning of Fuzzy Control Systems with a Reduced Parametric Sensitivity. In *Soft Computing in Industrial Applications, Proceedings of the 15th Online World Conference on Soft Computing in Industrial Applications*; Springer: Berlin/Heidelberg, Germany, pp. 141–150.
27. Li, X.; Chen, L.; Tang, Y. HARD: Bit-Split String Matching Using a Heuristic Algorithm to Reduce Memory Demand. *Rom. J. Inf. Sci. Technol.* **2020**, *23*, T94–T105.
28. Zamfirache, I.A.; Precup, R.E.; Roman, R.C.; Petriu, E.M. Policy Iteration Reinforcement Learning-based control using a Grey Wolf Optimizer algorithm. *Inf. Sci.* **2022**, *585*, 162–175. [\[CrossRef\]](#)
29. Pozna, C.; Precup, R.E.; Horvath, E.; Petriu, E.M. Hybrid Particle Filter-Particle Swarm Optimization Algorithm and Application to Fuzzy Controlled Servo Systems. *IEEE Trans. Fuzzy Syst.* **2022**. [\[CrossRef\]](#)
30. Yin, Z.; Xu, F.; Li, Y.; Fan, C.; Zhang, F.; Han, G.; Bi, Y. A Multi-Objective Task Scheduling Strategy for Intelligent Production Line Based on Cloud-Fog Computing. *Sensors* **2022**, *22*, 1555. [\[CrossRef\]](#)
31. Niendorf, M.; Girard, A.R. Exact and Approximate Stability of Solutions to Traveling Salesman Problems. *IEEE Trans. Cybern.* **2018**, *48*, 583–595. [\[CrossRef\]](#)
32. Bektas, T. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega-Int. J. Manag. Sci.* **2006**, *34*, 209–219. [\[CrossRef\]](#)
33. Venkatesh, P.; Singh, A. Two metaheuristic approaches for the multiple traveling salesperson problem. *Appl. Soft Comput.* **2015**, *26*, 74–89. [\[CrossRef\]](#)
34. Carter, A.E.; Ragsdale, C.T. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *Eur. J. Oper. Res.* **2006**, *175*, 246–257. [\[CrossRef\]](#)
35. Zhou, H.; Song, M.; Pedrycz, W. A comparative study of improved GA and PSO in solving multiple traveling salesmen problem. *Appl. Soft Comput.* **2018**, *64*, 564–580. [\[CrossRef\]](#)
36. Lu, Z.; Zhang, K.; He, J.; Niu, Y. Applying K-means Clustering and Genetic Algorithm for Solving MTSP. In *Bio-Inspired Computing—Theories and Applications*; Gong, M., Pan, L., Song, T., Zhang, G., Eds.; Springer: Singapore, 2016; pp. 278–284.
37. Heinzelman, W.B.; Chandrakasan, A.P.; Balakrishnan, H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Trans. Wirel. Commun.* **2002**, *1*, 660–670. [\[CrossRef\]](#)
38. Younis, O.; Fahmy, S. HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Trans. Mob. Comput.* **2004**, *3*, 366–379. [\[CrossRef\]](#)
39. Tarhani, M.; Kaviani, Y.S.; Siavoshi, S. SEECH: Scalable Energy Efficient Clustering Hierarchy Protocol in Wireless Sensor Networks. *IEEE Sens. J.* **2014**, *14*, 3944–3954. [\[CrossRef\]](#)
40. Bhushan, B.; Sahoo, G. FLEAC: Fuzzy Logic-based Energy Adequate Clustering Protocol for Wireless Sensor Networks using Improved Grasshopper Optimization Algorithm. *Wirel. Pers. Commun.* **2022**, *124*, 573–606. [\[CrossRef\]](#)
41. Sert, S.A.; Bagci, H.; Yazici, A. MOFCA: Multi-objective fuzzy clustering algorithm for wireless sensor networks. *Appl. Soft Comput.* **2015**, *30*, 151–165. [\[CrossRef\]](#)
42. Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [\[CrossRef\]](#)
43. Ramamoorthy, R.; Thangavelu, M. An enhanced hybrid ant colony optimization routing protocol for vehicular ad-hoc networks. *J. Ambient. Intell. Humaniz. Comput.* **2021**. [\[CrossRef\]](#)
44. Belgaum, M.R.; Ali, F.; Alansari, Z.; Musa, S.; Alam, M.M.; Mazliham, M.S. Artificial Intelligence Based Reliable Load Balancing Framework in Software-Defined Networks. *CMC—Comput. Mater. Contin.* **2022**, *70*, 251–266. 018211. [\[CrossRef\]](#)
45. Govardhan, P.; Srinivasan, P. Multilevel controller-assisted intrinsically modified ant colony optimization heuristic-based load-balancing model for mega cloud infrastructures. *Int. J. Commun. Syst.* **2022**, *35*, e5091. [\[CrossRef\]](#)
46. Pop, P.; Raagaard, M.L.; Craciunas, S.S.; Steiner, W. Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks. *IET Cyber-Phys. Syst. Theory Appl.* **2017**, *1*, 86–94. [\[CrossRef\]](#)
47. Maxim, D.; Song, Y.Q. Delay Analysis of AVB Traffic in Time-Sensitive Networks (TSN). In Proceedings of the 25th International Conference on Real-Time Networks and Systems (RTNS '17), Grenoble, France, 4–6 October 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 18–27. [\[CrossRef\]](#)
48. Val, I.; Seijo, O.; Torrego, R.; Astarloa, A. IEEE 802.1AS Clock Synchronization Performance Evaluation of an Integrated Wired-Wireless TSN Architecture. *IEEE Trans. Ind. Inform.* **2022**, *18*, 2986–2999. [\[CrossRef\]](#)

49. IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as Amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015); IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic. IEEE: New York, NY, USA, 2016; pp. 1–57. [[CrossRef](#)]
50. Falk, J.; Hellmanns, D.; Carabelli, B.; Nayak, N.; Dürr, F.; Kehrer, S.; Rothermel, K. NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++. In Proceedings of the 2019 International Conference on Networked Systems (NetSys), Munich, Germany, 18–21 March 2019; pp. 1–8. [[CrossRef](#)]
51. Steinbach, T.; Kenfack, H.D.; Korf, F.; Schmidt, T.C. An Extension of the OMNeT++ INET Framework for Simulating Real-Time Ethernet with High Accuracy. In Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques (SIMU-Tools '11), Barcelona, Spain, 21–25 March 2011; ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Brussels, Belgium, 2011; pp. 375–382. [[CrossRef](#)]
52. Institute of Parallel and Distributed Systems, University of Stuttgart. NeSTiNg Project Repository. 2019. Available online: <https://gitlab.com/ipvs/nesting> (accessed on 1 April 2022).
53. Zhao, L.X.; Pop, P.; Gong, Z.J.; Fang, B.W. Improving Latency Analysis for Flexible Window-Based GCL Scheduling in TSN Networks by Integration of Consecutive Nodes Offsets. *IEEE Internet Things J.* **2021**, *8*, 5574–5584. 3031932. [[CrossRef](#)]
54. Zhao, L.X.; Pop, P.; Zheng, Z.; Daigmorte, H.; Boyer, M. Latency Analysis of Multiple Classes of AVB Traffic in TSN With Standard Credit Behavior Using Network Calculus. *IEEE Trans. Ind. Electron.* **2021**, *68*, 10291–10302. [[CrossRef](#)]
55. Zhao, L.; Pop, P.; Craciunas, S.S. Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus. *IEEE Access* **2018**, *6*, 41803–41815. [[CrossRef](#)]
56. Zhao, L.; Pop, P.; Zheng, Z.; Li, Q. Timing Analysis of AVB Traffic in TSN Networks Using Network Calculus. In Proceedings of the 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Porto, Portugal, 11–13 April 2018; pp. 25–36. [[CrossRef](#)]
57. Simulation Project Repository of This Paper. 2022. Available online: <https://gitee.com/LiYueUCAS/Heuristic-Routing-Algorithms-for-TSN.git> (accessed on 2 May 2022).
58. Soylu, B. A general variable neighborhood search heuristic for multiple traveling salesmen problem. *Comput. Ind. Eng.* **2015**, *90*, 390–401. [[CrossRef](#)]
59. Yuan, S.; Skinner, B.; Huang, S.; Liu, D. A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. *Eur. J. Oper. Res.* **2013**, *228*, 72–82. [[CrossRef](#)]