scientific reports

Check for updates

OPEN

Research on routing and scheduling algorithms for the simultaneous transmission of diverse data streaming services on the industrial internet

Yan Song[1,2]✉, Chenyang Guo[1], Panfeng Xu[1], Lina Li[1] & Rui Zhang[1]

OPC UA PubSub Over TSN is the core of the Industrial Internet and guarantees flexible interaction features for multiple parties in real-time for industrial communication. To achieve the transmission of time-triggered traffic in PubSub NetworkMessage, routing and scheduling data need to be analyzed. Traditional routing and scheduling methods have disadvantages such as low calculation efficiency, slow convergence speed, and poor reliability. Therefore, a routing and scheduling method for OPC UA PubSub NetworkMessage time-triggered traffic based on an improved ant colony algorithm is proposed. First, we analyze the network topology model, traffic model, and traffic transmission constraints of TSN; then, we apply the K-means clustering algorithm, the KSP algorithm based on the shortest path idea, and an improved ant colony algorithm for traffic classification, routing, and scheduling calculation. Experimental results show that this method can effectively reduce the delay increase caused by link congestion, improve the ability to schedule time-triggered traffic, and accelerate the convergence rate of iteration.

With the development of the Industrial Internet[1], the integration of information technology (IT) and industrial technology (OT) has been greatly accelerated[2]. Smart Factory, a product of the Industrial Internet, promotes the integration of heterogeneous networks with a distributed control system, making simultaneous transmission of data streaming services a reality, including enterprise resource planning (ERP) data, product lifecycle management (PLM) data, factory and workshop manufacturing execution system (MES) data, and office data. In addition, in the transmission of diverse data streaming services, the integration of OPC UA and TSN technologies can effectively ensure the real-time and semantic interoperability of data streams.

OPC UA. OLE for Process Control Unified Architecture (OPC UA)[3], is the standard meeting platform for industry integration and information sharing. It virtualizes physical devices into an OPC UA address space containing multi-sided service architecture (SOA) tags[50]. It integrates data access, alarm information, event information, and historical data into one accessible address space[4]. Each tag contains real-time control data, project specifications, and equipment information. By standardizing the format of data exchange, real-time control, access to the physical equipment can be achieved[1,5]. Access to different devices is designed to ensure interoperability and data integration, ensuring that physical devices communicate seamlessly with the network. In addition, OPC UA has advanced self-organization capabilities. The manufacturing system not only realizes the horizontal integration of access to different devices (PLC, RFID, etc.), but also the vertical integration of SCADA, ERP, and MES data. This transforms the manufacturing system into a decentralized organization[5–7], as shown in Fig. 1.

Time sensitive network. The main purpose of the IEEE802.1 time sensitive network (TSN) is to service time-sensitive projects and systems[8]. According to the requirements of different data streams for quality of service (QoS), it transmits a variety of traffic on the same Ethernet link, including Time-Triggered traffic, Audio–Video Bridge traffic, and Best-Effort traffic. Clock synchronization and transmission scheduling enable

[1]College of Physics, Liaoning University, Shenyang 110000, China. [2]Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110000, China. ✉email: profsongyan@163.com

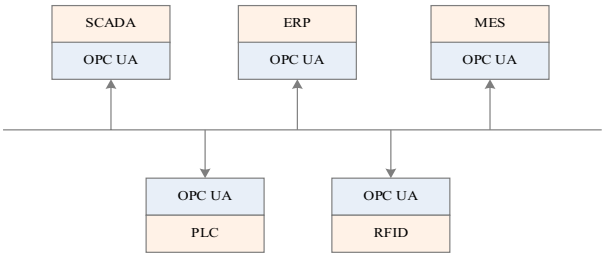

Figure 1. Integration of OPC UA in the automation.

data to be transmitted with the smallest possible delay and jitter to facilitate the coexistence of BE, TT, and AVB traffic[8–10]. At present, commonly used protocols include: IEEE802.1 AS[11], IEEE802.1 Qat[12], IEEE802.1 Qav[13], IEEE802.1 Qbu[14], IEEE802.1 Qbv[15] and IEEE802.1 Qcc[16].

OPC UA over TSN. With the current advancement of the IoT and Industry 4.0, automation manufacturers are eager to evolve from the automation model to the "self-hosted character tower model", combining OPC UA with TSN[17,18]. As the standard application-layer interface for automated manufacturing hardware, OPC UA integrates address space and provides semantic operations to achieve OPC UA server access to data on hardware devices[19]. Furthermore, users can add a TSN tag to the head of the OPC UA data stream to enable it to be transmitted throughout the TSN network. To ensure certainty and reliability, vertical integration from the application layer to the data link layer is created[20].

Work of this paper. In this paper, we propose a method to directly map OPC UA NetworkMessage data from the application layer to the TSN network. To determine the TT traffic of NetworkMessage data, we apply a clustering algorithm to classify NetworkMessage information in a real-time network data set. In addition, we use the BA model to generate a static network topology and combine it with the KSP algorithm to reduce the routing search space. Finally, we thoroughly consider the impact of the transmission constraints of TT traffic on global scheduling performance and apply the improved ant colony algorithm to optimize TT traffic scheduling. Our algorithm converges faster than other available methods and best solves the routing and scheduling problems of OPC UA NetworkMessage on TSN networks.

Related work

In research of OPC UA Over TSN, Bucknerder et al.[21], Tian and Hu[22] developed cross-level communication of the automatic font tower through the establishment of the OPC UA Over TSN model, making network communication horizontal and simplifying the network transmission process. Kobzan et al.[23] proposed a TSN network for OPC UA based on SDN configuration and designed and verified the IEEE802.1 Qbv standard. Andreas Eckhardt and Sebastian Müller developed the round-trip time test for a point-to-point communication process based on a development board, and integrated the two standards of IEEE802.1 Qbv and IEEE802.1AS.

In research of TSN frames routing and scheduling, Pahlevan and Obermaisser proposed a genetic algorithm-based heuristic scheduling method, which improves the efficiency of scheduling, transmission, and link utilization for TT traffic[24]. Sune and Steiner proposed search space reduction technology and a heuristic algorithm based on a greedy randomized adaptive search process to inform AVB traffic of the worst-case end-to-end delay and minimal network utilization[25]. Combining the TT Ethernet protocol, Steiner et al. proposed a meta-heuristic method based on Tabu Search that achieves comprehensive TT traffic and RC traffic scheduling and provides minimal end-to-end delay for RC traffic[26]. Bingqian and Yong et al. proposed a Hybrid-GA-based time-triggered Ethernet static scheduling algorithm, which effectively increases the speed of the scheduling process and improves scheduling flexibility of the TT Ethernet[27]. Wang et al. proposed an improved ant colony algorithm for routing and scheduling of time-triggered traffic, which provides deterministic network delay and jitter for time-triggered traffic transmission[28].

OPC UA over TSN mapping and system model

OPC UA over TSN mapping. OPC UA and TSN are standards for the application layer and data link layer, respectively. The combination of OPC UA and TSN is one of the cores of the Industrial Internet, providing real-time and interoperability for data streams. To ensure that OPC UA NetworkMessage data can be transmitted over the TSN network, the NetworkMessage data must be mapped to the TSN frame. OPC UA provides two communication modes, CS mode and PubSub mode. PubSub mode provides multi-to-multi communication and periodic data transfer between devices. Because the transmission of NetworkMessage data to the TSN network is periodic, we perform the mapping from OPC UA to TSN through PubSub mode[29].Compared with XML and JSON, binary-coded PubSub NetworkMessages are adapted to the transmission environment with high frequency and low bandwidth occupation[29,30]. To provide a security mechanism for data transmission, we send data in a binary-coded mode.

As shown in Fig. 2, the specific mapping process of NetworkMessage data to the TSN data stream is as follows.

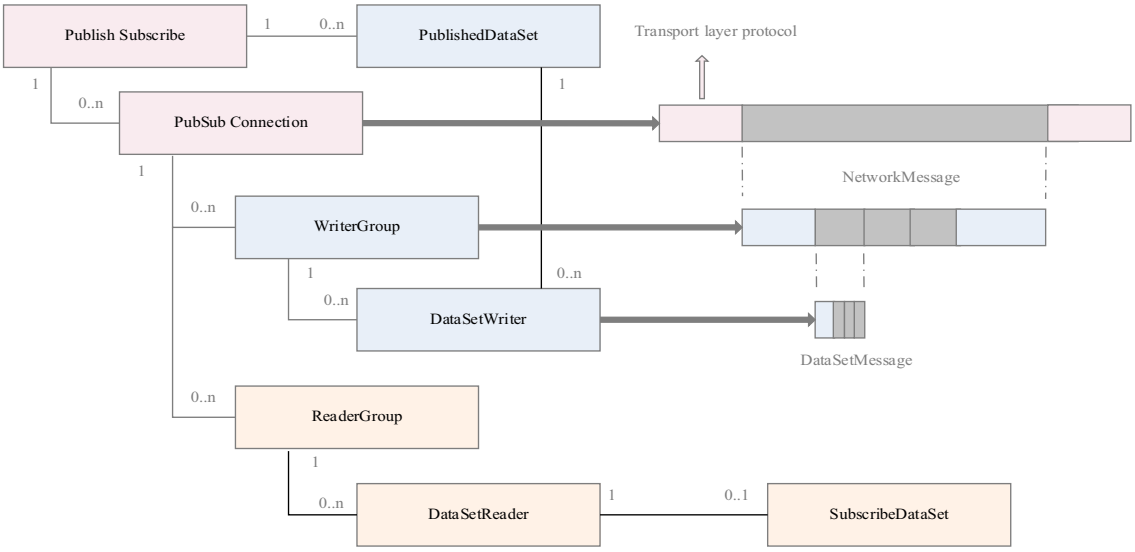

Figure 2. OPC UA PubSub configuration.

OPC UA Over TSN Application Regulations		
Application Layer	OPC UA Information Model	
	OPC UA PubSub	
	UADP	
Session Layer Presentation Layer	OPC UA Over TSN Mapping	OPC UA Over TSN Mapping
Transport Layer	UDP	
Network Layer	IP	
Data Link Layer	TSN	

Figure 3. OPC UA over TSN for converged architecture protocol.

Through configuring OPC UA PubSub, a concrete example was created. PublishSubscrible is the root node of the object information model and contains the data content of PubSub. A NetworkMessage is the object of a WriterGroup. The WriterGroup contains at least one DataSetWriter sub-object, and the DataSet encapsulates the DataSetMessage in the DataSetWriter object and the PublishedDataSet binding. The PublishedDataSet contains metadata and encoding methods that describe the data set. The NetworkMessage is the final payload of the application layer, transmitted to the bottom layer, and finally completing the mapping as the TSN data frame's payload. The mapping from the application layer to the data link layer is divided into two forms, as shown in Fig. 3.

1. Application layer NetworkMessage passes through the transport layer and network layer, and finally maps to the data link layer;
2. Application layer NetworkMessage directly maps to the data link layer;

The message of form (1) needs to be encoded by the UDP at the transport layer and the IP at the network layer. For the purpose of real-time transmission of the OPC UA PubSub data stream, form (2) is used to map the NetworkMessage message to the data. At the link layer, since it bypasses the transport layer and the network layer, our mapping method will provide better real-time guarantee. In completing NetworkMessage mapping, the TSN stream identifier is added to the header according to the IEEE802.1 standard.

As shown in Fig. 4, flow identification involves VLAN ID as the destination of MAC, Submitted to PublishconnectionType for system configuration. The specific Publishconnection object parameters include PublisherID, TransportProfileUri, Address, and ConnectionProperties. Before completing the NetworkMessage mapping, the created DataSet, DataSetMessage, and NetworkMessage configurations are set according to the OPC UA PubSub protocol. I will not repeat them here.

As shown in Fig. 5, first, we collect the information obtained from the address space into the DataSet, and secondly, map it to the DataSetMessage through the DataSetWriter, then one or more groups of DataSetMessage are mapped to the NetworkMessage through the NetWorkMessage WriterGroup, and finally, the NetworkMessage gets the TSN Frame through the TSN Mapping.

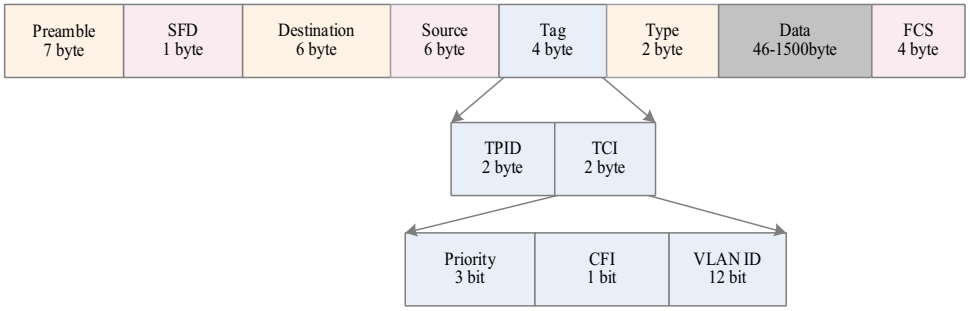

Figure 4. TSN header package.

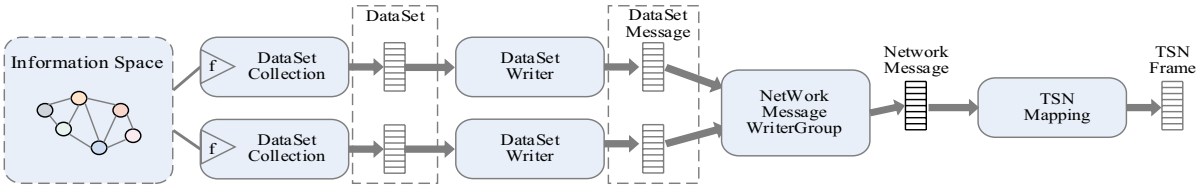

Figure 5. NetworkMessage mapping process.

System model. In this paper, we use application diagrams and structural diagrams to model TT traffic and network topology. An application diagram $G_p(T_p, F_{TT})$ represents the flow graph. The function T_p represents the directed graph's vertices and performs the scheduling calculation task for TT traffic; F_{TT} represents the edges of the directed graph, which is composed of TT traffic. The structure diagram is represented by an undirected graph $G(V, E)$, where V represents a node of an undirected graph composed of a host and a switch and E represents a collection of full-duplex physical links between adjacent nodes.

We map the application diagram to the architecture diagram, and the result is the scheduling process for TT traffic. The process is as follows: first, the scheduled calculation task is assigned to the source node. Then, TT traffic that satisfies the scheduling calculation result is mapped to the full-duplex physical link. Finally, it is transmitted to the destination node according to the routing path. We assume that all hosts and TSN switches have achieved IEEE802.1 AS global time synchronization in the mathematical model.

In the TSN network, the switch, and the host exchange three kinds of traffic: TT traffic, AVB traffic, and BE traffic. To ensure interference-free transmission of TT traffic, we use the scheduling algorithm to generate a Gate Control List. The GCL only contains the static schedule of TT traffic, and the transmission of AVB traffic and BE traffic is performed after the transmission of TT traffic is completed.

The time-sensitive data stream is sent from the source node to the destination node. The time-sensitive data stream is represented by the set s, $s = \{s_1, s_2, s_3, ..., s_n\}$. We use s^{TT} to represent the TT traffic set, with the quadruple s_i to represent TT traffic, where $s_i = \{f_{route}, f_{size}, f_{period}, f_{deadline}\}$. The function f_{route} represents the routing link set of TT traffic, where $f_{route} = \{f_{sender}, ..., f_{receiver}\}$. The TT traffic transmitted in the link contains at least one TSN frame, and f_{period} represents the transmission period of TT traffic; f_{size} represents the load size of each TSN frame multiplied by the number of TSN frames in a transmission period; $f_{deadline}$ represents the maximum allowable end-to-end delay. According to the assumption of[31,32], TT traffic is only kept in TSN within the processing time of the switch.

To improve the algorithm's applications, we consider the impact on the cost function $\cos t(s)$ from three aspects of data flow: scheduling performance, delayed performance, and routing hop performance, which are represented by the functions $c_1(s)$, $c_2(s)$, and $c_3(s)$, respectively. The mathematical model of OPC UA Over TSN data flow routing and scheduling is established, as shown in Eq. (1).

$$\cos t(s) = w_1 \times c_1(s) + w_2 \times c_2(s) + w_3 \times c_3(s) \tag{1}$$

In the formula, w_1,w_2, and w_3 are the weights of the influencing factors, and the sum is as shown in Eq. (2).

$$w_1 + w_2 + w_3 = 1 \tag{2}$$

The first objective function $c_1(s)$ represents the level of ability to schedule NetworkMessage data, as shown in Eq. (3), where s^{TT} represents TT traffic waiting to be scheduled. When a NetworkMessage is schedulable, $f_{s_i} = 0$, otherwise $f_{s_i} = 1$.

$$c_1(s) = \sum_{s_i \in s^{TT}} f_{s_i} \tag{3}$$

There are n data flows in the link, where the scheduled situation is represented by the set of $F_s = \{f_{s_1}, f_{s_2}, ..., f_{s_n} | 1 \le i \le n, s_i \in s^{TT}\}$.

The delay function $c_2(s)$ is the sum function of end-to-end delay. End-to-end delay is shown in Eqs. (4) and (5), where $t_{propagation}$ is the propagation delay, $t_{TimeDelay}$ is the transmission delay, and t_{Queue} is the queuing delay. The variables a_1, a_2, and a_3 represent the weights of the two types of delay on the overall delay, and their sum is 1.

$$t_{e2eDelay} = a_1 \times t_{propagation} + a_2 \times t_{TimeDelay} + a_3 \times t_{Queue} \tag{4}$$

$$a_1 + a_2 + a_3 = 1 \tag{5}$$

The propagation delay calculation formula in the process of NetworkMessage transmission is shown in Eq. (6).

$$t_{propagation} = f_{size}/v_c \tag{6}$$

The v_c represents the speed of electromagnetic wave transmission. To ensure the deterministic transmission of TT traffic, the switch nodes in the Gate Control List are activated simultaneously in the network topology, in a fixed cycle.

TT traffic is routed according to the routing table f_{route} and sent from one node to another. During transmission, the data stream is occupied exclusively by the time slot of the routing link. In the channel, the calculation formula for the transmission delay of TT traffic is shown as Eq. (7), where bw represents the bandwidth of the corresponding link.

$$t_{TimeDelay} = f_{size}/bw \tag{7}$$

In the TSN switch port, we set up a queue for TT traffic scheduling. To ensure the determinism of scheduling, we set a guard band for TT traffic[31,32]. During $t_{TimeDelay}$ and $t_{propagation}$, the link is exclusive.

The queuing delay of TT traffic in the transmission process is shown in Eq. (8), where c is a constant.

$$t_{queue} = \frac{t_{TimeDelay}^{\frac{1}{2(1-c)}}}{\mathrm{f_{size}} \times \left(1 - t_{TimeDelay}^{\frac{1}{2(1-c)}}\right)^{c/(1-c)}} \tag{8}$$

When TT traffic is received by the next-hop node, the scheduling calculation task is executed. According to the propagation delay $t_{propagation}$ and transmission delay $t_{TimeDelay}$ between the links, the overall delay $c_2(s)$ from the source node to the destination node is the function shown in Eq. (9).

$$\mathrm{c_2}(s) = \sum_{r_{ij} \in f_{route}} r_{ij} \times t_{e2eDelay} \tag{9}$$

We assume that the network topology contains N nodes, and the link occupancy situation can be represented by an N matrix f_{route} containing r_{ij}, where the value range of i and j is $(1 \le i, j \le N)$, and TT traffic arrives at node j via node i, $r_{ij} = 1$ otherwise $r_{ij} = 0$, as shown in Eq. (10).

$$f_{route} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1N} \\ r_{21} & r_{22} & r_{23} & \cdots & r_{2N} \\ r_{31} & r_{32} & r_{33} & \cdots & r_{3N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{N1} & r_{N2} & r_{N3} & \cdots & r_{NN} \end{pmatrix} \tag{10}$$

The third objective function $c_3(s)$ represents the routing hop of NetworkMessage data from the source node to the destination node, as shown in Eq. (11), where e_{ij} represent weight.

$$c_3(s) = \sum_{i,j \in N} r_{ij} \times e_{ij} \tag{11}$$

To ensure the determinism of TT traffic transmission, we need to restrict the network accordingly. This paper aims to provide algorithms for minimizing delay of TT traffic routing and scheduling under specified conditions. The core of the routing problem is to reduce the search space of TT traffic and obtain a set of feasible paths. The core of the scheduling problem is to determine the transmission time slot of TT traffic through an algorithm, to optimize the rate of scheduling success.

Condition 1 The sending time of TT traffic should be greater than or equal to 0.

$$t_{StartTime} \ge 0$$

Condition 2 The link should be monopolized during the transmission of TT traffic.

Condition 3 The TSN switch should not buffer TT traffic but only store and forward TT traffic[31,32]. The sending time of TT traffic to the next-hop node should be greater than or equal to the corresponding receiving time.

$$t_{NextStartTime} \ge t_{StartTime} + t_{transmission} + t_{TimeDelay}$$

Condition 4 The transmission of TT traffic should be carried out in the path order specified by the route.

$$t_{StartTime}^{i} < t_{StartTime}^{j} (i < j)$$

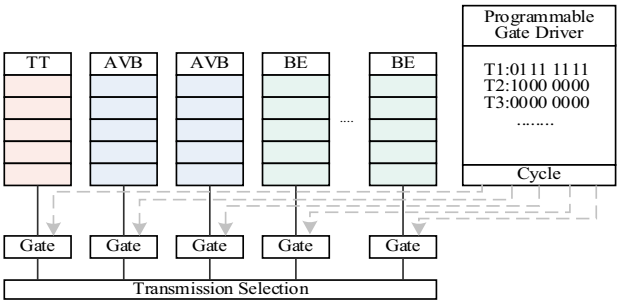

Figure 6. Working principle of IEEE802.1Qbv time-aware-shaper shaper.

Condition 5 The TT traffic that executes the scheduling calculation task must be sent to the next node within the deadline.

$$t_{StartTime} + t_{\text{propagation}} + t_{TimeDelay} \leq f_{deadline}$$

CKSPACO algorithm

Our CKSPACO algorithm contains three sub-algorithms: the cluster analysis algorithm, the K-Shortest-Paths algorithm, and the improved ant colony algorithm.

K-means cluster algorithm. A time-aware shaper (TAS) is defined in *IEEE802.1 Qbv*, as shown in Fig. 6. It contains time-aware queues and gate control lists. The data flow enters different queues according to category, and the Gate Control List handles the opening and closing of the gate.

In the configuration of the TAS, we set up a single transmission queue for TT traffic. Considering that the network data set contains multiple traffic types (TT, AVB, BE) and that our algorithm only considers the routing and scheduling of TT traffic, the data set needs to be classified. We obtained the TT traffic data source through a clustering algorithm. According to the Gate Control List, we can schedule the data stream periodically to ensure interference-free transmission and provide deterministic end-to-end delay.

The frame format used in the encoding of NetworkMessage information at the data link layer is IEEE802.1Q[33]. The TCI of the tag contains a 12-bit VLAN Identifier (VID) to distinguish frames of distinct traffic. Three-bit priority is used to indicate the QoS priority of the frame and determine the type of traffic. We map the three types of traffic to different priorities. TT traffic is indicated by 111, 110, and 101 indicate AVB traffic, and 100, 011, 010, 001, and 000 indicate BE traffic.

In the process of information processing, unsupervised learning methods such as anomaly detection[34], dimensionality reduction[35], and clustering[36] are often used. Considering that TT traffic needs to be classified in three categories of data streams, we adopt a clustering method. In the K-Means clustering algorithm, we classify the data stream by identifying the priority of the frame format in the OPC UA PubSub NetworkMessage.

The core of the clustering algorithm involves feature extraction, similarity calculation, and grouping[37]. We randomly select three initial cluster centers c_i $(1 \leq i \leq 3)$ from the network data and determine the distance between each time-sensitive data stream in the data set and the Euclidean distance to the center of the cluster[38,39]. Then we target the nearest cluster center c_i of the data object, and assign the time-sensitive stream of the corresponding category to it, calculating the average value of the data in each cluster as the new cluster center proceeds to the next iteration, until the cluster center no longer changes or reaches the maximum number of iterations[40,41]. Finally, three different types of time-sensitive data streams are produced.

Through the clustering algorithm, we can determine the quantity of time-triggered traffic from complex NetworkMessages and use it as the data source in the subsequent routing and scheduling algorithms. Effectively reduce the time complexity of the overall algorithm. The pseudo-code of Procedure One is as follows.

Procedure 1 K-Means Clustering Algorithm

Input: number of clusters K, set of data streams S, data stream dimension D
Output: Classified data stream set s^{TT}, s^{AVB}, and s^{BE}
1: Initialize the center point
2: ***for*** i←1, D ***do***
3: 3bit binary priority converted to decimal
4: ***end***
5: ***while*** 1 ***do***
6: ***for*** i←1, S ***do***
7: ***for*** j←1, K ***do***
8: Calculate the Euclidean distance to center point
9: ***end***
10: Give data classification labels
11: ***for*** j←1, K ***do***
12: Calculate the new center point
13: ***If*** center point is different
14: Update center point
15: ***end***
16: ***end***
17: ***end***

K-shortest paths algorithm. Architecture diagram $G\ (V, E)$ represents the TSN network topology, where V is a collection of N nodes, including switch R and host H, $V = R \cup H$, E is a collection of m links, $E = \{e_1, ..., e_m | m \geq 1\}$, and the weights from node i to node j in the set can be represented by e_{ij}. The weights between adjacent nodes in the network topology can be represented by matrix E, as shown in Eq. (12).

$$E = \begin{pmatrix} e_{11} & e_{12} & e_{13} & \ldots & e_{1N} \\ e_{21} & e_{22} & e_{23} & \ldots & e_{2N} \\ e_{31} & e_{32} & e_{33} & \ldots & e_{3N} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ e_{N1} & e_{N2} & e_{N3} & \ldots & e_{NN} \end{pmatrix} \tag{12}$$

In the process of TT traffic transmission, routing is the receiving and forwarding aspect. When TT traffic is routed through a TSN switch, we try to limit the number of TT traffic routing hops to less than eight hops. Considering that the data stream path finding process increases in complexity with the growth of the network topology, we use the KSP algorithm to optimize network routing[42]. This algorithm can determine the first k shortest paths by weight from the source node to the destination node, reduce the search space of TT traffic by generating the shortest path group, and speed up scheduling algorithm search efficiency[43]. The KSP algorithm consists of two parts, recursion; and determining the shortest path. Finally, the result of the algorithm is used as a new routing table to process the scheduling of TT traffic.

According to the recursion concept, we use the deviation path algorithm to find the shortest path p_k from the source node to the destination node, where the set of paths $p_k = \{f_{sender}, ..., f_{receiver}\}$. Then we delete the source node and the destination node in the path set p_k, label the remaining nodes in the set as deviating nodes in the routing order and select the new shortest path in the path set determined for the new nodes. The first k shortest paths from the source node to the destination node are derived by iteration $p = \{p_1, ..., p_k | k \geq 1\}$. The f_{route} represents the best routing path selected for TT traffic in the path set p.

The pseudo-code of procedure two is as follows.

Procedure 2 K-Shortest-Paths Algorithm

Input: Network topology G, Source node f_{sender}, Destination node $f_{receiver}$
Output: K-shortest paths set P
1: Initialize parameter k as the number of paths
2: Calculate the shortest path p_l
3: $P \leftarrow \{ p_l \}$
4: ***for*** $l \leftarrow 1$, l_{max} ***do***
5: ***If*** $l \neq 1$ **then**
6: ***If*** $c(p_l) \geq c(p_{l-1})$ ***then***
7: ***return*** $P=\{p_1, p_2, \ldots p_{l-1}\}$
8: ***end***
9: ***end***
10: Generate offset path
11: Calculate cost c
12: Calculate the new shortest path p_{l+1}
13: $P \leftarrow p_{l+1}$
14: ***end***

Improved ant colony algorithm. To ensure that the data stream schedule in the link can be transmitted to the destination node with minimum delay, we use an improved ant colony optimization algorithm to find the minimum delay of the data stream during routing and determine the optimal scheduling plan[44,45].

To ensure efficiency in the convergence rate of the ant colony algorithm, it is necessary to prevent the ant colony from falling into a local optimum due to a fast convergence rate[46–49]. We introduce the concept of information entropy to express the pros and cons of each path determined by the ant colony. By calculating the information entropy, we make adjustment the parameters to the pheromone heuristic factor α and the expected heuristic factor β. In the optimal ant colony path, when the ant colony selects the next-hop path, if each point's probability is the same, then the information entropy is at the maximum value. The formula is shown in Eq. (13).

$$H(x) = -\sum_{t=1}^{r} p(x_t) \log p(x_t) \tag{13}$$

The set X of next-hop path alternatives is represented as $X = \{x_1, x_2, \ldots, x_r\}$, and P is the probability of corresponding set elements. To normalize the information entropy, we need to find the maximum quantity of information entropy, which is when the probability of the next hop of the path set elements is equal, as shown in Eq. (14).

$$H_{\max} = -\sum_{i=1}^{r} P_i \log P_i = -\sum_{i=1}^{r} \frac{1}{r} \log \frac{1}{r} \tag{14}$$

The maximum value of information entropy is $H_{\max}$, and r represents the number of elements in the next-hop selection set. Normalization processing is shown in Eq. (15).

$$H^{'} = H(x)/H_{\max} \tag{15}$$

The value of ρ influences the convergence rate of the ant colony algorithm and the global searchability.

Accordingly, we propose a formula for adjusting ρ in real time based on a negative feedback mechanism, which reduces premature convergence of the positive feedback mechanism that depends on pheromone concentration in the ant colony algorithm, as shown in Eq. (16).

$$\rho = [(\rho_{\max} - \rho_{\min}) \times t + (T - 1) \times \rho_{\min} + \rho_{\max}]/T \tag{16}$$

The maximum evaporation coefficient is $\rho_{\max}$, $\rho_{\min}$ is the minimum evaporation coefficient, t is the current iteration number, and T is the maximum iteration number. The pseudo-code of Procedure Three is as follows.

Procedure 3 Improved Ant Colony Algorithm

Input: K-Shortest-Path set P, TT traffic set s^{TT}
Output: global fitness *cost(s)*
1. Initialize the parameters α , β , ρ , Q of ACO
2.The master sends the parameters to the slaves
3. ***While*** < MaxIterations ***do***
4. Put m ants on the source node
5. ***for*** $i \leftarrow 1$, m ***do***
6. ***for*** $j \leftarrow 1$, (n-1) ***do***
7. Find the neighboring node
8. Remove visited nodes from neighboring nodes
9. Roulette wheel selection next hop node
10. ***If*** arrive at the destination node ***then***
11. Record the number of ants hop
12. ***end***
13. ***end***
14. ***end***
15. Calculate the global fitness function
16. Dynamic parameter adjustment
17. ***end***

Simulation result

To prove the superiority of the algorithm, this paper compares the CKSPACO and the IACO[12] algorithms. The program is written using Matlab, and the hardware specifications are Intel Core i7-8750H, 8 GB RAM. We consider two factors that affect the transmission efficiency of time-triggered traffic: the quantity of time-triggered traffic, and the network topology scale. According to these factors, the improved algorithm was tested in various scenarios. In our mathematical model, the weight parameters are set to the following values: $w_1 = 0.6$, $w_2 = 0.2$, $w_3 = 0.2$, $a_1 = 0.2$, $a_2 = 0.3$, $c = 0.5$.

We selected 100, 250, and 500 data streams from the network data set as the input of the K-Means clustering algorithm. When the input data stream is 100, the clustering result contains 38 TT traffic, 31 AVB traffic, and 31 BE traffic. The result is shown in Fig. 7.

The other two data stream distributions are shown in Table 1.

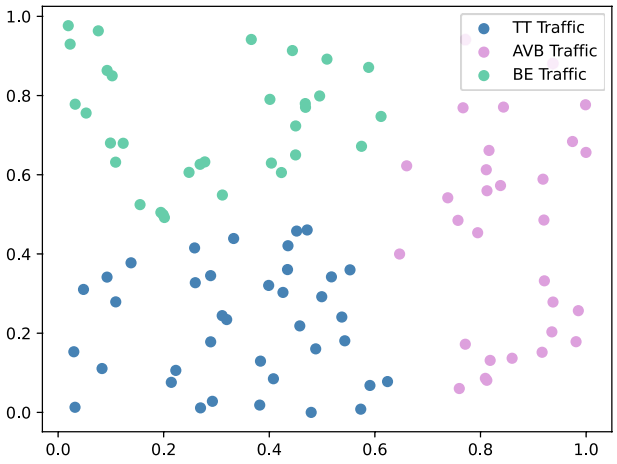

Figure 7. The number of data streams is 100.

Algorithm category	The number of TT traffic		
	38 TT traffic	123 TT traffic	208 TT traffic
KSPACO	1.7	4.8	11.7
IACO	3.0	11.2	23.4

Table 1. Data stream category result.

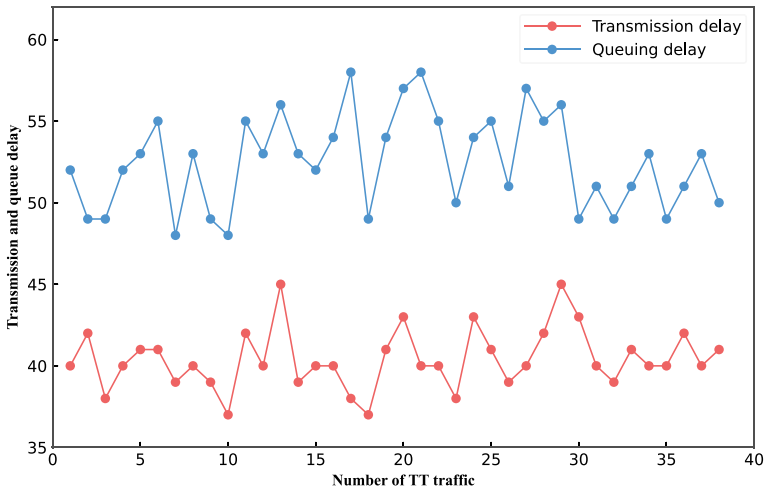

Figure 8. 38 TT traffic transmission delay and queuing delay.

In the KSP algorithm, we set up three scales of network topology scales of 10, 20, and 30. Take the network topology of 20 nodes as an example, and calculate the first k shortest paths from the source node 1 to the destination node 20, expressed in a matrix. Generate a new routing table to reduce the search range of the improved ant colony algorithm.

$$\begin{pmatrix} 1 & 8 & 15 & 14 & 20 & 0 \\ 1 & 3 & 17 & 14 & 20 & 0 \\ 1 & 2 & 4 & 19 & 20 & 0 \\ 1 & 8 & 9 & 12 & 14 & 20 \\ 1 & 3 & 6 & 17 & 14 & 20 \end{pmatrix}$$

Finally, we use the quantity of TT traffic obtained by the K-Means clustering algorithm as the data source for routing and scheduling in the improved ant colony algorithm. The KSP algorithm utilizes the first k shortest paths for the routing table of the improved ant colony algorithm. Using the ant colony algorithm, we obtain the single-hop path, the delay of the global path, and improve TT traffic scheduling performance.

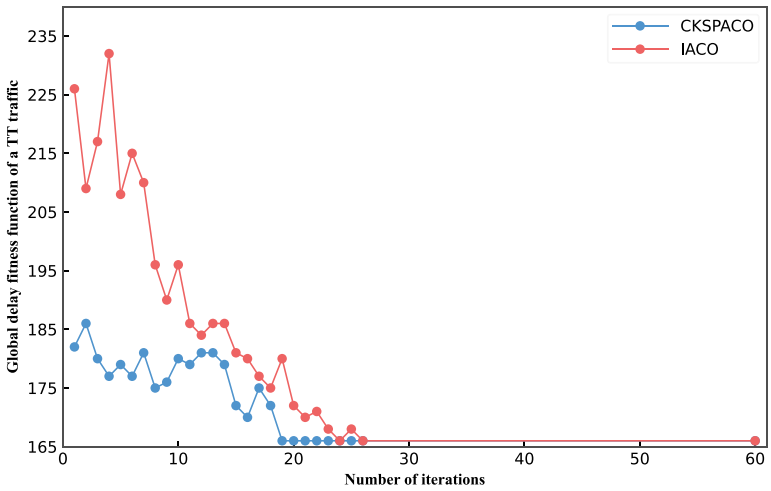

Figure 9. One TT traffic global delay.

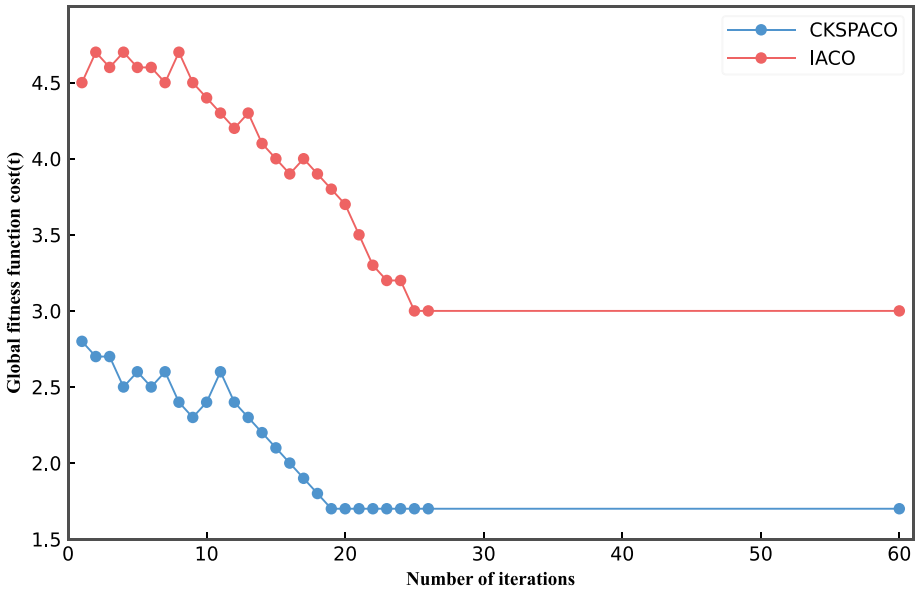

Figure 10. Comparison of overall cost fitness function between CKSPACO algorithm and IACO algorithm.

Figure 8 shows the transmission delay and queuing delay generated when 38 TT traffic passes through nodes in the network topology. We can know that One TT traffic transmission delay and queuing delay are stable at 40 μs and 50 μs.

Figure 9 shows the CKSPACO and IACO algorithms' global delay function in a TT traffic routing and scheduling process. From the figure, we see that the random search strategy of the IACO algorithm greatly affects the global delay of the algorithm in the beginning. Still, it also converges to the corresponding value as the number of iterations increases. Compared with the IACO algorithm, the CKSPACO algorithm performs better in the convergence rate and the number of iterations.

Figure 10 shows performance in terms of the global fitness function cost of the CKSPACO algorithm and the IACO algorithm. We test the algorithms based on 38 TT traffic with 20 network topology nodes. Because the IACO algorithm schedules TT traffic based on a fixed routing strategy, there is a large queuing delay in the process, which increases overall cost. Compared with the IACO algorithm, the CKSPACO algorithm provides a dynamic routing strategy for TT traffic that ensures a good convergence rate and number of iterations of the algorithm. On the whole, the CKSPACO algorithm can provide better scheduling performance.

We test algorithm performance by adjusting the quantity of TT traffic, the scale of the network topology, and network bandwidth. The quantity of TT traffic in different scenarios is obtained by the clustering algorithm, represented by the set C. $C=\{38,123,208\}$. Set R represents different network topology scales, $R=\{10,20,30\}$; the combined influence of the two factors is used to test the algorithm.

The number of data stream	Data stream category		
	TT traffic	AVB traffic	BE traffic
250	123	84	43
500	208	178	114

Table 2. Data the influence of the quantity of TT traffic on the overall fitness function.

Algorithm category	Network topology scale		
	10 topology	20 topology	30 topology
KSPACO	2.1	1.7	1.6
IACO	3.1	3.0	2.8

Table 3. The influence of network topology scale on the overall fitness function.

When the network topology scale is 20 nodes, the algorithm performance is tested by adjusting the quantity of TT traffic to be dispatched, as shown in Table 2.

As the quantity of TT traffic increases, the fitness functions of the two algorithms gradually become larger. Since the IACO algorithm utilizes a fixed routing strategy, its fitness function value increases significantly as the quantity of TT traffic increases. Although the value of the CKSPACO algorithm fitness function also increases with increase in the quantity of TT traffic, it still maintains good scheduling performance.

When the quantity of TT traffic to be scheduled is 38, algorithm performance is tested by adjusting the network topology scale, as shown in Table 3.

As the scale of the network topology expands, the values of the fitness functions of the two algorithms fluctuate less, because we limit the number of routing hops to less than 8 hops in the KSP algorithm, and the scale of TT traffic to be scheduled is small. The IACO algorithm based on fixed routing is virtually unaffected by changes in the network topology. In the CKSPACO algorithm, the fitness function value based on 10 network nodes fluctuates slightly, due to the connectivity of our initially generated network topology.

Conclusion

In OPC UA over TSN, we have solved the routing and scheduling problems of time sensitive data streaming in OPC UA PubSub NetworkMessage with the improved ant colony algorithm CKSPACO, of great significance to the combination of OPC UA with TSN. Using the improved algorithm to schedule TT traffic can effectively increase algorithm convergence rate and improve the real-time performance of network message transmission.

Received: 17 April 2021; Accepted: 28 July 2021
Published online: 15 September 2021

References

1. Li, J. Q. *et al.* Industrial internet: A survey on the enabling technologies, applications, and challenges. *IEEE Commun. Surv. Tutor.* **19**(3), 1504–1526 (2017).
2. Lasi, H., Fettke, P., Kemper, H. G., Feld, T. & Hoffmann, M. Industry 4.0. *Bus. Inf. Syst. Eng.* **6**(4), 239–242 (2014).
3. OPC Foundation, "OPC Unified Architecture Specification", Parts 1–11. www.opcfoundation.org2009.
4. Leitner, S. H. & Mahnke, W. OPC UA-service-oriented architecture for industrial applications. *ABB Corp. Res. Center* **48**, 61–66 (2006).
5. Hannelius, T., Salmenpera, M., & Kuikka, S. Roadmap to adopting OPC UA. In *2008 6th IEEE International Conference on Industrial Informatics* (pp. 756–761). IEEE (2008).
6. Pauker, F., Frühwirth, T., Kittl, B. & Kastner, W. A systematic approach to OPC UA information model design. *Procardia CIRP* **57**, 321–326 (2016).
7. Garcia, M. V., Irisarri, E., Pérez, F., Estévez, E., & Marcos, M. OPC-UA communications integration using a CPPS architecture. In *2016 IEEE Ecuador Technical Chapters Meeting (ETCM)* (pp. 1–6). IEEE (2016).
8. Finn, N. Introduction to time-sensitive networking. *IEEE Commun. Stand. Mag.* **2**(2), 22–28 (2018).
9. Farkas, J., Bello, L. L. & Gunther, C. Time-sensitive networking standards. *IEEE Commun. Stand. Mag.* **2**(2), 20–21 (2018).
10. Messenger, J. L. Time-sensitive networking: An introduction. *IEEE Commun. Stand. Mag.* **2**(2), 29–33 (2018).
11. IEEE 802. *1AS-2011. Standard for Local and Metropolitan Area Networks-Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks* (IEEE Press, 2011).
12. IEEE 802. *1Qat-2010. Standard for Local and Metropolitan Area Networks-Virtual Bridged Local Area Networks: Stream Reservation Protocol (SRP)* (IEEE Press, 2010).
13. IEEE 802. *1Qav-2009. Standard for Local and Metropolitan Area Networks-Virtual Bridged Local Area Networks: Forwarding and Queuing Enhancements for Time-Sensitive Streams* (IEEE Press, 2009).
14. IEEE 802. *1Qbu-2016. Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks: Frame Preemption* (IEEE Press, 2016).
15. IEEE 802. *1Qbv-2015. Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks: Enhancements for Scheduled Traffic* (IEEE Press, 2016).
16. IEEE 8021Qcc-2018. *Standard for Local and metropolitan area networks-Bridges and Bridged Networks: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements* (IEEE Press, 2018).

17. Li, Y., Jiang, J., Lee, C. & Hong, S. H. Practical implementation of an OPC UA TSN communication architecture for a manufacturing system. *IEEE Access* **8**, 200100–200111 (2020).
18. He, Q. *et al.* TIM: A two-stage iterative framework for influence maximization in social networks. *Appl. Math. Comput.* **354**, 338–352 (2019).
19. Gogolev, A., Mendoza, F., & Braun, R. TSN-enabled OPC UA in field devices. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)* (Vol. 1, pp. 297–303). IEEE (2018).
20. Eckhardt, A., & Müller, S. Analysis of the round trip time of opc ua and tsn based peer-to-peer communication. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (pp. 161–167). IEEE (2019).
21. Bruckner, D. *et al.* An introduction to OPC UA TSN for industrial communication systems. *Proc. IEEE* **107**(6), 1121–1131 (2019).
22. Tian, S., & Hu, Y. The role of opc ua tsn in it and ot convergence. In *2019 Chinese Automation Congress (CAC)* (pp. 2272–2276). IEEE (2019).
23. Kobzan, T., Blöcher, I., Hendel, M., Althoff, S., Gerhard, A., Schriegel, S., & Jasperneite, J. Configuration Solution for TSN-based Industrial Networks utilizing SDN and OPC UA. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (Vol. 1, pp. 1629–1636). IEEE (2020).
24. Pahlevan, M., & Obermaisser, R. Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)* (Vol. 1, pp. 337–344). IEEE (2018).
25. Laursen, S. M., Pop, P. & Steiner, W. Routing optimization of avb streams in tsn networks. *ACM SIGBED Rev.* **13**(4), 43–48 (2016).
26. Tma-Selicean, D., Pop, P. & Steiner, W. Design optimization of tt-ethernet-based distributed real-time systems. *Real-Time Syst.* **51**, 1 (2014).
27. Bingqian, L., & Yong, W. Hybrid-GA based static schedule generation for time-triggered ethernet. In *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. IEEE (2016).
28. Wang, Y., Chen, J., Ning, W., Yu, H., & Chen, C. A time-sensitive network scheduling algorithm based on improved ant colony optimization (2020).
29. OPC. *10000–14-UA Specification Part 14 PubSub* (OPC Foundation, 2018).
30. Huang, Q., Sun, K., Li, X., & Wu, D. O. Just FUN: A joint fountain coding and network coding approach to loss-tolerant information spreading. In *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing* (pp. 83–92) (2014).
31. Pahlevan, M., & Obermaisser, R. Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)* (Vol. 1, pp. 337–344). IEEE (2018).
32. Pahlevan, M., Tabassam, N. & Obermaisser, R. Heuristic list scheduler for time triggered traffic in time sensitive networks. *ACM Sigbed Rev.* **16**(1), 15–20 (2019).
33. IEEE 802.1Q-2014. *Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks: Per-Stream Filtering and Policing* (IEEE Press, 2014).
34. Sotiris, V. A., Peter, W. T. & Pecht, M. G. Anomaly detection through a bayesian support vector machine. *IEEE Trans. Reliab.* **59**(2), 277–286 (2010).
35. Raymer, M. L., Punch, W. F., Goodman, E. D., Kuhn, L. A. & Jain, A. K. Dimensionality reduction using genetic algorithms. *IEEE Trans. Evol. Comput.* **4**(2), 164–171 (2000).
36. Kanungo, T. *et al.* An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(7), 881–892 (2002).
37. Tan, L., Li, C., Xia, J. & Cao, J. Application of self-organizing feature map neural network based on K-means clustering in network intrusion detection. *Comput. Mater. Contin.* **61**(1), 275–288 (2019).
38. Lianbo, M., Shi, C. & Yuhui, S. Enhancing learning efficiency of brain storm optimization via orthogonal learning design. *IEEE Trans. Syst. Man Cybern. Syst.* https://doi.org/10.1109/TSMC.2020.2963943 (2020).
39. Cheng, S., Ma, L., Hui, Lu., Lei, X. & Shi, Y. Evolutionary computation for solving search-based data analytics problems. *Artif. Intell. Rev.* https://doi.org/10.1007/s10462-020-09882-x (2020).
40. Wang, K., Qi, X., Liu, H. & Song, J. Deep belief network based k-means cluster approach for short-term wind power forecasting. *Energy* **165**, 840–852 (2018).
41. Steinley, D. & Brusco, M. J. A new variable weighting and selection procedure for K-means cluster analysis. *Multivar. Behav. Res.* **43**(1), 77–108 (2008).
42. Sedeño-Noda, A. An efficient time and space k point-to-point shortest simple paths algorithm. *Appl. Math. Comput.* (2012).
43. Roditty, L. On the k shortest simple paths problem in weighted directed graphs. *SIAM J. Comput.* **39**(6), 2363–2376 (2010).
44. Dorigo, M., Maniezzo, V. & Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **26**(1), 29–41 (1996).
45. Ma, L. *et al.* A novel many-objective evolutionary algorithm based on transfer learning with kriging model. *Inf. Sci.* **509**, 437–456 (2020).
46. Mahi, M., Baykan, Ö. K. & Kodaz, H. A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. *Appl. Soft Comput.* **30**, 484–490 (2015).
47. Jun-man, K. & Yi, Z. Application of an improved ant colony optimization on generalized traveling salesman problem. *Energy Proced.* **17**, 319–325 (2012).
48. Yang, J., Shi, X., Marchese, M. & Liang, Y. An ant colony optimization method for generalized TSP problem. *Prog. Natl. Sci.* **18**(11), 1417–1422 (2008).
49. Hlaing, Z. C. S. S., & Khine, M. A. An ant colony optimization algorithm for solving traveling salesman problem. In *Fifth Local Conference on Parallel and Soft Computing* (2010).
50. Cavalieri, S. & Chiacchio, F. Analysis of OPC UA performances. *Comput. Stand. Interfaces* **36**(1), 165–177 (2013).

Acknowledgements

This work was supported by the National Key Research and Development Program of China (2018YFB1700103).

Author contributions

Y.S. was responsible for the overall structure and review of the paper, C.G. wrote the main manuscript text, P.X., L.L. and R.Z. were responsible for post proofreading.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Y.S.