Collision-free Scheduling of Multi-bridge Machining Systems: A Colored Traveling Salesman Problem-based Approach

Jun Li, *Senior Member, IEEE,* Xianghu Meng, and Xing Dai

***Abstract*—Multi-bridge machining systems (MBMS) have gained wide applications in industry due to their high production capacity and efficiency. They contain multiple bridge machines working in parallel within their partially overlapping workspaces. Their scheduling problems can be abstracted into a serial-colored travelling salesman problem in which each salesman has some exclusive cities and some cities shared with its neighbor(s). To solve it, we develop a greedy algorithm that selects a neighboring city satisfying proximity. The algorithm allows a salesman to select randomly its shared cities and runs accordingly many times. It can thus be used to solve job scheduling problems for MBMS. Subsequently, a collision-free scheduling method is proposed to address both job scheduling and collision resolution issues of MBMS. It is an extension of the greedy algorithm by introducing time window constraints and a collision resolution mechanism. Thus, the augmented greedy algorithm can try its best to select stepwise a job for an individual machine such that no time overlaps exist between it and the job sequence of the neighboring machine dealt in the corresponding overlapping workspace; and remove such a time overlap only when it is inevitable. Finally, we conduct a case study of a large triple-bridge waterjet cutting system by applying the proposed method.**

***Index Terms*—Collision resolution, greedy algorithm, modeling, multiple traveling salesman problem, scheduling.**

I. Introduction

MULTI-BRIDGE machining systems (MBMS) as important processing equipment for production have been paid great attention by both academia and industry. Such a system contains at least two concurrent individual bridge machines and an overlapping workspace shared by them. Each machine is required not only to perform the tasks in its own section of workspace but also to complete together with others the tasks in the overlapping section shared by them. Meanwhile, they are subject to some process and geometric constraints, e.g., collision avoidance. Two MBMS are cited as typical examples, i.e., a dual-bridge waterjet machining center and a dual-manipulator hull welding system, as illustrated in Fig. 1. Both consist of two independent individual machines, i.e., cutting machines and manipulators. The overlap between two individual workspaces is required in order to guarantee no dead processing zone. Either but only one of two neighboring machines is allowed to perform any shared jobs in their overlapping zone.

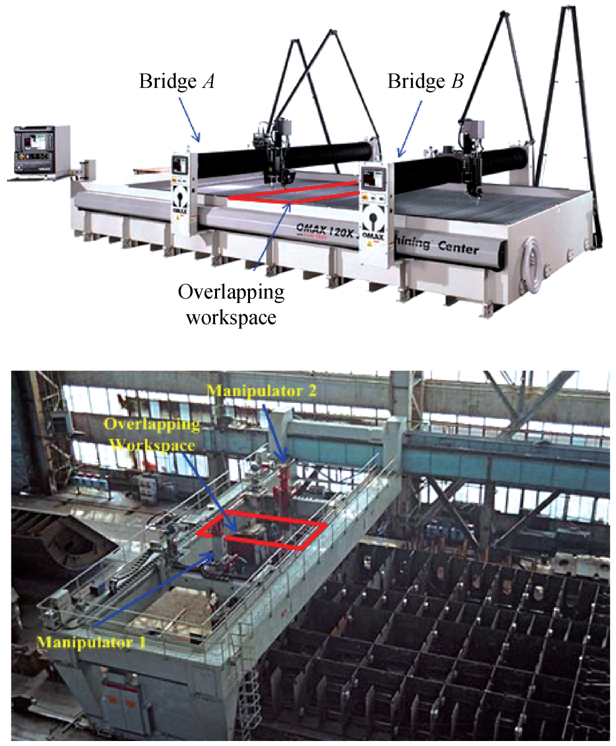


Fig. 1. Two MBMS. (a) Dual-bridge waterjet cutting machine tool, and (b) Gantry dual-manipulator hull welding system.

Such systems face three common problems at the operational level, i.e., job partition/assignment, job scheduling and collision resolution or machine coordination. A job scheduling problem is to determine the best job sequence for each individual machine after job partition and assignment. Collision resolution is to implement orderly execution of jobs assigned to multiple machines while avoiding their collision. Solving them is crucial for promoting the production efficiency and enhancing the reliability of the systems.

Manuscript received February 19, 2016; accepted August 31, 2016. This work was supported in part by the National Natural Science Foundation of China (61773115, 61374069, 61374148) and the Natural Science Foundation of Jiangsu Province (BK20161427). Recommended by Associate Editor Naiqi Wu. *(Corresponding auther: Jun Li.)*

Citation: J. Li, X. H. Meng, and X. Dai, "Collision-free scheduling of multi-bridge machining systems: a colored traveling salesman problem-based approach," *IEEE/CAA J. of Autom. Sinica*, vol. 5, no. 1, pp. 139−147, Jan. 2018.

J. Li, X. H. Meng, and X. Dai are with the Ministry of Education Key Laboratory of Measurement and Control, School of Computer Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: j.li@seu.edu.cn; xdmengxianghu@163.com; 1508178155@qq.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2017.7510415

Our prior work [1], [2 has presented a genetic algorithm for collision-free path planning of three-bridge waterjet cutting. Meng *et al.*, [3] examine a cutting task sequencing method for dual-bridge waterjet cutting with collision avoidance via population-based incremental learning (PBIL). All achieve a collision-free schedule by searching first and then increasing time intervals among jobs to resolve potential collisions between two neighboring bridges once the search fails. Our prior work [4]−[7] proposes a colored traveling salesman problem (CTSP) in which each salesman has an exclusive city set of the same color and all salesmen share only one common city set of multiple colors. Here, we rename it a radial-CTSP (R-CTSP) to differentiate it from the one to be proposed. It has been applied to the path planning of a dual-bridge waterjet cutting machine tool [8]. It is applicable to multiple machine systems containing radially arranged individual machines whose workspaces have a single common section. The scheduling of an MBMS with two machines can be formulated as a specific case of R-CTSP with two salesmen. However, R-CTSP cannot be applied to MBMS with multiple machines sharing multiple common sections.

This work proposes a serial-CTSP (S-CTSP) to abstract the scheduling problems of MBMS. Based on it and its solution algorithm, we develop an integrated method for resolving both job scheduling and coordination problems of MBMS. The main contributions are to:

1) Propose an S-CTSP in which each salesman has some exclusive cities and shares some cities with its neighbor(s). It differs from R-CTSP but they are identical when both have only two salesmen. A greedy algorithm is developed for solving S-CTSP. It assigns a shared city in a random manner during stepwise neighboring city search satisfying proximity.

2) Develop a collision-free scheduling method to perform both job scheduling and collision resolution of MBMS. It is an augmented greedy algorithm that introduces some timing constraints for a collision-free job selection together with a collision resolution mechanism which is only used when such a selection fails.

3) Apply S-CTSP and the proposed methods to a triple-bridge waterjet cutting process of a large world map. The cutting job assignment, basic path scheduling, and collision-free path scheduling are demonstrated.

The rest of the paper is organized as follows. Section II reviews the related work. S-CTSP and its algorithm are proposed in Section III. Section IV gives the collision-free scheduling method. A case study of a three-bridge waterjet cutting process is presented in Section V. Section VI concludes this paper.

II. Related Work

In some multiple machine systems, machines are required to perform all jobs and a job can be executed exactly once by a machine. After all jobs are accomplished all machines return to their depots. It seems that the job scheduling problem of such a system can be attributed to a multiple traveling salesman problem (MTSP) [9] or transformed into a traveling salesman problem (TSP) [10]. Namely, jobs, machines, and the objective of job scheduling can be represented as cities, salesmen, and an objective function of MTSP or TSP, respectively. For example, the motion of multiple robots is planned to cover a point set as multiple single TSP [11]; the job scheduling of a dual-manipulator manufacturing cell is modelled as an MTSP [12]; and the test ordering problem of a new equipment with four mobile probes for testing printed circuit boards is formulated as a TSP [13]. The authors have made some changes to the original problems for convenience of solutions. Either of the similarity or difference of tasks is thus neglected. Note that some tasks in MBMS are exclusive for specified machines and some allow any one of neighboring machines to perform. Both MTSP and multiple individual TSPs cannot represent simultaneously such different and identical groups of tasks since their cities are either identical or different from each other in term of their accessibility by salesmen. Formulating the scheduling problems as MTSP or TSP may change their original solution space. Our prior work [4]−[7] presents an R-CTSP by differently coloring cities to catch the features of the scheduling problems. The differences among R-CTSP, MTSP and multiple single TSPs are disclosed in [4], [7]. It confirms that CTSP has a much larger solution space than the combination of multiple individual TSPs but smaller than MTSP with respect to the same problem size and coding scheme. Then, we apply R-CTSP to the path planning of a dual-bridge waterjet cutting machine tool [8].

There are many approaches to both the scheduling and coordination problems of multimachine systems not limited to MBMS. Motion planning and collaboration of robots are often addressed together in multi-robot collaboration systems [14], [15]. A scheduling method for deadlock avoidance of a robotized manufacturing cell is reported in [16]. As mentioned, Chakraborty *et al.* [11] plan collision-free motions of multiple robots in an electronics manufacturing system to cover a point set as TSPs subject to geometric constraints, and Xidias *et al.* [12] develop a genetic algorithm (GA) coding with an inverse kinematics solution for collision-free scheduling of a dual-manipulator manufacturing cell. Gonzalez-Rodriguez *et al.* [13] examine new equipment with four mobile probes for testing printed circuit boards. They solve the test ordering problem for multiple probes as a TSP first and then implement their motion coordination. Other approaches to task allocation and interaction of multiple robots are reported by Dahl *et al.* [17], Korsah *et al.* [18], and Zheng *et al.* [19]. Our prior works propose a motion control method for collaborative welding of multiple manipulators under the circumstance without fixtures [20], and a collision-free path planning method based on a genetic algorithm for three-bridge waterjet cutting [1], [2] and one based on PBIL for dual-bridge waterjet cutting [3]. Li *et al.* [2] and Du [1] insert waiting time to the obtained schedule to avoid two neighboring bridges co-locating in the same overlapping processing zone to resolve their collision. Meng *et al.* [3] give a method to derive a collision-free schedule. These methods have to be generalized in order to make their application wider.

III. S-CTSP and Greedy Algorithm

A. S-CTSP Formulation

Our prior work [4]−[7] propose a basic CTSP, radial-CTSP. This paper focuses on a new class of CTSP, i.e., serialCTSP with m salesmen and n cities, where $m < n$ and $n, m \in Z = \{1, 2, 3, \ldots\}$. It is defined over a complete digraph $G = (\aleph, E)$, where vertex set $\aleph = \{1, 2, \ldots, n\}$ numbers the cities; and each edge $(i, j) \in E$, $i \neq j$, is associated with a weight ω_{ij} representing a visit cost (e.g., distance) between two cities i and j. $\aleph$ is divided into $2m - 1$ disjoint non-empty sets, i.e., exclusive city sets V_i, $\forall i \in Z_m = \{1, 2, \ldots, m\}$, and shared city sets U_j, $\forall j \in Z_{m-1}$. A salesman i is assigned with color i. $\forall a \in V_i$, its color i can be only be visited by salesman i. $\forall a \in U_j$, $\rho(a) = \{j, j+1\}$ meaning both salesmen i and j can visit a. For example, salesmen 1−3 visit $V_1 \cup U_1$, $U_1 \cup V_2 \cup U_2$, and $U_2 \cup V_3$, in Fig. 2, respectively. Vertex $d_i \in V_i$ represents the depot where salesman i departs and returns. The objective of S-CTSP is to determine m Hamiltonian cycles over G with the least total tour cost.

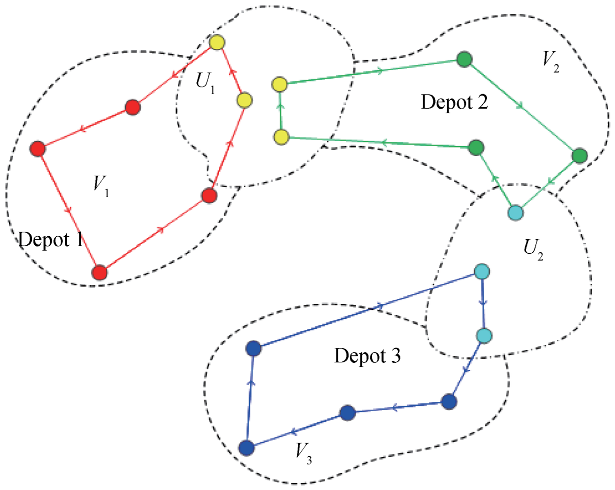


Fig. 2. Example of S-CTSP with 3 salesmen sharing 2 common city sets.

Next, S-CTSP is formulated as a 0-1 integer programming model. Let $U_0 = U_m = V_{m+1} = \emptyset$ for mathematical convenience. The accessible city set of salesman k is $\aleph_k = U_{k-1} \cup V_k \cup U_k$, $\forall k \in Z_m$. Binary access variables $x_{ijk} = 1$, $i \neq j$, $\forall i, j \in \aleph$, if salesman k passes through edge (i, j); otherwise, $x_{ijk} = 0$. u_{ik} is the number of nodes visited on the tour of salesman k from d_k to node i.

$$\min f = \sum_k \sum_i \sum_j \omega_{ij} x_{ijk} \quad \forall k \in Z_m,\ \forall i, j \in \aleph_k. \tag{1}$$

Subject to: Each salesman k is required to start from and return to depot d_k:

$$\sum_i x_{d_k ik} = 1, \quad \text{and} \tag{2}$$

$$\sum_i x_{id_k k} = 1 \quad \forall i \in \aleph_k \backslash \{d_k\},\ \forall k \in Z_m. \tag{3}$$

Salesman k can neither travel to nor from a city outside $\aleph_k$:

$$\sum_i \sum_j x_{ijk} = 0, \quad \text{and} \tag{4}$$

$$\sum_i \sum_j x_{jik} = 0 \quad \forall i \in V_k,\ \forall j \in \aleph \backslash \aleph_k,\ \forall k \in Z_m. \tag{5}$$

Another salesman l is forbidden to travel to or from an exclusive city of salesman k:

$$\sum_i \sum_j x_{ijl} = 0, \quad \text{and} \tag{6}$$

$$\sum_i \sum_j x_{jil} = 0 \quad \forall i \in V_k,\ \forall j \in \aleph,\ \forall l \in Z_m \backslash \{k\}. \tag{7}$$

Salesman k cannot visit a city outside its accessible city set from or back to its shared cities:

$$\sum_i \sum_j x_{ijk} = 0, \quad \text{and} \tag{8}$$

$$\sum_i \sum_j x_{jik} = 0 \quad \forall i \in U_{k\text{-}1} \cup U_k,\ \forall j \in \aleph \backslash \aleph_k,\ \forall k \in Z_m. \tag{9}$$

A salesman other than salesmen k and $k+1$ is forbidden to visit the cities in U_k shared by salesmen k and $k+1$:

$$\sum_i \sum_j x_{ijl} = 0, \quad \text{and} \tag{10}$$

$$\sum_i \sum_j x_{jil} = 0 \quad \forall i \in U_k,\ \forall j \in \aleph,\ \forall l \in Z_m \backslash \{k, k+1\}. \tag{11}$$

Each city can be visited by one salesman exactly once:

$$\sum_i \sum_j x_{ijk} = 1, \quad \text{and} \tag{12}$$

$$\sum_i \sum_j x_{jik} = 1, \quad j \neq i \quad \forall i, j \in \aleph_k,\ \forall k \in Z_m. \tag{13}$$

A shared city cannot be disconnected from a tour once it is visited by a salesman:

$$\sum_i x_{jik} = \sum_i x_{ijk} \quad i \neq j \neq h \quad \forall j \in U_k,\ \forall i \in V_k \cup U_k \cup V_{k+1}. \tag{14}$$

Any solution consisting of several disconnected sub-tours for a salesman must be forbidden, as indicated by the following equation incorporated with (14):

$$u_{ik} - u_{jk} + n_k \times x_{ijk} \leq n_k - 1 \quad j \neq i \quad \forall i, j \in \aleph_k \backslash \{d_k\},\ \forall k \in Z_m. \tag{15}$$

S-CTSP is new and was never formulated before to the best knowledge of the authors. S-CTSP and R-CTSP become identical when they have only two salesmen. R-CTSP is proven to be NP-hard [6]. So is S-CTSP.

B. Greedy Algorithm for S-CTSP

We cannot solve exactly S-CTSP with city size 40 via LINGO within two days. A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum [21]. In many problems, a greedy strategy may not produce a global optimal solution but a local one that approximates the global one in a reasonable time. This algorithm has been applied to TSP and can quickly yield an effectively short route [22]. We develop it to solve S-CTSP. To grasp the characteristics of S-CTSP, we allow the algorithm to assign a shared city to a salesman in a random manner during stepwise neighboring city search satisfying proximity. It contributes to the greater solution space and more chances to achieve the best solution than the original greedy algorithm does. Note that the assignment of shared cities is fixed and thus only one solution exists in the original greedy algorithm.

Suppose that the solution of an S-CTSP is denoted as $X = (X_1, X_2, \ldots, X_m)$, where X_i represents the route of salesman i, $\forall i \in Z_m$. The depot of a salesman is only regarded as the head of the corresponding route sequence and is taken into account as the home to close the route so as to satisfy (2) and (3). All the following symbols are the same as those mentioned in Section III. We let $q_k = |U_k|$, $r_k = |V_k|$, and $n_k = |\aleph_k|$, $\forall k \in Z_m$.

Each salesman in an S-CTSP needs to not only traverse all its exclusive cities but also visit randomly some shared cities. The city visit needs to meet proximity stepwise. Proximity represents a criterion of the least visit cost not limited to the shortest visit distance. A salesman at the current city selects the next unvisited one that is closest to it. It is different for a salesman to select an exclusive city and a shared city. Particularly, given candidate city a unvisited from accessible city set $\aleph_k$, if $a \notin U_k$, it must belong to V_k and should be added to the route of salesman k; otherwise, the Roulette method with a fixed probability is used to determine if a should be added into the route of salesman k or assigned to salesman $k+1$ for later selecting. This route search process generates in turn a path for each salesman, starting from and returning to its depot.

Considering the randomness of shared city selection, the route search process should be repeated to achieve a better result. Shared city set U_j can bring forth maximally $\sum_0^{q_j} C_{q_j}^i = 2^{q_j}$, $\forall j \in Z_{m-1}$, possible route combinations. Thus, the total running count should be $\max(2^{q_0+q_1}, 2^{q_1+q_2}, \ldots, 2^{q_{m-2}+q_{m-1}})$ for m salesmen. Algorithm 1 gives a detailed description of our greedy heuristics. It has a theoretical worst case running time of $O(2^N N^2)$.

Algorithm 1: Greedy algorithm for S-CTSP

```
1. Input: City distribution, $n_1, n_2, \ldots, n_m$, and probability $r$.
2. repeat
3.   $k := 1$     // Current salesman $k$
4.   repeat
5.     repeat    // Select cities for salesman $k$
6.       Select city $a$ from $\aleph_k$ satisfying proximity
7.       if $a \notin U_k$ then
8.         Add $a$ into $X_k$
9.       else assign $a$ to salesman $k$ or $k+1$ by Roulette with $r$
10.        if $a$ assigned to salesman $k$ then
11.          Add $a$ into $X_k$
12.        end if
13.      end if
14.      $\aleph_k := \aleph_k \backslash \{a\}$
15.    until more than $n_k$ cities were searched
16.    $k := k + 1$
17.  until $k > m$
18. until maximum of running is met
19. Output: the best solution $X = (X_1, X_2, \ldots, X_m)$
```

IV. S-CTSP-based Collision-free Scheduling of MBMS

A. Job Assignment, Scheduling, and Collision Avoidance Problems

The multi-bridge machining systems investigated herein have multiple bridge machines. A bridge moves along a long system rail and the machine on it performs tasks in a planar or spatial workspace that partially overlaps with its neighbor in series. For example, a waterjet cutting center and welding system have planar and spatial workspaces, as shown in Fig. 1, respectively.

As shown in Fig. 3, an MBMS has m individual bridge machines numbered from 1 to m. The entire system workspace can be divided into m non-overlapping workspaces and $m-1$ overlapping workspaces, denoted as W_i and $\bar{W}_j$, $\forall i \in Z_m = \{1, 2, \ldots, m\}$ and $\forall j \in Z_{m-1}$. Machine 1 has an accessible workspace consisting of W_1 and $\bar{W}_1$, and machine m, W_m and $\bar{W}_{m\text{-}1}$. The accessible workspace of machine $i \in Z_m \backslash \{1, m\}$ consists of $\bar{W}_{i-1}$, W_i, and $\bar{W}_i$.

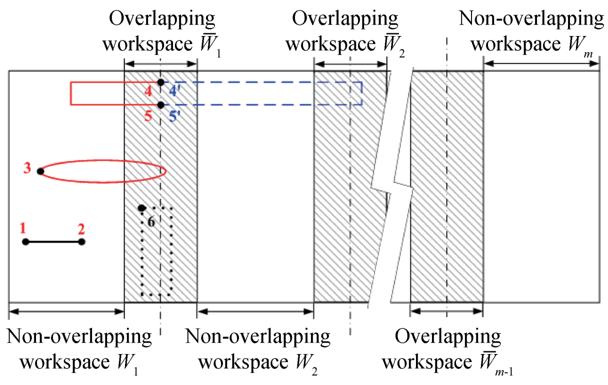

Overlapping workspace $\bar{W}_1$
Overlapping workspace $\bar{W}_2$
Non-overlapping workspace W_m
4
4'
5
5'
3
6
1
2
Non-overlapping workspace W_1
Non-overlapping workspace W_2
Overlapping workspace $\bar{W}_{m\text{-}1}$

Fig. 3. Topview of workspace partition of an MBMS.

Three problems arising from MBMS are job assignment, job scheduling, and collision resolution problems. Jobs are continuous line segments, curves, or shapes for machining. They usually distribute over different workspaces and even some cross multiple workspaces. First, these jobs should be partitioned and assigned to specific machines. Usually, there are many jobs in a large workspace in such systems. It is significant to find the best job sequences for multiple machines so as to improve processing efficiency and reduce energy consumption. It is basically a job scheduling problem. In addition, due to an overlapping processing region in such systems, two neighboring machines may collide when both co-locate in it to perform their respective tasks. Hence, one has to resolve this collision problem to ensure the system reliability.

The numbered lines and curves, either planar or spatial such as cutting curves and weld seams, are the tasks to be performed. Note that a basic principle to be followed by during task assignment is that, i.e., a task should be processed continuously by one machine as far as possible. Thus, we have the below policies of task partition and assignment:

a) A curve fully in a machine's accessible workspace and partially in its non-overlapping workspace must be assigned to it;

b) A task totally in a shared workspace can be assigned at random to either neighboring machines; and

c) A curve crossing over a shared workspace should be partitioned first by the central line of the workspace and then the segments are assigned to the particular machines according to a).

For example, the line segment with endpoints 1 and 2 and ellipse 3 meeting a), should be assigned to machine 1. Rectangle 6 totally in a shared workspace can be assigned to either neighboring machines. Following c), the long rectangle crossing over shared workspace 1 is partitioned into two non-overlapping segments, i.e., the left for machine 1 and the right for machine 2.

We call an assigned task or task segment a job. The jobs can be classified into several different types, namely, exclusive jobs for each machine and shared jobs for two neighboring machines. We note that the grouping of jobs is not in line with the partition of workspaces. For example, an exclusive job of a machine, e.g., job 3, meeting a) is not necessarily a complete one in its non-overlapping workspace or partially appears in its overlapping workspace, while a shared job of a machine, e.g., job 6, must fall entirely in one of its overlapping workspaces.

Each task can be abstracted as one or two dots with a processing duration. In particular, a closed curve will be abstracted to its most left point, e.g., 3 and 6; otherwise, it is abstracted into the two end points, e.g., 1 and 2. Such two endpoints cannot be separately accessed and the corresponding job's duration will be set as an attribute to the starting point, e.g., if point 1 is selected as a starting point, the processing duration of the line segment will be assigned to it and the endpoint of processing must be 2, accordingly.

B. Augmented Greedy Algorithm for Collision-free Scheduling

Job scheduling of MBMS can be modeled by an S-CTSP. The assigned jobs and individual machines are modeled as cities and salesmen in an S-CTSP, respectively. Namely, each group of exclusive (shared) jobs is represented as a set of exclusive (shared) cities. The objective of scheduling is to search a job sequence for each machine such that the total job visiting cost of all machines is the least. It can be formulated by the objective function of the corresponding S-CTSP with a specific definition of visit cost. Therefore, Algorithm 1 for S-CTSP is also for basic job scheduling of MBMS.

Collision resolution of any two neighboring bridges can be solved together by such a scheduling. A collision between two bridges must mean that their execution durations have an overlap in their workspaces. Therefore, it can be resolved by eliminating such overlaps when scheduling. Specifically, a solution can try its best to search a job sequence for a machine without such overlaps with its neighboring machine; and remove such an overlap only when it is inevitable in search.

We define the ith time window of bridge k as the execution duration defined from its entering time to its exiting time its overlapping workspace k', denoted as $[s_{k',i}, e_{k',i}]$. Given a route X_k, the time window sequence of bridge k, $T_{k,k'}$ is a sequence, $[s_{k',1}, e_{k',1}]$, $[s_{k',2}, e_{k',2}]$, $\ldots$, and $[s_{k',l}, e_{k',l}]$, where l is the times of bridge k entering overlapping workspace k', $k' = k$ and $k' \neq m$, $\forall k \in Z_m$, Before adding a current job into the present job sequence of the current machine, it is necessary to adjust if the job has any time overlaps with those of the previous bridge if it exists, in the corresponding overlapping workspace. When its job sequence is in searching, its time window sequence is unavailable. Therefore, it needs to define a job's nominal time window (NTW) to represent the period from when a bridge enters an overlapping workspace from the outside to execute the job to when it exits this workspace. NTW is an extension to the job processing period. It makes it possible to estimate potential time overlaps between the current bridge and its previous neighbor.

During search for a job sequence of the current machine, two cases of overlap resolution should be taken into account, as shown in Fig. 4, if such a time overlap cannot be avoided by

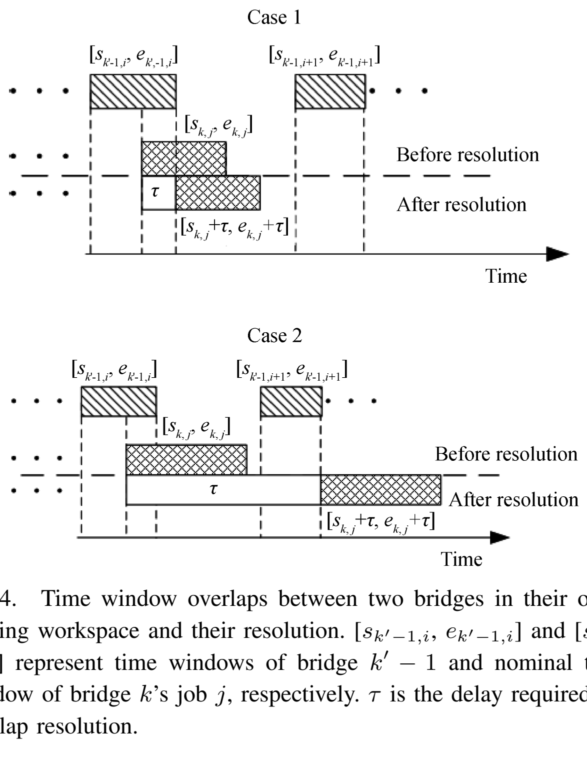

Case 1
$[s_{k'-1,i}, e_{k'-1,i}]$
$[s_{k'-1,i+1}, e_{k'-1,i+1}]$
$[s_{k,j}, e_{k,j}]$
Before resolution
τ
After resolution
$[s_{k,j}+\tau, e_{k,j}+\tau]$
Time
Case 2
$[s_{k'-1,i}, e_{k'-1,i}]$
$[s_{k'-1,i+1}, e_{k'-1,i+1}]$
$[s_{k,j}, e_{k,j}]$
Before resolution
τ
After resolution
$[s_{k,j}+\tau, e_{k,j}+\tau]$
Time

Fig. 4. Time window overlaps between two bridges in their overlapping workspace and their resolution. $[s_{k'-1,i}, e_{k'-1,i}]$ and $[s_{k,j}, e_{k,j}]$ represent time windows of bridge $k' - 1$ and nominal time window of bridge k's job j, respectively. τ is the delay required for overlap resolution.

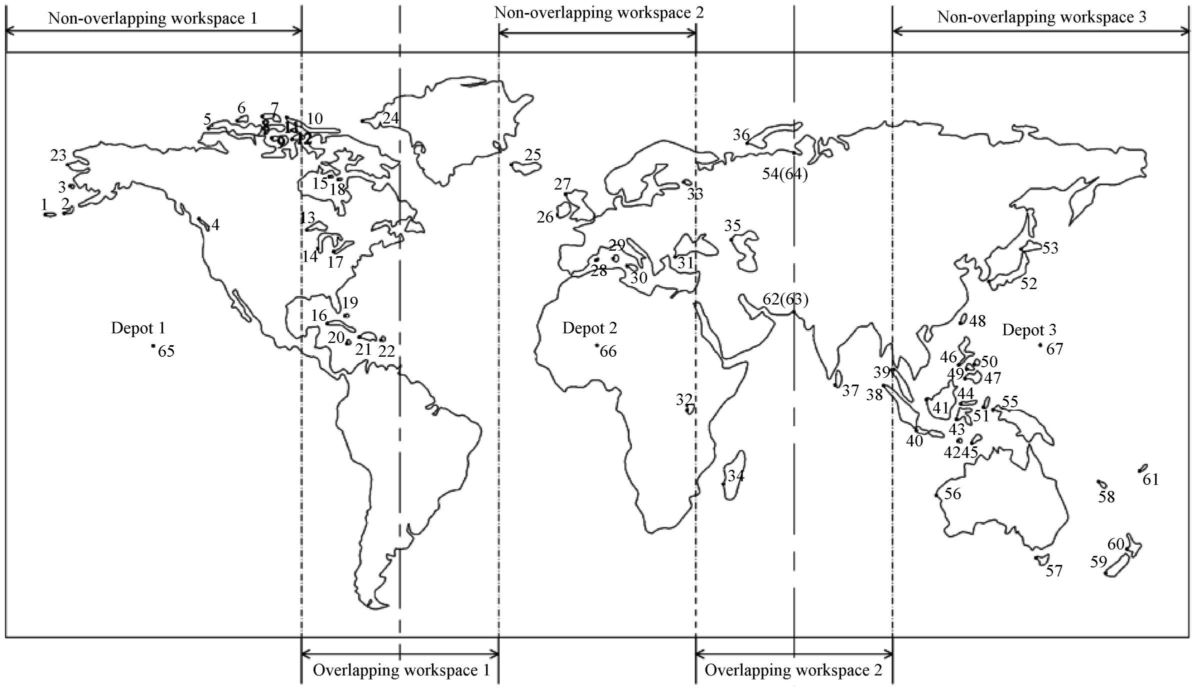

Non-overlapping workspace 1
Non-overlapping workspace 2
Non-overlapping workspace 3
Overlapping workspace 1
Overlapping workspace 2
Depot 1
Depot 2
Depot 3
65
66
67

Fig. 5. Layout of triple-bridge waterjet cutting and a world map to be cut.

reselecting another job. First, the overlap between the NTW of a to-be-executed job j by bridge k and one of the execution time windows of bridge $k-1$ in their overlapping workspace can be removed, once the former is shifted by a time span. Second, if a new overlap is brought by such a shift, it needs one more shift until no overlaps exist. Thus, solving both problems of an MBMS together can be realized by repeating the above process for bridges one by one.

We augment the greedy algorithm to deal with it as given in Algorithm 2. It takes $O(2^N N^2)$ time in the worst case as same as Algorithm 1. Step 12 represents that bridge k cannot enter region $\bar{W}_{k'-1}$ until its neighbor $k-1$ leaves the region so as to resolve the potential conflicts between them. As mentioned in Section II, some exclusive jobs may fall partially in the overlapping workspace besides the shared jobs. Therefore, it is necessary to analyze and deal with the emerging time window overlaps for an exclusive job once it appears in an overlapping workspace.

V. Case Study of Multi-bridge Waterjet Cutting

At present, triple-bridge water cutting is under development. There are neither algorithms nor related CAM software, released for triple-bridge water cutting, to the best knowledge of the authors. In this section we apply the proposed greedy algorithms for S-CTSP and collision-free scheduling to a triple-bridge waterjet cutting application. Fig. 5 illustrates the sketch of the center and a world map to be cut by it. Its maximum cutting size is 12 m × 6 m, where the maximum travel of bridges 1−3 is 5 m, 6 m, and 5m, respectively. The three non-overlapping workspaces and two overlapping ones have been marked out. In the map, all separated land shapes are numbered and to be cut from 16 marble baseplates of size 3 m × 1.5 m × 2 cm. The resultant cavities will be filled in with marble blocks of the same shapes but different colors so as to form a mosaic artwork for wall decoration. The traveling and cutting speeds of the machine tool are 10 m/min and 40 cm/min, respectively.

Algorithm 2: Augmented greedy algorithm for collision-free scheduling of MBMS

```
1. Input: $n_1, n_2, \ldots, n_m$, $r$, cities distribution, workhead traveling
   and execution speed, and job execution time
2. Omitted // As same as Steps 2−4, Algorithm 1
3.   **repeat** // Select jobs to be visited by machine $k$
4.     Calculate the rest jobs' NTW set $\tilde{T}_{k,k'-1}$ in $\bar{W}_{k'-1}$
5.     **if** overlaps exist between $T_{k-1,k'-1}$ and $\tilde{T}_{k,k'-1}$
6.       Add the jobs into set $M_k^i$
7.       **if** $\aleph_k \neq M_k^i$ **then**
8.         Select job $a$ from $\aleph_k - M_k^i$
9.         Omitted // As same as Steps 7−13, Algorithm 1
10.      **else then**
11.        Select job $a$ from $\aleph_k$and add into $X_k$
12.        Shift $a$'s NTW to resolve the time overlaps
13.      **end if**
14.    **end if**
15.  **until** more than $n_k$ cities searched
16. Omitted // refer to Steps 16−18, Algorithm 1
17. Output: the best solution $X = (X_1, X_2, \ldots, X_m)$
```

A. Cutting Path Planning Considering No Collisions

The land shapes to be cut are numbered from 1 to 64 and the black spots represent the starting or ending spots. Jobs are assigned, in accordance with the policies given in Section

II, as follows: jobs 1–12 and 23 as the exclusive ones for bridge 1, jobs 24–33 for bridge 2, and jobs 38–61 for bridge 3. The close outline of the Eurasian and African continent is partitioned by the central line of the first overlapping workspace into two unclosed curves that are assigned to bridges 2 and 3, i.e., the left curve with two endpoints 54 and 62, respectively. Jobs 13–22 in overlapping region 1 are shared by bridges 1 and 2 and jobs 34–37 in overlapping region 2 are shared by bridges 2 and 3. Points 65–67 represent the depots of bridges 1–3, respectively.

The optimization objective is to search a route for each head such that the total tour length of all heads is the shortest. The cutting path planning problem of the tri-bridge system can be modeled as an S-CTSP with $m = 3$ and $n = 67$. Namely, the three bridges are modeled as traveling salesmen, three collections of exclusive jobs as three sets of exclusive cities and the two sets of shared jobs as two sets of shared cities. Note that, it is necessary to sequence continuously both endpoints 54 and 62 of the left unclosed curve and endpoints 63 and 64 of the right unclosed curve when planning the cutting paths to represent that the two curves should be cut fully.

Algorithm 1 is used to solve the S-CTSP. The accessible city counts of the three salesmen are $n_1 = 23$, $n_2 = 27$, and $n_3 = 30$; and shared city counts are $q_1 = 10$ and $q_2 = 4$. The maximum number of iterations is $2^{14} = 16\,384$. The probability of city selection r is set to 0.7. The algorithms are implemented in C# language with Microsoft Visual Studio 2012 and runs on Dell Optiplex 390 installed with Windows 7 with CPU Intel Core i3-2130 at 3.40 GHz and 4 GB RAM. After discarding the duplicate solutions, the total tour length (L) of 10 000 different intermediate ones generated by Algorithm I is plotted in Fig. 6, where L is the distance sum between two shape endpoints, i.e., the total distance of rapid travel. The obtained minimal L, denoted by L^*, is 487 563 mm and mean L denoted by $\bar{L}$, 52 850.5 mm. The total computing time spent is about 3.0 seconds. Taking arbitrarily 10 fixed assignments of the shared cities, we perform accordingly the basic greedy algorithm without randomized modification 10 times and obtain the worse results than Algorithm I does, i.e., bigger $L^* = 50\,941.2$ mm and $\bar{L} = 54\,357.1$ mm, as shown in Fig. 7. The mean computing time is 0.0014 seconds. Predetermined assignment of shared cities transforms S-CTSP into multiple individual TSP. The results indicate that S-CTSP has greater solution space than multiple individual TSP with the same problem size.

B. Collision-free Cutting Path Planning

We apply the augmented greedy heuristics to solve collision-free cutting paths for the three bridges and the algorithm parameters remain as before. Its obtained results are listed in Table I. L^* is 50 045.6 mm when the maximum of concurrent processing time (MCPT) spent by the bridges is 1338 minutes, including travel and cutting time. A resolution of time window overlap has to be performed, thereby resulting in a delay of 18 minutes. The basic greedy algorithm with collision resolution is also applied to solve collision-free paths for the purpose of comparison. The ten assignments of shared cities above are adopted. It shows that the obtained best and mean results are worse than those of Algorithm 2 with respect to L^*, MCPT, and collision resolution delays (CRD) as listed in Table I. The total computing time spent by Algorithm 2 and the basic greedy algorithm with collision resolution are 3.5 and 0.015 seconds, respectively. We also note that collision-free path schedules are achieved at the cost of route length and processing delays.

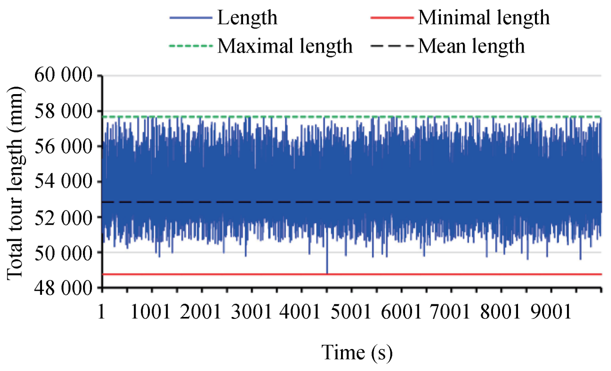


Fig. 6. Total route length of 10 000 different intermediate results obtained by Algorithm 1.

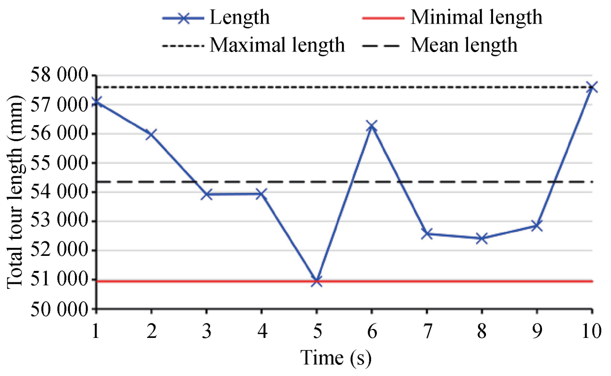


Fig. 7. Total route length of 10 cases with predetermined shared city assignment by the basic greedy algorithm.

TABLE I
RESULTS COMPARISON OF COLLISION-FREE PATH PLANNING
(UNITS: MM AND MIN)

		L^*	MCPT	CRC	CRD
Algorithm 2	Best solution	50 045.6	112.7	1	18.0
	Worst solution	55 830.5	152.9	4	52.1
	$\bar{x}$	53 551.8	133.8	2.3	37.8
BGAC	Best solution	52855.4	113.8	1	20.2
	Worst solution	55 301.2	152.9	4	52.1
	$\bar{x}$	54 276.6	135.6	2.8	37.9

Note: CRC represents collision resolution counts; BGAC is the basic greedy algorithm with collision resolution.

We exploit the plug-in component for multi-bridge waterjet cutting simulation under the platform of AutoCAD to validate the above twenty solutions. It is built with the ObjectARX programming environment by our group (Du, 2011) and can simulate the travel and cutting processes of three workheads. The results show that no cases of two bridges both exist in the same overlapping processing region at the same time. It indicates that the potential collisions of bridges are avoided by all solutions in Table I. Also, with respect to the solutions with collision resolution, a bridge has to wait to enter an overlapping zone before its neighbor exits from it. In addition, we also note that a change in the timing and sequencing of a solution, e.g., a change in the sequence of jobs or insertion of waiting, may incur new collisions of bridges. Such a change represents a fault. Usually, all bridges have to halt and resume their execution after the fault treatment.

VI. Conclusions

This work investigates the job scheduling and coordination problems commonly faced by multi-bridge machining systems. First, a serial colored traveling salesman problem and its greedy algorithm are proposed. The basic scheduling problems of MBMS can be modeled by S-CTSP and solved via our algorithms after the job partition and assignment following some process requirements. An integrated method is developed to solve simultaneously the job scheduling and coordination problems of MBMS. It is implemented by integrating timing constraints into the greedy algorithm to avoid collisions between any two neighboring bridges. Finally, we apply both to a triple-bridge waterjet cutting process and compare their performance. Our ongoing work is to develop new heuristics and evolutionary algorithms. In addition, to achieve fault-tolerance ability for MBMS, we also intend to investigate dynamic multi-machine coordination solutions and combine them with S-CTSP-based scheduling. These methods will be applied to the challenging manipulation problem of large-size 3D printing.

References

[1] Z. Du, "A coordination and optimization method for cutting processes of multi-bridge waterjet systems," M.S. thesis, School of Automation, Southeast University, China, 2011.

[2] J. Li, Q. Sun, and X. Dai, "A coordination and optimization method for multi-bridge waterjet cutting processes," in *Proc. 42nd International Conference on Computers & Industrial Engineering*, Cape Town, South Africa, pp. 9575−9580, 2012.

[3] X. Meng, J. Li, M. C. Zhou, and X. Dai, "Improved population-based incremental learning algorithm for scheduling multi-bridge waterjet cutting processes," in *Proc. IEEE 11th International Conference on Networking, Sensing and Control (ICNSC)*, Miami, FL, USA, pp. 496−500, 2014.

[4] Q. Sun, "A colored traveling salesman problem and its application," M.S. thesis, *School of Automation*, Southeast University, China, 2013.

[5] J. Li, Q. Sun, M. C. Zhou, and X. Dai, "A new multiple traveling salesman problem and its genetic algorithm-based solution," in *Proc. 2013 IEEE International Conference on Systems, Man, and Cybernetics* (SMC), Manchester, UK, pp. 627−632, 2013.

[6] J. Li, Q. Sun, M. C. Zhou, X. Yu, and X. Dai, "Colored traveling salesman problem and solution," in *Proc. 19th World Congress of the International Federation of Automatic Control*, Cape Town, South Africa August 24−29, vol. 47, no. 3, pp. 9575−9580, 2014.

[7] J. Li, M. C. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem. *IEEE Transactions on Cybernetics*," vol. 45, no. 11, pp. 2390−2401, 2015.

[8] J. Li, X. Meng, M. C. Zhou, and X. Dai, "A two-stage approach to path planning and collision avoidance of multibridge machining systems," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, DOI: 10.1109/TSMC.2016.2531648.

[9] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures" *Omega*, vol. 34, no. 3, pp. 209−219, 2006.

[10] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "The traveling salesman problem," *Princeton: Princeton University Press*, 2006.

[11] N. Chakraborty, S. Akella, and J. T. Wen, "Coverage of a planar point set with multiple robots subject to geometric constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 111−122, 2010.

[12] E. K. Xidias, P. Th. Zacharia, and N. A. Aspragathos, "Time-optimal task scheduling for two robotic manipulators operating in a three-dimensional environment," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 224, no. 7, pp. 845−855, 2010.

[13] A. G. Gonzalez-Rodriguez and A. Gonzalez-Rodriguez, "Collision-free motion planning and scheduling," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 3, pp. 657−665.

[14] R. Kala, "Multi-robot path planning using co-evolutionary genetic programming," *Expert Systems with Applications* vol. 39, no. 3, pp. 3817−3831, 2012.

[15] H. Qu, K. Xing, and T. Alexander, "An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots," *Neurocomputing* vol. 120, pp. 509−517, 2013.

[16] H. J. Yoon, "Scheduling for deadlock avoidance operation in robotic manufacturing cells," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 224, no. 2, pp. 329−340, 2010.

[17] T. S. Dahl, M. Mataric, and G. S. Sukhatme, "Multi-robot task allocation through vacancy chain scheduling," *Robotics and Autonomous Systems*, vol. 57, no. 6−7, pp. 674−687, 2009.

[18] G. A. Korsah, B. Kannan, B. Browning, A. Stentz, and M. B. Dias, "xBots: an approach to generating and executing optimal multi-robot plans with cross-schedule dependencies," in *Proc. IEEE International Conference on Robotics and Automation (ICRA 2012)*, St. Paul, MN, USA, pp. 115−122, 2012.

[19] T. Zheng and J. Li, "Multi-robot task allocation and scheduling based on fish swarm algorithm," in *Proc. 8th World Congress on Intelligent Control and Automation (WCICA 2010)*, Jinan, China, pp. 6681−6685, 2010.

[20] Y. Gan, X. Dai, and J. Li, "Cooperative path planning and constraints analysis for master-slave industrial robots," *International Journal of Advanced Robotic Systems*, vol. 9, no. 88, pp. 1−13, 2012.

[21] G. Gutin, A. Yeo, and A. Zverovich, "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP," *Discrete Applied Mathematics*, vol. 117, no. 1−3, pp. 81−86, 2002.

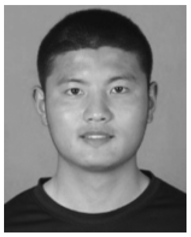
[22] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study in local optimization," *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (eds.), John Wiley and Sons, London, pp. 215−310, 1997.

Jun Li (M'07–SM'13) received the M.S. degree in mechanical engineering from Jiangsu University, Zhenjiang, China, in 2003, and the Ph.D. degree in control theory and control engineering from Southeast University, Nanjing, China, in 2007.

From 2008 to 2010, he was a post-doctoral fellow with Southeast University, where he is currently an Associate Professor with the School of Automation. In 2014, he was a Visiting Scholar with the New Jersey Institute of Technology, Newark, NJ, USA. He has been the Principal Investigator of ten national and provincial research projects. His innovative achievements over the past four years include proposing the colored traveling salesman problem, which exploits colors to distinguish the accessibility of cities toward salesmen, an image-guided touch teaching method for industrial manipulators in smart phone testing, and the lean reachability tree for general unbounded Petri nets.

He has published over 30 journal and conference papers and holds five invention patents. His current research interests include robotics, machine learning, optimization and control of discrete production systems, and Petri nets.

Dr. Li was a recipient of the Best Ph.D. Dissertation Award of Southeast University, the Outstanding Young Teacher Foundation Award of Southeast University, the Phoenix Contract-Southeast University Teaching Award, and the Best Paper Award of the 12th IEEE International Conference on Networking, Sensing and Control (ICNSC'2015). He was invited to lecture in the IEEE Northern New Jersey Section in 2014 and National Central University, Taipei, in 2015. He served as a Program Committee Member of the 9th and 10th IEEE International Conference on Networking, Sensing and Control. He organized a special session at the ICNSC'2014, Miami, FL, USA, and co-sponsored an international seminar "Discrete Event System Control and Optimization" at Southeast University in 2016. He is a Peer Reviewer of 10+ international journals and NSFC proposals. He is a Senior Member of the China Computer Federation (CCF) and the Chinese Mechanical Engineering Society, and a member of ACM, CCF Technical Committee on Petri nets, the Chinese Association of Automation Technical Committee on Network Information Service, and the Jiangsu Province Computer Federation Technical Committee on Big Data.



Xianghu Meng received the B.S. degree from Shenyang Institute of Engineering, Shenyang, China in 2008, and M.S. degree from Kunming University of Science and Technology, Kunming, China in 2013. He is presently a Ph.D. candidate of Southeast University, Nanjing, China. His current research interests include evolutionary algorithms and production scheduling.

Xing Dai received the B.S. degree from Hunan University of Technology, Zhuzhou, China in 2014. He is currently a postgraduate in control theory and control engineering with the School of Automation, Southeast University, Nanjing, China. His current research interests include evolutionary algorithms and production scheduling.