





Multiobjective Scheduling Strategy With Genetic Algorithm and Time-Enhanced A* Planning for Autonomous Parking Robotics in High-Density Unmanned Parking Lots

Guang Chen , Member, IEEE, Jing Hou , Jinhu Dong, Zhijun Li , Senior Member, IEEE, Shangding Gu, Bo Zhang, Junwei Yu, and Alois Knoll , Senior Member, IEEE

Abstract—With the process of urbanization, the problem of insufficient parking spaces has become prominent. Adopting a high-density parking lot with parking robots can greatly improve the land utilization rate of the parking lot. This article tackles the multiple parking robots scheduling problem of high-density layout parking lots, including task execution sequence decision, robot allocation, and cooperative path planning. First, we mathematically describe the parking robot scheduling problem. Existing approximation algorithms are often far from the optimal solution. This article proposes an improved genetic algorithm and a time-enhanced A* path planning algorithm for high-density parking lots. The improved genetic algorithm can efficiently search task execution sequence and robot allocation and converge to the optimal solution even in large-scale complex scenarios. Meanwhile, the time-enhanced A* algorithm takes a new dimension “the time” into consideration, together with the distance, and security factors, to solve the multi-parking-robot path planning problem. Simulation

experiments show that our algorithm can improve scheduling performance in many aspects such as task execution time, driving distance, and security in large-scale high-density parking lots. This article provides an efficient and convenient scheduling solution for the implementation of the high-density unmanned parking lot.

Index Terms—Genetic algorithm, high-density automatic parking, multirobot systems, optimal scheduling algorithm.

I. INTRODUCTION

WITH the increase of vehicle ownership, achieving effective parking and increasing the space utilization of the parking lot has been an essential problem of today’s society. Therefore, the unmanned parking lot that uses the autonomous parking robot to execute pick-up or set-down operation will be the future development direction. Recently, with the development of perception technology, researches utilize intelligent sensors and devices to improve the efficiency of the traditional parking (TP) lots [1]–[4]. Using the parking robot in a parking lot can increase the effectiveness of parking and decrease the time that drivers spend in parking [5]. Although they are able to improve the management intelligence of the parking lot, the intelligent device does not have the ability to fundamentally increase the land use efficiency of the parking lot. Therefore, a high-density parking (HDP) lot (see Fig. 1) needs to be designed for the revolutionary improvement of the parking lot.

A. Literature Review

1) *High-Density Parking*: The use of parking robots can change the TP lot to unmanned parking lot, even to the HDP lot. For example, as the whole process of parking does not require the involvement of users, the high-density layout can be used in the unmanned parking lot [see Fig. 1(a)]. In this case, the parking positions are closely arranged without the room for roads and car doors, so the land use efficiency of the parking lot can be greatly improved. Serpen and Dou [2] focus on strategies such as spatial scheduling and path planning for HDP lots, enabling parking lots to prevent deadlocks and operate efficiently.

Wu *et al.* [6] design a mechanical multistorey garage, which uses a rotary vehicle lift and is surrounded by HDP spaces.

Manuscript received February 1, 2020; revised April 10, 2020; accepted August 28, 2020. Date of publication September 10, 2020; date of current version June 15, 2021. Recommended by Technical Editor Y. Liu and Senior Editor K. Kyriakopoulos. This work was supported in part by the European Union’s Horizon 2020 Framework Program for Research and Innovation under the Specific Grant Agreement 945539 (Human Brain Project SGA3), in part by the Shanghai AI Innovation Development Program 2018, and in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China under Grant 2018AAA0102900. (Corresponding author: Zhijun Li.)

Guang Chen is with the Department of Automotive Engineering, Tongji University, Shanghai 200092, China, and also with the Technical University of Munich, 85748 Munich, Germany (e-mail: guangchen@tongji.edu.cn).

Jing Hou, Jinhu Dong, and Junwei Yu are with the Department of Automotive Engineering, Tongji University, Shanghai 200092, China (e-mail: 1911061@tongji.edu.cn; 317684920@qq.com; 1510997@tongji.edu.cn).

Zhijun Li is with the Department of Automation, University of Science and Technology of China, Hefei 230026, China (e-mail: zjli@ieee.org).

Shangding Gu is with the Wuhan University of Technology, Wuhan 430070, China (e-mail: gushangding@gmail.com).

Bo Zhang is with the Autonomous Driving Group, Shanghai Westwell Information and Technology Company Ltd., Shanghai 200050, China (e-mail: bob.zhang@westwell-lab.com).

Alois Knoll is with the Technical University of Munich, Munich 85748, Germany (e-mail: knoll@in.tum.de).

Color versions of one or more of the figures in this article are available online at <https://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TMECH.2020.3023261

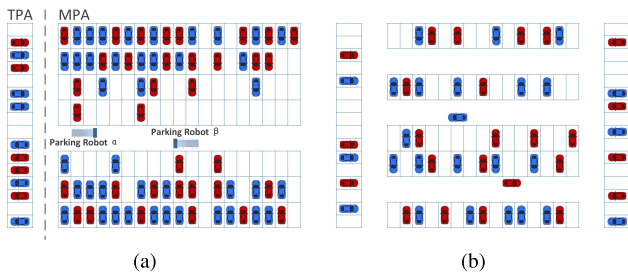


Fig. 1. Comparison between the HDP and TP lot. (a) Typical stack-based layout of an HDP lot with 149 parking positions. Vehicles need to be handled by parking robots in the temporary parking area on the left and the high-density storage area on the right. (b) Typical TP lot layout with 107 parking positions. Vehicles are free to drive and park on their own in TP lot. The TP lot can be modified to unmanned parking lot with parking robotics.

This type of equipment does not require the vehicle to have an automatic driving function and has a higher parking density due to the full use of multilayer space. But with the increasing maturity of autonomous driving technology, the construction cost of large mechanical multistorey garages is much higher than the input cost of several parking robots in HDP lots.

In the field of the HDP lot, the stack-based layout is widely adopted because of its convenience and efficiency [5], [7]. Nourinejad *et al.* [5] focus on the structural design of stack-based HDP lots and study the impact of stack depth on the operational efficiency of the parking lot.

d'Orey *et al.* [7] propose five different stack scheduling selection strategies of stack-based HDP layout and conduct comparative analysis. However, the above algorithms only determine the spatial location where the vehicles should be stored in high density, which is suitable for vehicles entering and exiting a small parking lot in sequence. The algorithm we proposed is suitable for large-scale parking lots where multiple cars are moved in and out by multiple parking robots.

2) Task Execution Sequence Scheduling: In addition to HDP task generation strategy, the task execution sequence scheduling algorithm for HDP lots is also worth studying. The scheduling algorithm of the HDP lots mainly draws on the execution sequence of the terminal container dispatching task [8], [9] such as using a genetic algorithm, colony, auction algorithm, branch and bound algorithm to solve the model of integer programming model. Genetic algorithm combined with a new decoding method for total delay target has been used for hybrid flow shop scheduling problem [8]. The branch-price-and-cut algorithm can also be used to solve the scheduling problem with the stack structure and time windows, but obviously the calculation time is unacceptable when the scene size is too large [9].

In this article, improved genetic algorithm (IGA) combined with Tabu search not only considers the priority of stacked parking, but also can quickly obtain the scheduling results in large-scale scenarios.

3) Multirobot Collision-Free Path Planning: After the task scheduling is assigned to each parking robot, the multirobot collaborative path planning algorithm needs to plan a reasonable collision-free path for each robot. The research on path planning

is relatively mature, especially the multirobot collaborative planning [10]–[16].

The multi-automated guided vehicle (AGV) path planning algorithm for warehouses is generally divided into two steps: initial path planning and conflict resolution [13], [14]. This leads to complex types and solutions of conflicts, and it is difficult to guarantee the optimality of the path due to excessive security space reservation.

Inspired by unmanned aerial vehicle's (UAV) three-dimensional (3-D) A* path planning algorithm [15], a time-enhanced A* (TEA*) algorithm was proposed [16]. Unlike the height dimension in the 3-D A* algorithm that can change up or down, the time dimension in the TEA* algorithm can only flow in one direction, which brings many differences between the two algorithms.

The TEA* algorithm is ideally suitable for automated warehouse transportation scenarios where vehicles communicate with each other and there are no outside vehicles. It can realize safe and efficient multirobot collaborative path planning.

This article further designs the multirobot relative priority relationship of the TEA* algorithm, so that it can handle multirobot collaborative planning in HDP lots. In summary, to the best of our knowledge, the HDP lot multirobot scheduling and planning problem considered in this article is hardly solved by genetic algorithms and A* algorithms in the literature.

B. Our Contributions

Our proposed scheduling system mainly consists of the task execution sequence scheduling and path planning algorithm, aiming to solve the multiple parking robot scheduling problem for the HDP lots.

In this article, the genetic algorithm (GA)-based task execution sequence scheduling algorithm is greatly improved with the help of directional variation based on Tabu search. Based on the TEA* algorithm, we expand the related theories to solve the priority problem in the stacked-based HDP lots. Through the simulation experiment, the realization of HDP has greatly improved land utilization, and the combined improved scheduling algorithm applied in HDP lots can significantly improve the operation efficiency and vehicle travel distance of the parking lot. The main contributions of this article are as follows:

- 1) A mathematical formulation of the HDP problem is designed for subsequent scheduling algorithms.
- 2) We propose an IGA with large mutation probability and a chromosome calibration method based on Tabu search to ensure the task execution sequence satisfying the priority relationship.
- 3) The 3-D searching strategy based on the TEA* algorithm is proposed to generate multirobot collaborative planned path.

II. HDP SCHEDULING PROBLEM DESCRIPTION

The layout of the HDP lot is different with the TP lot in many aspects (see Fig. 1). In a TP lot, the parking robot will not be disturbed by other robots. However, in an HDP lot, there will be mutual interference in pick-up and set-down operations. Fig. 2

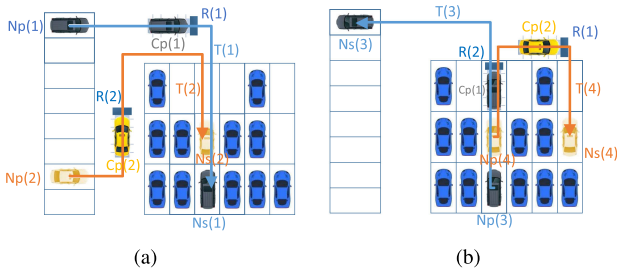


Fig. 2. (a) Set-down operation [Task T(1): Black Ca Cp(1): Np(1) → Ns(1); Task T(2): Yellow Car Cp(2): Np(2) → Ns(2)] by parking robots R(1) and R(2). Because the black vehicle is parked deeper, the Th(2) (relatively high priority task for handling the yellow vehicle) is the T(1) (setting the black car down). (b) Pick-up operation [Task T(3): Black Car Cp(1): Np(3) → Ns(3)]. The yellow car has to move to another parking position [Task T(4)] to clear the path for the black car pick-up operation.

shows the vehicle pick-up and set-down operations in an HDP lot.

The purpose of scheduling is to ensure the total parking lot operating effectively and inexpensively. A reasonable scheduling algorithm is needed to enable as few parking robots as possible to complete the handling task. Meanwhile, a good path planning algorithm can reduce the distance traveled by the vehicle and reduce power consumption.

The HDP scheduling problem can be regarded as a kind of the warehouse AGV scheduling problem, which needs to consider the stack-based vehicle storage structure and the priority of the handling tasks it brings.

A. Geometric Properties of the Facility

In a typical HDP lot like Fig. 1(a), the parking lot is divided into temporary parking area (TPA) and main parking area (MPA). People drive their cars to the TPA, set the retrieve time, and leave the parking lot. Then, the parking robots will come to the TPA and bring the cars to the MPA and store them in a reasonable order. When people are going to retrieve their cars, the parking robots will bring their cars back to the TPA. The MPA consists of many parking stacks. The stacks follow the last in first out (LIFO) principle. When a car in the stack is going to get out, all shallower cars in the stack must be moved out first.

B. Scheduling System Architecture

The scheduling system architecture mainly includes tasks scheduling algorithm and path planning algorithm (see Fig. 3). When a vehicle comes to the parking lot, it will be stored at the TPA. The customer can set the estimated time to retrieve his vehicle and leave the parking lot. Then, when the TPA is almost full or the scheduling program has not been triggered for a long time, the scheduling algorithm will be triggered. When there are previous tasks that have not been completed, the algorithm will adopt the rescheduling strategy. Otherwise, it will adopt the normal scheduling strategy. Three main components of the scheduling system are tasks generation, task execution sequence scheduling and the multirobot path planning.

1) **Task Generation Strategy:** The task generation strategy mainly determines the parking position of the vehicle. The task

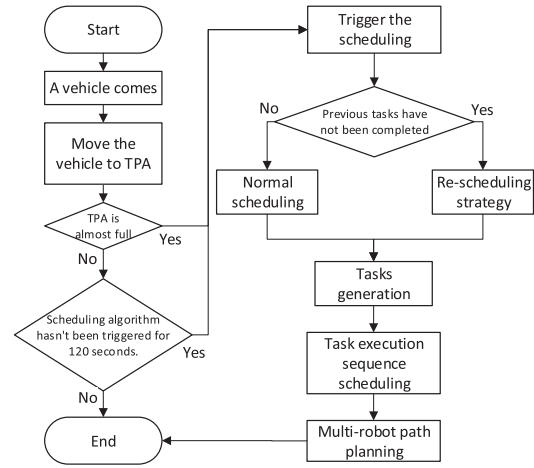


Fig. 3. Scheduling system architecture.

generation model will create the handling tasks of this car and the car that interferes with it according to its estimated departure time.

Relevant work has been studied by many scholars [5], [7], [17], [18], and this article will not delve into this field. The algorithm will generate handling tasks, including the start positions, the goal positions and the relative priorities, similar to the conditional order on arrival (“COA”) method in [7].

2) **Task Execution Sequence Scheduling:** The task scheduling algorithm determines the task execution sequence and task allocation for each parking robot. To prevent the path interference and ensure the safety operation of the HDP, the relative priorities of the handling tasks must be guaranteed in the task execution sequence scheduling.

3) **Multirobot Path Planning:** The path planning algorithm generates collision-free paths for multiple parking robots. It is also of great importance to confirm relative priorities of the tasks to ensure the priority. The planned path considers multiple factors including path safety, path length, and path time consumption.

C. Definitions of Model Parameters and Variables

In this section, model parameters and variables according to the work requirements are defined in detail. The given parameters and defined variables are described as follows (some of the variables are described in Fig. 2):

- 1) C : set of all cars.
- 2) C_p : set of the position of all cars.
- 3) C_r : set of the request situation of all cars.
- 4) R : set of all parking robots.
- 5) T : sets of all tasks.
- 6) T_h : sets of priority task, $T(i)$'s high priority task is the $T_h(i)$.
- 7) N_p : set of all pick-up nodes.
- 8) N_s : set of all set-down nodes.
- 9) N_j : set of all pick-up and set-down nodes, $N_j = N_p + N_s$.
- 10) N_r^i : set of all robots initial nodes.
- 11) N_r^f : set of all robots final nodes.

- 12) N_a : set of all nodes (pick-up nodes, set-down nodes, robots initial nodes, and robots final nodes), $N_a = N_j + N_r^i + N_r^f$.
- 13) n_r : the total number of parking robots.
- 14) n_j : the total number of jobs.
- 15) Δt_p : the time of each parking robot picking up a single vehicle.
- 16) Δt_d : the time of each parking robot setting down a single vehicle.
- 17) Δt_{ij} : the travel time of a parking robot from node i to node j , node i , and node j belonging to N_a .
- 18) λ_1 : the weighting factor of parking lot operating cost.
- 19) λ_2 : the weighting factor of parking lot time cost.
- 20) ${}^aT_i^r$: the time robot r arrives at node i .
- 21) ${}^lT_i^r$: the time robot r leaves node i .
- 22) P_{ij}^r : the path of the robot r moving from the node i to the node j .
- 23) t_r : the total time the robot r has completed all the tasks.
- 24) $x_{ij}^r \in \{0, 1\}$: binary integer decision variable, $r \in R$, $i, j \in N_a$, $i \neq j$, if robot r moves from node i to node j , $x_{ij}^r = 1$, otherwise $x_{ij}^r = 0$.

D. Model Formulation

We model the scheduling problem as the following binary integer programming formulation.

Objective function

$$Q_1 = \sum_{v \in V} t_r \quad (1)$$

$$t_r = \max_{i \in P} ({}^aT_i^r) \quad r \in R \quad (2)$$

$$Q_2 = \max_{r \in R} (t_r) \quad (3)$$

$$\text{Minimize : } Q = \lambda_1 \cdot Q_1 + \lambda_2 \cdot Q_2. \quad (4)$$

Constraint

$$\sum_{r \in R} \sum_{i \in N_p} X_{ii+n}^r = n \quad (5)$$

$$\sum_{r \in R} \sum_{i \in N_a} x_{ij}^r = 1 \quad j \in N_p \quad (6)$$

$$\sum_{j \in N_p + N_r^f(v)} x_{ij}^r = 1 \quad r \in R, i = N_r^i(r) \quad (7)$$

$$\sum_{i \in N_s + N_r^i(r)} x_{ij}^r = 1 \quad r \in R, j = N_r^f(r) \quad (8)$$

$$\sum_{j \in N_a} x_{ij}^r - x_{ik}^r = 0 \quad r \in R, i \in N_p, k = N_s(i) \quad (9)$$

$$\sum_{j \in N_a} x_{ij}^r - \sum_{j \in N_r^f(r) + N_p} x_{ij}^r = 0 \quad i \in N_s \quad (10)$$

$${}^aT_j^r \geq x_{ij}^r \cdot ({}^lT_i^r + t_{ij}) \quad i \in N_a, j \in N_a, r \in R \quad (11)$$

$${}^lT_j^r \geq x_{ij}^r \cdot ({}^aT_j^r + \Delta t_p) \quad i \in N_a, j \in N_p, r \in R \quad (12)$$

$${}^lT_j^r \geq x_{ij}^r \cdot ({}^aT_j^r + \Delta t_d) \quad i \in N_a, j \in N_s, r \in R \quad (13)$$

$${}^aT_i^r = {}^lT_i^r = 0 \quad r \in R, i = N_r^i(r) \quad (14)$$

$${}^aT_i^r = {}^lT_i^r \quad r \in R, i = N_r^f(r) \quad (15)$$

$$h_{ij} \cdot \left(\sum_{r \in R} \sum_{k \in N_a} x_{kN_p(i)}^r {}^lT_i^r - \sum_{r \in R} \sum_{k \in N_a} x_{kN_p(j)}^r {}^aT_j^r \right) \leq 0 \quad i, j \in N_p. \quad (16)$$

Function (1) is the parking lot operating cost function and function (3) is the parking lot time cost function. Function (2) indicates that the max time for a robot to reach its each node is the total time it takes to complete all tasks for each robot. The time and operating costs of the parking lot must be considered when we set up the scheduling model for the effective operation of the parking lot. Therefore, we formulate linear function (4), which minimizes the weighted sum of operating and time cost as the task scheduling objective function. λ_1 and λ_2 are weighting factors that can be set by the parking lot operator according to the operation requirement of the parking lot. For example, if a parking lot has many customers but few parking robots, the operator can increase λ_2 to improve the efficiency of the parking lot, otherwise λ_1 can be increased to reduce operating costs.

Function (5) forces that all the tasks should be finished; function (6) forces that each task should be accomplished once by a single parking robot. Constraints (7) and (8) are to restrict the initial and final positions of parking robots. Every parking robot should start from its initial node and travel to its final node after accomplishing all the tasks. Equation (9) forces that each robot should travel to the set-down node of the vehicle that it carries currently after leaving the vehicle's pick-up node; constraint (10) ensures that every robot must travel to the next vehicle's pick-up node or its final node after it fulfills carrying the current vehicle. Functions (11)–(16) are the set of constraints that restrict time windows. Constraint (11) ensures that the robot's arrival time of the node is equal to the sum of the departure time of the last node and the robot's travel time between two nodes. Constraints (12) and (13) force that the robot's leaving time of the current node is equal to the sum of the arrival time of the node and the robot's working time (the time of picking up a single vehicle or time of setting down the only vehicle). Equations (14) and (15) restrict the time windows of the robots' initial and final nodes, the robot's arrival time and departure time of its initial node are equal to 0, and the time of the robot leaving its final node is equal to its arrival time. The last function (16) ensures that the robot's arrival time of a task's pick-up node is late for departure time of its high priority task's pick-up node, that is to say, high priority task will be fulfilled first.

III. GA-BASED TASK SCHEDULING ALGORITHM

The task scheduling algorithm is the key to ensuring efficient operation of an HDP lot. In this section, based on the traditional genetic algorithm, we adopt a two-part chromosome representation and propose a new crossover method, a chromosome calibration approach, and a two-mutation method to improve the genetic algorithm.

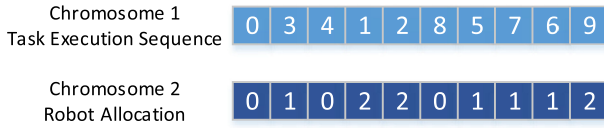


Fig. 4. Example of two-part chromosome representation for a ten-task schedule with three parking robots. The arrangement order of these numbers in the boxes of the chromosome 1 represents the task execution sequence. And the numbers 0–2 below the corresponding positions in the boxes of the chromosome 2 indicate the numbers of the robots assigned to the corresponding tasks.

A. Application of Genetic Algorithm in Parking Scheduling

1) *Chromosome Coding Method*: The chromosome coding method is the basis to implement the genetic algorithm. In this problem, the scheduling strategy needs to decide the task execution sequence and the parking robot allocation concurrently. Therefore, a two-part chromosome coding method is adopted to represent the scheduling strategy. The first chromosome represents the execution sequence of each task, whereas the second chromosome represents each task is allocated to which parking robot.

Fig. 4 shows the schematic of chromosome coding. There are ten tasks to be accomplished by three robots. The task sequence of the robot 1 is in turn tasks 3, 5, 7, and 6.

Compared with other chromosome coding methods about the AGV scheduling problem [19]–[21], our method costs some more memory, but the memory consumption is negligible.

However, because our chromosome coding method intuitively and appropriately expresses the correspondence between robots and tasks, the mutation and crossover process effectively inherits the superior traits of parents, which makes the algorithm obtain higher iteration efficiency.

2) *Cost and Fitness Function Calculation*: The cost function mainly considers two indicators of HDP scheduling problems: efficiency and economy. The efficiency cost is equal to the time consumption for all tasks to be completed. While the economy cost is equal to the sum of operating time for each parking robot, which represents the power consumption of parking robots.

Therefore, to get the operating time of the parking robots, we need to estimate the time window for each task in detail. The algorithm can calculate the robot's traveling time according to the robot's speed and the length of the shortest path. After obtaining the traveling time, each node's time window can be calculated. The pseudocode of time window calculation can be seen in Algorithm 1.

Through the above calculations, the efficiency cost and the economic cost can be known, and the final cost function can be obtained by a linear combination of them. The fitness function of each individual is not only related to their respective cost functions but also related to the evolution condition of the entire population. The fitness function of each individual is the difference between its cost function and the highest cost function in the population, as shown in the following equation:

$$f_i = Q_{\max} - Q_i \quad (17)$$

Algorithm 1: Time window calculation.

Input: chromosome1, chromosome2

Output: time window

```

1 for  $i$  in  $N_a$  do
2   for  $j$  in  $N_a$  do
3     Find the shortest path from node  $i$  to node  $j$ 
       using the A-STAR algorithm;
4     Calculate the travel time from node  $i$  to node  $j$ ;
5 for  $r$  in  $R$  do
6    ${}^aT_{N_i}^r \leftarrow 0$ ;
7    ${}^lT_{N_i}^r \leftarrow 0$ ;
8 for  $k = 0 \rightarrow n_j$  do
9    $i \leftarrow \text{chromosome1}[i]$ ;
10   $r \leftarrow \text{chromosome2}[i]$ ;
11   ${}^aT_{N_p(i)}^r \leftarrow$  the robot leaving time of the last node +
       traveling time from the last node to the current
       node;
12   ${}^lT_{N_p(i)}^r \leftarrow {}^aT_{N_p(i)}^r + \Delta t_p$ ;
13   ${}^aT_{N_s(i)}^r \leftarrow$  the robot leaving time of the last node +
       traveling time from the last node to the current
       node;
14   ${}^lT_{N_s(i)}^r \leftarrow {}^aT_{N_s(i)}^r + \Delta t_d$ ;
15 for  $r$  in  $R$  do
16   ${}^aT_{N_f}^r \leftarrow$  the robot leaving time of the last node +
       traveling time from the last node to the current
       node;
17   ${}^lT_{N_f}^r \leftarrow {}^aT_{N_f}^r$ ;

```

where f_i represents the fitness of the i th individual.

3) *Task Priority Calibration Based on Tabu Search*: In HDP lots, the relative priorities of tasks ought to be obeyed. A randomly generated individual or a child individual may not obey the priority rule. The algorithm would calibrate these individuals rather than directly delete them to prevent the waste of the computing resource.

The first chromosome of an individual can roughly reflect the task execution sequence, but not strictly because the parking lot has multiple robots and the time consumption of each task is different. Therefore, the strict way to judge whether an individual obeys the priority rule needs to make the use of the task time windows. By comparing the end time of the high-priority task and the start time of the low-priority time, the priority rule can be strictly checked. Moreover, the algorithm can calculate not only whether the individual obeys the priority rule but also how many pairs of relative priorities are violated, which is called morbid value (MV).

Calibrating the chromosomes is to adjust the task execution sequence. To calibrate the chromosomes in a short time and change the structure of the chromosomes as little as possible, a calibration strategy adopting Tabu search algorithm is proposed. The MV is used as the cost function of the Tabu search. The termination condition of the search algorithm is getting an

Algorithm 2: Chromosomes calibration.**Input:** chromosome1, chromosome2**Output:** chromosome1' // the chromosome having been calibrated

```

1  $TL \leftarrow \phi$  // the Tabu searching list;
2  $S_{MV}(MV_i) \leftarrow \phi$ ;
3  $m \leftarrow 1$  // the minimum of  $S_{MV}$ ;
4  $m_h \leftarrow +\infty$  //the historical minimum of  $S_{MV}$ ;
5 while True do
6   Find total exchange ways of chromosome1 and
   obtain the set of these exchange ways  $S_e(e_i)$ ;
7   Obtain the set of chromosomes  $S_c(c_i)$  according to
   the above exchange ways;
8   for  $c_i$  in  $S_c$  do
9     calculate  $c_i$ 's morbid value  $MV_i$  and add it into
     the  $S_{MV}$ ;
10  while True do
11     $m \leftarrow$  the minimum of the  $S_{MV}$ ;
12     $i \leftarrow m$ 's index in the  $S_{MV}$ ;
13    if  $e_i \in TL$  and  $m > m_h$  then
14      Remove  $c_i, e_i, m_i$  from  $S_c, S_e$  and  $S_{MV}$ ;
15     $m_h \leftarrow m$ ;
16    Add  $e_i$  into  $TL$ ;
17    break;
18  if  $m_h == 0$  then
19    chromosome1'  $\leftarrow c_i$ ;
20    break;
21   $S_{MV}(MV_i) \leftarrow \phi$ ;
22   $m \leftarrow 1$ ;
23 return chromosome1';

```

individual whose MV is equal to zero. An individual's neighborhoods include the individuals who only perform one exchange of a pair of gene positions based on the original individual. But only one gene position exchange usually cannot directly obtain an individual with zero MV, so the algorithm will perform another neighborhood search for the individual with the least MV after one exchange, until the individual is completely calibrated. To prevent the algorithm from falling into a local optimal situation, we send the locally optimal position exchange to the Tabu list to prevent repeated exchanges, and the forgetting limitation of the Tabu list is 20 exchanges. Algorithm III-A3 is the pseudocode of the task priority calibration process.

4) **Crossover and Mutation Method:** Crossover and mutation are important processes in inheriting parental good traits in genetic algorithms. The algorithm will cross with the probability of p_c and mutate with the probability of p_m . The random mutation operation is relatively simple, as long as the task is not repeated by the robot.

A typical crossover process of the first part chromosome is shown as Fig. 5. The start and end positions of the chromosome exchange are randomly determined. So the tasks "1285" are

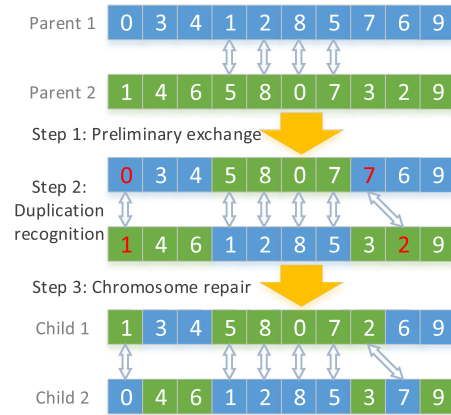


Fig. 5. Crossover method for the first part of the chromosomes with the chromosome repair operation. Step 1: Determining the gene exchange position and performing the corresponding gene exchange. Step 2: Identifying repetitive tasks in the chromosomes. Step 3: Matching and swapping repetitive tasks to repair the children chromosomes.

going to exchange with the tasks "5807." The first part chromosome represents the task execution sequence, so each task should be arranged once. But after the exchange, there will be repeated tasks 0 and 7 in child 1 and repeated tasks 1 and 2 in child 2. Then, the algorithm should perform a chromosome repair operation, swapping the duplicate tasks in child 1 and child 2. Since the second part of the chromosome represents the robot assignment, there is no nonrepetition principle, so the crossover operation can be completed by the double-points crossover operator, which directly exchanges the chromosome of the corresponding position.

B. Improved Genetic Scheduling Algorithm

Although the genetic algorithm can obtain a global good solution, the use of simple genetic algorithms (SGA) for scheduling mainly has two shortcomings, namely, falling into local optimum and prematurity. In the early stage of the genetic algorithm performing, the differences among the individuals are vast; the children of better individuals may fill the whole population quickly, resulting in the prematurity of the community. And in the late stage, one's fitness is very similar to the fitness of others, so children of the better individuals do not have the apparent advantage, which causes the evolution of the entire population to stall.

To deal with the above problems, we propose an IGA whose probability of mutation is significant. The possibility of variation is a critical parameter in the genetic algorithm. Gene mutations help the algorithm to try different strategies to approximate the optimal solution. But in SGA, the high probability of gene mutation will cause the algorithm not to converge and the result cannot be obtained. In other studies of the genetic algorithm, the value of mutation probability is selected between 0.01 and 0.1 [22]. But in this article, we choose a higher mutation probability to quickly get a new generation and solve the above problems of the genetic algorithm. To solve the problem caused by the higher mutation probability, we mainly adopt the following approaches

to improve the genetic algorithm: 1) modify the method of the fitness calculation; 2) adopt another directional mutation method based on Tabu search to form the two-mutation method.

1) *Fitness Function Calculation*: The process of genetic algorithms can usually be divided into two phases. In the earlier stages of the algorithm, the algorithm could try different types of strategies as much as possible so that the various strategies can almost completely cover the feasible domain. Therefore, the algorithm ought to have higher tolerance to suboptimal results in the earlier stage. In the later stage of the algorithm, the main purpose is to continuously optimize the existing optimal solution to approach the analytical optimal solution, so the algorithm should be very sensitive to the individual's fitness function.

Therefore, the IGA adopts a dynamic fitness function calculation method formulated as

$$f_i = (Q_{\max} - Q_i)^{\sqrt{g}} \quad (18)$$

where g is the number of algorithm's generation. The algorithm adopts the power function to improve the fitness calculation. So, the original fitness $Q_{\max} - Q_i$ can be continuously amplified as the generation time increases, and the enhancement effect is more obvious in the later stage.

2) *Directional Mutation Based on Tabu Search*: The aim of using the two-mutation method in this article is to quickly generate the new individuals to increase the diversity of population genotypes and maintain the stability of the population speeding up convergence simultaneously. In our two-mutation method, the first-mutation method executed with p_{m1} probability is a random method mentioned above. The other method executed with p_{m2} probability is formulated based on the Tabu search algorithm, which is a directional method and operates mutation toward the direction of increasing the individual's fitness.

The first mutation is used to generate more individuals whose genotypes have not appeared in the past generations, so a high probability is chosen for the first mutation operator. The second mutation operator is to maintain the stability of the population and reduce the negative impact of the high probability of the first mutation operators. But considering the calculation cost of the second mutation operator is larger than that of the first one, we choose the probability of the second mutation less than the first mutation.

The mutated neighborhood is the chromosome that selects a pair of gene positions for exchange from its parent. Different with the Tabu search in the chromosome calibration process, the objective function is not the MV but the time cost. Mutant progeny with a nonzero MV will be deleted directly because they are in the infeasible domain. To prevent the algorithm from falling into a local optimal situation, the locally optimal individuals will be placed in the Tabu list, so that the algorithm can explore other strategies and jump out of the local optimal situation. If no other better individuals are found, the algorithm will perform the pardon operation because the Tabu list has a length limit of 20.

In summary, the flowchart of the improved genetic scheduling algorithm can be obtained as Fig. 6. Note that since all children chromosomes will still be calibrated and repaired (described in

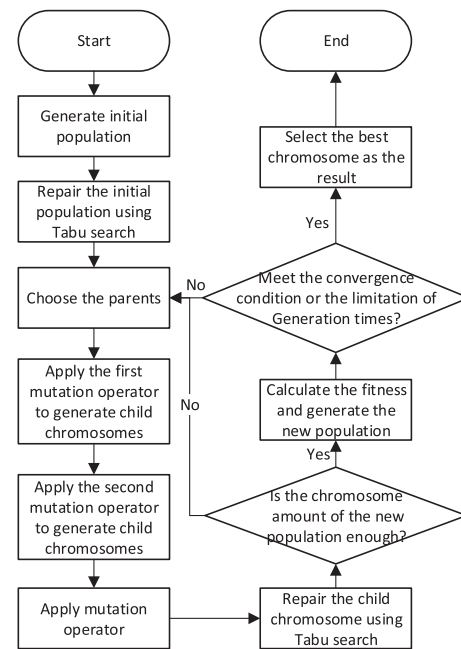


Fig. 6. GA algorithm flowchart.

Section III-A3) in IGA, the improved algorithm can ensure the priority of task execution.

IV. IMPROVED MULTIROBOT TEA* PATH PLANNING ALGORITHM

It is necessary to arrange several parking robots' paths without any collision or danger in time in a HDP lot. No any other vehicle or pedestrian will get in the HDP lot to which only the parking robots can have assessed. We adopt the centered schedule structure, which means all our path planning work will be calculated in a scheduled host. Meanwhile, the parking robots running in the parking lot will not be very fast, so it is unnecessary to considerate complex system dynamics such as inertia [23], [24].

Based on this situation, we improve the TEA* algorithm according to the heuristic function so that short and coordinated paths can be found in HDP lots. Compared with other AGV path planning algorithms that are divided into two steps (initial path planning and conflict solution) [13], [14], the method in this article can consider the above two steps under the same framework due to the time dimension of the state space considered. And the algorithm can only reserve the minimum safety space, thereby improving the performance of the planned path.

A. Time Dimension

The significant difference between the TEA* and the traditional A* is that the TEA* adds the time dimension as the third dimension. In the grid map, the start point will need a third constraint: time "t," which means when the parking robot can start to move. However, the goal point will not have a time constrain. Similar to the goal point, the static obstacles such as pillars and walls will also be modified to vertical lines.

One dynamic obstacle will only occupy one space at a fixed time. Because the accurate path histories and intentions of every parking robot are known, we can draw the dynamic obstacles' occupancy lines of them in the 3-D grid map.

When it turns to the search process, the expansion method should be modified first. As we know, the 8-connection expansion is usually adopted in the 2-D tradition A* algorithm. But in the 3-D grid map, adding a "waiting in place" strategy, the 8-connection is modified to 9-connection.

To encourage the parking robot to arrive at the goal as soon as possible, the cost function will include not only the path distance cost but also the time consumption cost. The heuristic function will not be changed and only be decided by the xy distance to the goal. Different from the distance including the distance to start and distance to goal, the time can only be evaluated according to only one benchmark, the time of the start point. Therefore, the time dimension can only be considered once. Thus, the function relation of the heuristic and cost function can be obtained as follows:

$$\begin{cases} h(s) = K_{\text{greedy}} \cdot K_{\text{dis}} \cdot d_{\text{goal}} + K_{\text{safe}} \cdot d_{\text{obst}} \\ c(s) = K_{\text{dis}} \cdot d_{\text{start}} + K_{\text{time}} \cdot t_{\text{start}} \end{cases} \quad (19)$$

B. Real-Time Challenge

The third dimension expands the grid map significantly and brings up a great challenge of the algorithm's real-time performance. The most time-consuming operation of the A* algorithm is the calculation of the heuristic function. Moreover, in our improved time-enhanced A* algorithm (ITEA*), the heuristic function includes not only the distance to the goal but also the distance to the nearest obstacle. The distance includes not only the physical distance on the plane but also the time gap. So the calculation time for the heuristic function of each node will be much longer than the one of traditional A* algorithm.

Therefore, it is a time-saving measure to set a storage map to store all the heuristic function information. It is noticed that we will not calculate all the values of the map at the beginning of the algorithm. The algorithm will still only calculate the function of the nodes in the open set. Therefore, the algorithm can directly read a wanted function that has been calculated before and stored in the storage map.

In the step of calculating the distance to the nearest static obstacles, the time dimension is redundant. So there is no time gap between a feasible note and a static obstacle. Another time-saving measure is to set a 2-D storage map for the distance to the nearest static obstacle. The nodes in 3-D grid map with the same x and y values can share one value in the 2-D distance storage map. Therefore, most redundant calculations for the static obstacles can be saved.

V. SIMULATION RESULTS

In this section, the task scheduling algorithm and the path planning algorithm are jointly simulated to realize the comprehensive scheduling management of an HDP. Our proposed IGA and ITEA* algorithms will be compared with SGA and TEA* algorithms.

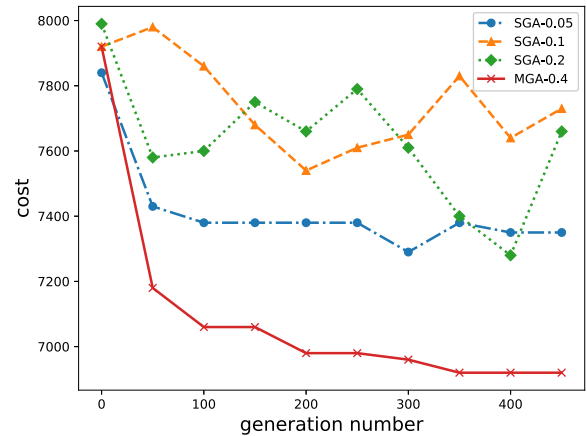


Fig. 7. Cost function convergence comparison of SGA and IGA of the "4 robots–15 tasks" problem. "SGA-0.05," "SGA-0.1," and "SGA-0.2" represent simple genetic algorithm with the mutation probabilities of 0.05, 0.1, and 0.2, respectively. "IGA-0.4" represents the improved genetic algorithm with the mutation probability of 0.4.

A. Simulation Setup

All the above algorithms are implemented in Python and all experiments are done on a PC with an Intel i7 CPU and 8 GB of RAM. The input data of the simulation are the starting point, the ending point, and the relative priority of the vehicles to be moved in the HDP lot generated by the task generating module. The output data of the simulation are the loading and unloading process and driving path of each parking robot. Through calibration, the parameters in the algorithm are set as follows: $\lambda_1 = 1$, $\lambda_2 = 1$, $p_c = 0.85$, $p_m = 0.4$, $p_{m1} = 0.35$, $p_{m2} = 0.05$, $K_{\text{greedy}} = 5$, $K_{\text{dis}} = 0.02$, $K_{\text{safe}} = 5000$, and $K_{\text{time}} = 0.05$.

B. Genetic Algorithm Converge Comparison

In order to study the performance of the task scheduling algorithm separately, we compared the convergence effects of SGA and IGA. The SGA and IGA are used to solve the "4 robots–15 tasks" problem. And each task's start and end nodes are created in advance by the task generation module. The crossover probability p_c of SGA and IGA is 0.85 and the mutation probability p_m of IGA is 0.4. Three mutation probabilities 0.05, 0.1, and 0.2 are chosen for SGA. Then we record the relationship between the cost and the generation number. The results are shown in Fig. 7.

We can see that our IGA can obtain a better task execution sequence than the SGA. And when a bigger mutation probability is chosen for the SGA, its effect becomes worse and it even cannot converge in the end. For SGA, we should choose a mutation probability in the range of 0.01 and 0.1. Due to the large mutation probability, there will be more random individuals appearing in the population, which will destroy the current better individual's chromosome structure and affect the convergence of the algorithm. In contrast, because of our improved strategy, choosing a large mutation probability not only does worsen but also improves the performance of the algorithm.

TABLE I
HIGH-DENSITY PARK SIMULATION EXPERIMENTS

R	T	Method	$D_{p,avr}(m)$		$T_{p,avr}(s)$		$D_{safe}(m)$		$T_{GA}(s)$		$T_{A^*}(s)$		$T_{calc}(s)$	
			HDP	TP	HDP	TP	HDP	TP	HDP	TP	HDP	TP	HDP	TP
2	4	SGA + TEA*	49.5	57.0	112.5	106.5	8.63	9.75	8.0	8.0	2.7	3.2	10.7	11.2
		IGA + ITEA*	57.8	57.0	105.8	105.0	10.40	10.09	8.6	8.3	4.0	3.9	12.6	12.2
2	8	SGA + TEA*	39.0	36.8	88.5	85.9	7.26	7.59	18.6	11.2	4.5	3.7	23.1	14.9
		IGA + ITEA*	34.1	38.6	82.1	81.0	7.37	8.77	17.1	12.7	5.1	5.8	22.2	18.5
2	16	SGA + TEA*	36.3	36.8	82.5	84.0	7.34	7.86	33.5	15.2	6.5	7.0	40.0	22.2
		IGA + ITEA*	32.8	33.5	77.8	79.8	9.33	8.26	25.6	18.7	8.2	9.9	33.8	28.6
5	10	SGA + TEA*	24.0	20.7	118.5	103.8	5.23	4.43	36.1	15.7	3.6	8.3	39.7	24.0
		IGA + ITEA*	23.7	20.7	112.2	107.4	5.16	4.41	27.9	17.0	3.6	12.7	31.5	29.7
5	20	SGA + TEA*	18.2	17.0	98.3	82.1	5.35	5.08	114.5	23.6	7.9	14.7	122.4	38.3
		IGA + ITEA*	16.5	17.1	86.0	87.5	5.59	5.90	57.1	42.0	7.5	24.9	64.6	66.9
5	30	SGA + TEA*	18.0	17.7	96.6	92.4	5.81	5.14	251.3	34.1	15.7	41.8	267.0	75.9
		IGA + ITEA*	17.9	18.0	92.0	92.0	6.12	5.00	199.4	93.6	44.8	64.2	164.2	157.8
8	16	SGA + TEA*	18.0	14.3	107.6	97.5	4.63	4.42	79.2	21.1	6.6	15.4	85.8	36.5
		IGA + ITEA*	12.8	14.3	87.4	83.4	3.66	3.82	45.9	32.3	18.1	15.5	64.0	47.8
8	32	SGA + TEA*	12.8	13.0	108.2	94.0	5.14	5.00	337.9	36.6	21.1	47.6	359.0	84.2
		IGA + ITEA*	12.6	12.4	89.7	97.1	4.81	4.34	164.8	129.7	66.6	81.4	231.4	211.1
8	48	SGA + TEA*	11.1	13.9	94.8	92.3	5.94	4.69	728.3	49.8	31.4	96.0	759.7	145.8
		IGA + ITEA*	10.9	11.9	80.2	92.5	4.17	5.36	282.6	107.5	140	440.1	422.6	
Avr		SGA + TEA*	25.2	25.2	100.8	93.2	6.15	6.00	178.6	23.9	11.1	26.4	189.7	50.3
		IGA + ITEA*	24.3	24.8	90.3	91.7	6.29	6.22	88.8	70.8	29.5	39.8	118.3	110.6

* “R” and “T” separately represent the number of robots and tasks. “HDP” and “TP” separately represent the simulated high-density parking lot and traditional parking lot. “SGA+TEA*” represents the simulation scheduling algorithm that is the simple genetic algorithm and the path planning algorithm is the time-enhanced A* algorithm, whereas “IGA+ITEA*” represents the algorithm that is the improved genetic algorithm and improved time-enhanced A* algorithm. Bold indicates better performance between “SGA + TEA*” and “IGA + ITEA*”, using the same number of robots and tasks, and the same method. “Avr” represents the average performance with different numbers of robots and tasks.

C. Comprehensive Comparison

Next, we can perform a comprehensive simulation of task scheduling and path planning.

1) *Evaluation Index*: Since the objective function of the genetic algorithm is only a rough estimate of the future state, after combining the path planning algorithm, we propose some more accurate evaluation indicators that are consistent with the characteristics of the path planning.

Total Travel Distance of Parking Robots D_p : The algorithm can calculate the exact path length of each robot. By accumulating them, the total travel distance of the parking robot can be obtained. Meanwhile, the average travel distance per task $D_{p,avr}$ can be calculated.

Total Task Execution Time T_p : The total task execution time can be conveniently obtained according to the time value of the last point in the path. Similarly, the average task execution time per task $T_{p,avr}$ can be calculated.

Average Safety Distance Between Robots D_{safe} : At each moment, each robot checks its distance from the nearest obstacle. This evaluation index is the average of above distances, reflecting the safety degree of robots while traveling.

Algorithm Calculation Time T_{calc} : In order to apply the algorithm to realistic HDP lots, the calculation time of the algorithm needs to be considered. The algorithm calculation time consists of GA scheduling algorithm calculation time T_{GA} and A* path planning algorithm calculation time T_{A^*} .

2) *Simulation Results*: This section compares the performance differences between our proposed algorithm and other algorithms in different parking scenarios and different task sizes. First, in an HDP lot, we generate 15 tasks that will be performed by four robots. Our proposed IGA and ITEA*

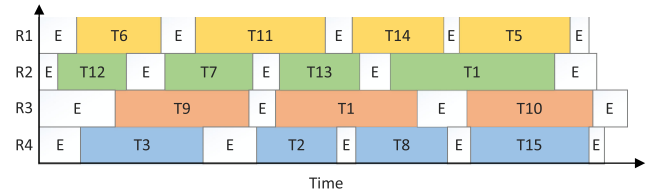


Fig. 8. Task flowchart obtained from the solution of the “4 robots–15 tasks” problem. Each line represents the task schedule of a robot. “E” represents the no-load travel between tasks.

algorithms are adopted for task scheduling and path planning. The task flowchart of a parking robot calculated by our algorithm is shown in Fig. 8. It can be found that after optimization of the scheduling algorithm, the tasks of each robot are approximately equal, and no-load travels between tasks are short. Next, we performed experiments on the number of different robots and tasks in an HDP lot scenario. The number of robot tests was selected as 2, 5, and 8, and the average number of tasks for each robot was selected as 2, 4, and 6. So, we have a total of nine combinations, and finally we will average them. Because our algorithm can also deal with the scheduling problems of TP lots, we conducted similar experiments on TP lots, as shown in Table I.

3) *Results Analysis*: According to the above experimental results, After averaging the nine HDP scenarios, the four indexes of distance, execution time, security and calculation time were optimized by 3.6%, 10.4%, 2.3%, and 37.6%, respectively. The improved algorithms have achieved steady advantages in terms of both execution time and distance. In terms of security, although the safety distance of the improved algorithm is not

significantly improved, it is still sufficient to ensure that the robot does not collide. When the tasks become complicated, the computation time of the traditional algorithm increases sharply, and the computation time of the proposed improved algorithm grows steadily due to the directed mutation strategy in the genetic algorithm.

In TP lots, the four indexes are optimized on average 1.6%, 1.6%, 3.7%, and -120% , respectively. The advantages of planned path performance are not as significant as the ones in HDP, and even a significant increase in computing time has occurred. However, compared with traditional algorithm, our algorithm can also find a better path. The reason why our algorithm has no advantage in calculation time is that there is no LIFO principle and strict priority relationship in the TP lots. In summary, our improved algorithm is more suitable for large multirobot multitasking HDP lots to reduce execution time and distance costs.

VI. CONCLUSION

This article studies the scheduling problem of parking robotics in high-density automated parking lot scenarios, including tasks scheduling algorithm and multirobot path planning algorithm. For the execution sequence problem of parking robot tasks, the genetic algorithm is used to find an optimal execution sequence. Tabu search is also adopted to improve the real-time performance of our algorithm. The parking robot allocation problem is also solved in the genetic algorithm. Besides, we improve the TEA* algorithm to solve the multi-parking-robot path planning problem. Due to the centralized multirobot path planning in automated unmanned parking lots, the goals and paths of all vehicles are known, and the TEA* algorithm is easy to implement and achieve comprehensively good performance. The simulation results show that the proposed scheduling algorithm gets better performance in high-density unmanned parking lot.

REFERENCES

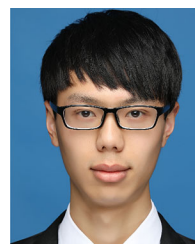
- [1] X. T. Kong, S. X. Xu, M. Cheng, and G. Q. Huang, "IoT-enabled parking space sharing and allocation mechanisms," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1654–1664, Oct. 2018.
- [2] G. Serpen and C. Dou, "Automated robotic parking systems: Real-time, concurrent and multi-robot path planning in dynamic environments," *Appl. Intell.*, vol. 42, pp. 231–251, 2015.
- [3] G. Chen, H. Cao, J. Conradt, H. Tang, F. Röhrbein, and A. Knoll, "Event-based neuromorphic vision for autonomous driving: A paradigm shift for bio-inspired visual sensing and perception," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 34–49, Jul. 2020.
- [4] G. Chen *et al.*, "Pseudo-image and sparse points: Vehicle detection with 2D LiDAR revisited by deep learning-based methods," *IEEE Trans. Intell. Transp. Syst.*, early access, Jul. 29, 2020, doi: 10.1109/TITS.2020.3007631.
- [5] M. Nourinejad, S. Bahrami, and M. J. Roorda, "Designing parking facilities for autonomous vehicles," *Transp. Res. B, Methodol.*, vol. 109, pp. 110–127, 2018.
- [6] G. Wu *et al.*, "Optimal design and planning for compact automated parking systems," *Eur. J. Oper. Res.*, vol. 273, pp. 948–967, 2019.
- [7] P. M. d'Orey, J. Azevedo, and M. Ferreira, "Exploring the solution space of self-automated parking lots: An empirical evaluation of vehicle control strategies," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 2016, pp. 1134–1140.
- [8] J. Mohammadi, K. Mirzaie, and V. Derhami, "Parallel genetic algorithm based on GPU for solving quadratic assignment problem," in *Proc. Int. Conf. Knowl.-Based Eng. Innov.*, 2015, pp. 569–572.

- [9] M. Cherklesly, G. Desaulniers, S. Irnich, and G. Laporte, "Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks," *Eur. J. Oper. Res.*, vol. 250, pp. 782–793, 2016.
- [10] B. Hu and S. Mishra, "Time-optimal trajectory generation for landing a quadrotor onto a moving platform," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 2, pp. 585–596, Apr. 2019.
- [11] Y.-D. Hong and B. Lee, "Real-time feasible footstep planning for bipedal robots in three-dimensional environments using particle swarm optimization," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 1, pp. 429–437, Feb. 2020.
- [12] V. Dwaracherla, S. Thakar, L. Vachhani, A. Gupta, A. Yadav, and S. Modi, "Motion planning for point-to-point navigation of spherical robot using position feedback," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 5, pp. 2416–2426, Oct. 2019.
- [13] Z. Zhang, Q. Guo, J. Chen, and P. Yuan, "Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification," *IEEE Access*, vol. 6, pp. 26022–26035, 2018.
- [14] I. Draganjac, D. Miklič, Z. Kovačić, G. Vasiljević, and S. Bogdan, "Decentralized control of multi-AGV systems in autonomous warehousing applications," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1433–1447, Oct. 2016.
- [15] C. Bentes and O. Saotome, "Dynamic swarm formation with potential fields and A* path planning in 3D environment," in *Proc. Brazilian Robot. Symp. Latin Amer. Robot. Symp.*, 2012, pp. 74–78.
- [16] J. Santos, P. Costa, L. F. Rocha, A. P. Moreira, and G. Veiga, "Time enhanced A*: Towards the development of a new approach for multi-robot coordination," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2015, pp. 3314–3319.
- [17] J. Azevedo, M. Macedo, P. M. d'Orey, and M. Ferreira, "High-density parking system enabled by vehicular networks," in *Proc. IEEE Veh. Netw. Conf.*, 2017, pp. 115–116.
- [18] H. Banzhaf, D. Nienhüser, S. Knoop, and J. M. Zöllner, "The future of parking: A survey on automated valet parking with an outlook on high density parking," in *Proc. IEEE Int. Conf. Intell. Veh. Symp.*, 2017, pp. 1827–1834.
- [19] X. Lyu, Y. Song, C. He, Q. Lei, and W. Guo, "Approach to integrated scheduling problems considering optimal number of automated guided vehicles and conflict-free routing in flexible manufacturing systems," *IEEE Access*, vol. 7, pp. 74909–74924, 2019.
- [20] C. Yu, Q. Semeraro, and A. Matta, "A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility," *Comput. Oper. Res.*, vol. 100, pp. 211–229, 2018.
- [21] N. Kundakcı and O. Kulak, "Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem," *Comput. Ind. Eng.*, vol. 96, pp. 31–51, 2016.
- [22] X. Wang and L. Cao, *Genetic Algorithm—Theory, Application and Software Implementation*. Xi'an, China: Xi'an Jiaotong Univ. Press, 2002.
- [23] Z. Li *et al.*, "Hybrid brain/muscle signals powered wearable walking exoskeleton enhancing motor ability in climbing stairs activity," *IEEE Trans. Med. Robot. Bionics*, vol. 1, no. 4, pp. 218–227, Nov. 2019.
- [24] Z. Li, J. Li, S. Zhao, Y. Yuan, Y. Kang, and C. P. Chen, "Adaptive neural control of a kinematically redundant exoskeleton robot using brain-machine interfaces," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3558–3571, Dec. 2019.

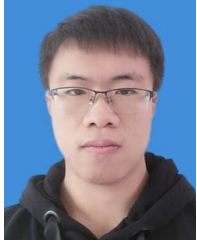


Guang Chen (Member, IEEE) received the Ph.D. degree in computer science from the Faculty of Informatics, Technical University of Munich, Munich, Germany, in 2017.

He is currently a Research Professor with Tongji University, Shanghai, China. He is leading the Intelligent Perception System Group, Tongji University. His research interests include computer vision, image processing, and machine learning.



Jing Hou received the B.E. degree in vehicle engineering from Hunan University, Changsha, China, in 2019. He is currently working toward the Ph.D. degree in vehicle engineering with Tongji University, Shanghai, China.



Jinhu Dong is currently working toward the B.E. degree in vehicle engineering with Tongji University, Shanghai, China.

His research interests include autonomous driving, computer vision, and SLAM.



Bo Zhang received the Ph.D. degree in precision instrument and mechanism from the Instrument Science and Engineering Department, Shanghai Jiao Tong University, Shanghai, China, in 2007.

He is currently the Head of the Autonomous Driving Team, Shanghai Westwell Information and Technology Company Ltd., Shanghai, China. His research interests include autonomous driving, computer vision, SLAM, and sensor fusion.



Zhijun Li (Senior Member, IEEE) received the Ph.D. degree in mechatronics from Shanghai Jiao Tong University, Shanghai, China, in 2002.

Since 2017, he has been a Professor with the University of Science and Technology, Hefei, China. His research interests include wearable robotics, teleoperation systems, nonlinear control, and neural network optimization.

Dr. Li has been the Co-Chair of the IEEE SMC TC (B²S) and the IEEE-RAS TC Neurorobotics Systems. He is currently an Associate Editor for

several IEEE transactions.



Junwei Yu received the B.E. degree in automation engineering from Tongji University, Shanghai, China, in 2001 and master's degree in automation from TU Darmstadt, Darmstadt, Germany, in 2008. He is currently working toward the Ph.D. degree in vehicle engineering with Tongji University.



Shangding Gu received the B.S. degree in transportation engineering from Hefei University of Technology, Hefei, China, in 2017. He is currently working toward the M.S. degree in traffic information engineering and control with Wuhan University of Technology, Wuhan, China.



Alois Knoll (Senior Member, IEEE) received the Diploma (M.Sc.) degree in electrical/communications engineering from the University of Stuttgart, Stuttgart, Germany, in 1985, and the Ph.D. (*summa cum laude*) degree in computer science from the Technical University of Berlin, Berlin, Germany, in 1988.

Since 2001, he has been a Professor with the Department of Informatics, TU München, Munich, Germany.