Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 6153848, 15 pages
https://doi.org/10.1155/2019/6153848

Research Article

A Building-Block-Based Genetic Algorithm for Solving the Robots Allocation Problem in a Robotic Mobile Fulfilment System

Jingtian Zhang, Fuxing Yang, and Xun Weng

Automation School, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Jingtian Zhang; buptzjt@163.com

Received 18 September 2018; Revised 8 December 2018; Accepted 12 February 2019; Published 25 February 2019

Academic Editor: Francesco Zammori

Robotic mobile fulfilment system (RMFS) is an efficient and flexible order picking system where robots ship the movable shelves with items to the picking stations. This innovative parts-to-picker system, known as Kiva system, is especially suited for e-commerce fulfilment centres and has been widely used in practice. However, there are lots of resource allocation problems in RMFS. The robots allocation problem of deciding which robot will be allocated to a delivery task has a significant impact on the productivity of the whole system. We model this problem as a resource-constrained project scheduling problem with transfer times (RCPSPTT) based on the accurate analysis of driving and delivering behaviour of robots. A dedicated serial schedule generation scheme and a genetic algorithm using building-blocks-based crossover (BBX) operator are proposed to solve this problem. The designed algorithm can be combined into a dynamic scheduling structure or used as the basis of calculation for other allocation problems. Experiment instances are generated based on the characteristics of RMFS, and the computation results show that the proposed algorithm outperforms the traditional rule-based scheduling method. The BBX operator is rapid and efficient which performs better than several classic and competitive crossover operators.

1. Introduction

With the development of e-commerce, it is crucial to improve the velocity and efficiency of operational processes in warehouses and distribution centres. A robotic mobile fulfilment system (RMFS) is a robot-based automated storage and parts-to-picker order picking system which is suited for distribution centres that face strong demand fluctuations and large assortments of small products [1]. Pioneered by famous Kiva systems, the RMFS is increasingly used in the e-commerce fulfilment centres to increase throughput capacity, reduce operational cost, and improve flexibility [2].

The resource allocation problems in RMFS are some intractable, dynamic, and stochastic optimization problems with incomplete information. The resources include things like inventory, open orders, movable shelves, and robots. The global optimization problem can be broken down into more approachable subproblems, such as inventory shelves selection problem, shelves storage allocation problem, order allocation problem, replenishment allocation problem, and robots allocation problem, some of which could be mapped to versions of well-known computational problems with modifications of altered workflow in RMFS [3]. Currently, there are few studies on RMFS. As pioneer researchers that study RMFS, Wurman et al. [4] formulated a multiagent architecture consisting of a drive unit agent, an inventory station agent, and a job manager to control the Kiva system. Instead of attempting global optimization, the resource allocation problems are solving on the fly using utility-based heuristics. Enright and Wurman [3] described the resource allocation problems in RMFS with enough detail to encourage other researchers to study. Among the recent researchers, Lamballais et al. [1] developed a semiopen queuing network (SOQN) to estimate the performance of an RMFS. It is one of the first papers to model RMFS with accurate driving behaviour of robots and multiline orders. Yuan and Gong [5] built queue network models to describe the RMFS with two protocols in sharing robots for pickers

and calculated the optimal number and velocity of robots to minimize the total throughput time. Boysen et al. [6] formalized the order processing problems in a single picking station which consists of batching and sequencing of picking orders and the interdependent sequencing of the shelves brought to a station. A decomposition procedure using a simulated annealing scheme and beam search mechanism was provided to solve the resulting decision problem. Zou et al. [2] proposed a handling-speeds-based assignment rule and designed a neighbourhood search approach to find a near optimal assignment rule of workstations to robots to improve the system performance. They also build a semiopen queuing network to estimate the performance of the RMFS and used a two-phase approximation method to solve the SOQN. Zou et al. [7] investigated plug-in battery charging, inductive battery charging, and battery swapping strategies in the RMFS regarding system throughput time and cost and built a SOQN to estimate the performance of the RMFS with battery recovery consideration. Most of the studies in RMFS mainly focus on performance estimation; some of them tackled the inventory shelves selection problems and order allocation problems. However, the dedicated robot allocation methods in the picking process for RMFS have yet not been considered in the scientific literature.

The robots allocation problems in RMFS decide which robot will be allocated to a movable shelve that needs to be delivered, which has affected the robots' travel distance and the makespan of the system. A suitable robot allocation method will improve the efficiency of the operation process and minimize the throughput time in warehouses. When the scheduling order of shelves in each picking station has been determined, the robot allocation problems in RMFS can be transformed into the resource-constrained scheduling problem with transfer times (RCPSPTT). RCPSPTT is an extension of the resource-constrained project scheduling problem (RCPSP) which consists of scheduling a set of activities linked by zero-lag and finish-to-start precedence relations, while pre-emption is not allowed. The resources required to perform activities are renewable and available in limited quantities in each period [8]. The priority rule-based construction heuristic is widely used in RCPSP to generate feasible initial schedules. Then, the obtained schedules are improved by various neighbourhood search methods and meta-heuristic algorithm, including tabu search [9], simulated annealing [10], scatter search [11], ant colony optimization [12], particle swarm optimization [13], and genetic algorithm [14–16].

In RCPSPTT, resource transfers with sequence- and resource-dependent transfer times between jobs in the same projects or between jobs in different projects are considered. Current studies on RCPSPTT are limited. Krüger and Scholl [17] first developed a heuristic solution framework for RCPSPTT, which formulated an integer linear program and proposed priority rule-based solution procedures. Poppenborg and Knust [18] proposed a tabu search algorithm in which solutions are represented by resource flows. Kadri and Boctor [8] used a new efficient genetic algorithm with a modified magnet-based crossover operator to solve RCPSPTT.

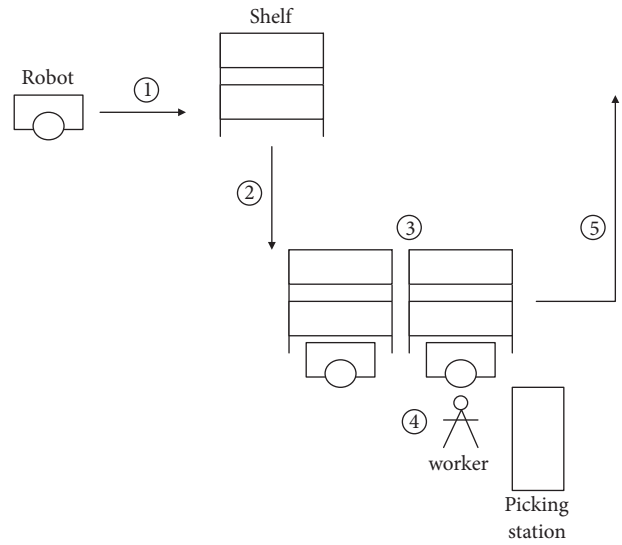

Figure 1: Five steps of a robot shipping task.

This paper mainly addresses the robots allocation problem under a determined sequence of required shelves in each picking station. Based on analyzing the accurate driving behaviour of robots and the operational process in picking station, the robot allocation problem in RMFS is modelled as RCPSPTT problem. Then, we develop a dedicated serial schedule generation scheme based on the characteristics of RMFS to solve this problem. Moreover, a new genetic algorithm using the building-block-based crossover (BBX) is proposed for further optimization. The proposed algorithm has significantly improved the efficiency of RMFS compared with traditional allocation method, and the BBX operator outperformed other competitive crossover operators in the designed experiment.

2. Problem Description and Mathematical Model

2.1. Robotic Mobile Fulfilment Systems. Robotic mobile fulfilment system is a part-to-picker picking system using movable shelves to store items and using robots to ship the shelves to the picking stations. The working process is as follows: when some picking orders enter the system, they are assigned to the picking stations according to assignment rules or some optimization methods [6]. Then, the sequence of shelves with required items for each picking station is determined, and the system will assign robots to ship these shelves to the target stations. As shown in Figure 1, a robot's delivery task involves five steps [3]:

(1) Driving from the robot's current location to the target shelf's storage location.

(2) Carrying the shelf from its storage location to the picking station.

(3) Queuing at the buffer of the station until it is the shelf's turn.

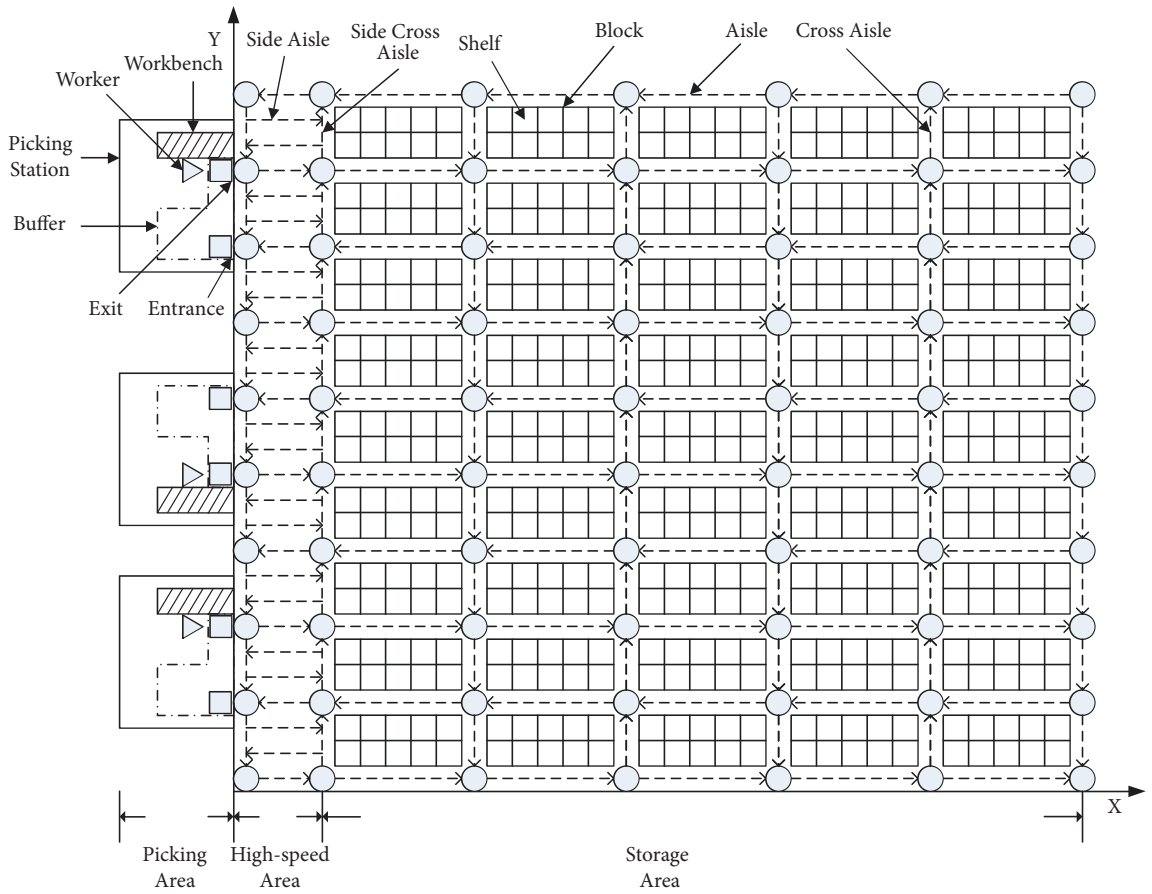

Figure 2: Top view of an RMFS with three picking stations on the left side.

(4) Letting the worker pick products from the shelf.

(5) Bringing the shelf back to its storage location.

A top view of a typical layout with RMFS is shown in Figure 2. We divide the system layout into three areas: storage area, high-speed area, and picking area. A shelf is the smallest square in the storage area and stored in rectangular blocks. Aisles and cross aisles in the storage area are marked by dashed lines with directional arrows and arranged along the X-axis and Y-axis, respectively. The circles between various aisles are the intersections. An idle robot can move underneath the shelves, but when the robot is carrying a shelf, it can only drive along an aisle or a cross-aisle following the aisles' travel direction. Both aisles and cross aisles are all unidirectional. The high-speed area is between the picking area and storage area, in which robots can drive at high speed. In picking area, robots carrying shelves with required items enter the picking station from the entrance and wait in the buffer for their turn. The worker picks items from the shelf in the exit of picking area and adds them to the order totes on the workbench. The robot that completes the picking task should leave the picking station from its exit, drives back to the storage area, and ships the shelf to its storage location. Since the robots are travelling along the unidirectional aisles, dedicated calculation method is needed to obtain the travel time. The detailed calculation process of travel time is listed in the Appendix.

2.2. Model of the Robots Allocation Problem. The robots allocation problem can be described as follows: given the sequence of the delivery tasks for each picking station and each task requiring a specified shelf, we should decide which robots will be allocated to ship these shelves while the sequence that the shelves arrive at each picking station need to be strictly satisfied. As an illustration in Figure 3, there are two picking stations, and each station has three delivery tasks. A delivery task can be performed only after its predecessor task has been shipped to the station. There are 150 shelves in the storage area, and each task requires a specified shelf. It is worthy of note that different tasks can require the same shelf. When performing a delivery task, a robot is selected to ship the required shelf to the station and bring it back to the storage location after its picking process has been done. Supposing two robots are used to perform these tasks, we need to determine the sequence of the delivery tasks for each robot. For example, robot 1 has been assigned three tasks: 2, 4, and 5. The robot firstly travels to the storage position of shelf 42, ships it to station 1, and ships it back to the storage area. Then robot 1 travels to shelf 30 and ships it to station 2 and travels to shelf 25 after shipping shelf 30 back to its storage position.

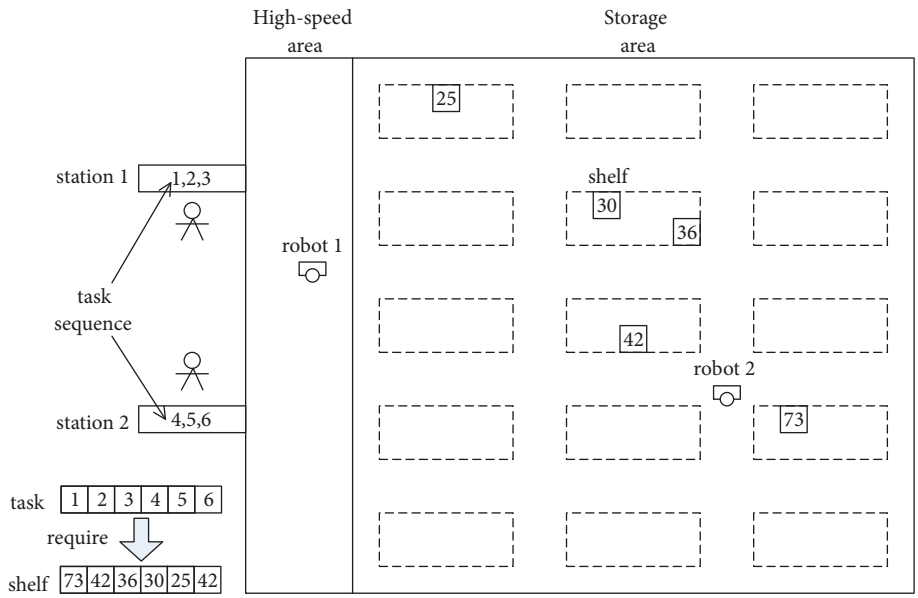

Figure 3: Robots allocation problem.

As discussed above, the robot allocation problem in RMFS can be transformed into the resource-constrained scheduling problem with transfer times (RCPSPTT). A project includes all the delivery tasks in a picking station, and the tasks set of each station is a set of activities linked by zero-lag and finish-to-start precedence relations. The robots, the station buffers, and the shelves are the resources which are used by activities and are renewable after the period of the activities' execution. Shelf becomes a resource because the same shelf may be shipped multiple times in single or multi picking stations. The resource "robots" need a transfer time between activities which is the travel time of robots in step1, and the transfer time is variable depending on which robot is selected. The objective of the robot allocation is to find the sequence of shipping tasks for each robot to minimize the makespan of the system while satisfying the precedence constraints between shelves and the capacity constraints of the resources. We transform the resources constraint into time constraints; the mathematical model of robot allocation problem in RMFS is as follows.

There is a set of L picking stations $P = 1, 2, \ldots, L$ with B_p capacity of a buffer in the system. There is a set of M delivery tasks $J = 1, 2, \ldots, M$ and a set of Q shelves $K = 1, 2, \ldots, Q$. Each delivery task $j \in J$ requires shipping a specific shelf $k_j \in K$ to its target picking station and back to its storage location after picking process. A fleet of N robots $R = 1, 2, \ldots, N$ is responsible for performing these tasks. Each picking station $p \in P$ has a given set of M_p delivery tasks $S_p = s_{p,1}, s_{p,2}, \ldots, s_{p,M_p}$, and all these tasks should be assigned to the fleet of robots. Supposing robot $r \in R$ has been assigned N_r tasks, the set of tasks for robot r is $S^r = s_0^r, s_1^r, s_2^r, \ldots, s_{N_r}^r$, where s_0^r is the starting position of robot r before performing the delivery tasks.

$$J = \bigcup_{p=1}^{L} S_p = \bigcup_{r=1}^{N} \left(\frac{S^r}{\{s_0^r\}} \right) \tag{1}$$

Equation (1) indicates that all the tasks of the picking stations are assigned to the fleet of robots. Supposing task $s_{p,m}$ ($m \in M_p$) of station p will be performed by robot r in the ith order, let $\mathrm{FT}_{s_i^r}$ be the finishing time that robot r completes task $s_i^r = s_{p,m}$, and it can be calculated as

$$\mathrm{AT}_{s_i^r} = \mathrm{ST}_{s_i^r} + T_2^{s_i^r} \tag{2}$$

$$\mathrm{LT}_{s_i^r} = \max\left\{ \mathrm{LT}_{s_{p,m-1}}, \mathrm{AT}_{s_i^r} + T_3^{s_i^r} \right\} + T_4^{s_i^r} \quad \left(s_i^r = s_{p,m}; s_{p,m-1}, s_{p,m} \in S_p \right) \tag{3}$$

$$\mathrm{FT}_{s_i^r} = \mathrm{LT}_{s_i^r} + T_5^{s_i^r} \tag{4}$$

$\mathrm{ST}_{s_i^r}$ represents the starting time to do step 2-5 of task s_i^r, $\mathrm{AT}_{s_i^r}$ represents the time that robot r performing task s_i^r arrives at the picking station, and $\mathrm{LT}_{s_i^r}$ represents the time that robot r performing task s_i^r leaves the picking station (i.e., the time that task s_i^r completes the picking process in the station). Equation (3) indicates that task $s_i^r = s_{p,m}$ can be handled in the station only after its direct predecessor task $s_{p,m-1}$ in the tasks set of the station has completed the picking process. $T_2^{s_i^r}$ is the travel time of task s_i^r in step 2, $T_3^{s_i^r}$ is the travel time of task s_i^r in step 3 (i.e., travel through the buffer of the station), $T_4^{s_i^r}$ is the picking time of task s_i^r in step 4, and $T_5^{s_i^r}$ is the travel time of task s_i^r in step 5. $T_4^{s_i^r}$ is constant which is determined beforehand, and $T_2^{s_i^r}$, $T_3^{s_i^r}$, and $T_5^{s_i^r}$ are constant when the

storage locations of the shelf and the buffer of the station are fixed.

Let s_e^r be the last task in the tasks set of robot r, and the model is given as

Decision Variables

$$z_{ij}^r = \begin{cases} 1 & \text{if robot } r \text{ moves to task } j \text{ after deliveried task } i \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij}^s = \begin{cases} 1 & \text{if shelf } s \text{ is used in task } j \text{ after used in task } i \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Minimize

$$\max_{r \in R} \quad \mathrm{FT}_{s_e^r} \quad (6)$$

Subject to

$$\Delta_{i,j} = z_{ij}^r \cdot T_1^j \quad (\forall i, j \in J) \quad (7)$$

$$\mathrm{ST}_j + T_2^j > \mathrm{AT}_{j-1} \quad (p \in P;\ j-1,\ j \in S_p) \quad (8)$$

$$\mathrm{ST}_j + T_2^j > \mathrm{LT}_{j-B_p} \quad (p \in P;\ j-B_p,\ j \in S_p) \quad (9)$$

$$\mathrm{ST}_j > \mathrm{FT}_i + \Delta_{i,j} + T\left(1 - z_{ij}^r\right) \quad (\forall i, j \in J) \quad (10)$$

$$\mathrm{ST}_j > \mathrm{FT}_i + T\left(1 - z_{ij}^s\right) \quad (\forall i, j \in J) \quad (11)$$

$$\sum_{r \in R} \sum_{i \in S^r} z_{ij}^r = 1 \quad (i \neq j,\ \forall j \in J) \quad (12)$$

$$\sum_{r \in R} \sum_{j \in S^r} z_{ij}^r = 1 \quad (i \neq j,\ \forall i \in J) \quad (13)$$

$$z_{ij}^r \in \{0, 1\} \quad (r \in R;\ i, j \in J) \quad (14)$$

$$z_{ij}^s \in \{0, 1\} \quad (i, j \in J) \quad (15)$$

where T_1^j is the travel time of task j in step 1 and $\Delta_{i,j}$ represents the transfer time of resource "robot" r between tasks i and j, which means the travel time for r to drive from the storage location of task i to the storage location of task j. T is a large positive quantity.

The objective function (6) minimizes the makespan of all the robots, which indicates that the finishing time of the last task is the makespan of the system. Equation (7) shows that the transfer time of resource "robots" between task i and task j is the travel time of task j in step 1, and T_1^j is variable depending on task i. Constraints (8)-(11) are the constraints on the starting time of the task so that the finishing time of the task can be calculated according to (2)-(4) after its starting time has been determined. Constraint (8) is the precedence constraint of tasks in the same picking station, and it means that task j can arrive at the picking station only after its direct predecessor task j-1 arrives. Constraint (9) is the constraint on resource "station buffers", and it means that task j can arrive at the picking station only after the buffer of the station has a free location (i.e., its indirect predecessor task $j-B_p$ leaves the station.) Constraint (10) is the constraint on resource "robots" with transfer time, and it means that task j can start only after the robot finished task i and travelled to the location of task j. Constraint (11) is the constraint on resource "shelves", and it means that task j can start only after task i (which used the same shelf with task j) is finished. Constraints (12) and (13) ensure that each task can only be delivered by one robot exactly once. Constraints (14) and (15) define the decision variables of the problem.

3. The Proposed Algorithm

When RMFS receives some picking orders, the order allocation problem should be tackled first, i.e., determining the batching and sequencing of picking orders and the interdependent sequencing of the shelves that will be shipped to all the picking stations [6]. Then the robots allocation problem, which is the issue of this paper, should be tackled, i.e., determining the scheduling and the sequencing of the shelves that each robot need to ship. Since the robots allocation problem in RMFS is modelled as RCPSPTT, several classical algorithms used in resource-constrained project scheduling problem can be considered for solving this problem. However, the schedule generation scheme should be especially designed according to the operational process of RMFS and the resource transfer time between tasks. Moreover, for the actual implementation of RMFS in the warehouse, the algorithms require having rapid convergence capability and can be applied in large-scale instances.

To solve this problem, we propose a priority rule-based constructive heuristics algorithm which is controlled by a schedule generation scheme and a priority rule. Considering the resource transfer time, the schedule generation scheme is serial and designed according to the characteristics of the RMFS. The presentation of solutions in the algorithm is an activity list, and it is the input to the schedule generation scheme to generate a schedule and obtain the makespan of the system. In particular, we design an additional code called "auxiliary code" to distinguish similar solutions and increase the diversity of the population. For further optimization, we propose an improved genetic algorithm (GA) with a new designed building-block based crossover (BBX) operator, which acts on the activity lists of solutions to generate new solutions. By finding the key permutation of the activity lists between solutions which is also called the building block, the BBX-GA balances the disruptiveness and the inheritance.

The algorithm starts with an initial population of D (the upper bound of the solutions pool) solutions generated by the designed serial schedule generation scheme with the random activity selection rule. The random activity selection rule is a priority rule, used by Krüger and Scholl [17] as the job selection rule "Random." It randomly selects one activity which

Generate D initial solutions using the serial scheduling generation scheme with the random activity selection rule
Select the best F ($F \leq D$) solutions with different auxiliary code into solutions pool
WHILE ($t < ET$)
  Select F ($F \leq D$) pairs of solutions using the k-tournament method
  Apply building-block based crossover and generate F ($F \leq D$) new solutions
  Use the serial schedule generation scheme to generate schedules
  Apply random-extract mutation to the new solutions and generate another F ($F \leq D$) new solutions
  Use the serial schedule generation scheme to generate schedules
  Add all the solutions together and select the best F ($F \leq D$) solutions with different auxiliary code into solutions pool
END WHILE
Output the best solution

Algorithm 1: The framework of the proposed algorithm.

satisfied the precedence and resource constraints at each decision stage to generate an activity list. When D solutions are generated, we use auxiliary codes to distinguish them; i.e., the solutions in the population should have different auxiliary codes with each other. If some solutions have the same auxiliary code, we only reserve the solution with the best makespan. So, the number of solutions in the solutions pool F may be less than D. Then, the BBX-GA is operated on the solutions pool. It first selects F pairs of solutions to do the crossover and generates new F solutions. Each solution in the solutions pool will be the father solution once, and the mother solution is selected using the k-tournament method. Use these new solutions (activity lists) as the input to the serial schedule generation scheme to generate schedules and obtain the makespan of each solution. Then, the mutation operator is applied to the new F solutions to generate new other F solutions. Again, these new solutions are also used as input to the serial schedule generation scheme. After obtaining the makespan of each of the new $2F$ solutions, we add all of them to the solutions pool. Then, we update the solutions pool by only reserving the best F solutions with different auxiliary code for the next generation. The BBX-GA iteratively runs ET times, and the best solution is output when the algorithm terminates. The pseudocode of the algorithm is shown in Algorithm 1.

3.1. Solution Representation. We use the activity list as the genotype of a solution. An activity list is a precedence-feasible permutation of tasks, in which no task appears before its predecessor [19]. In our algorithm, a task is an activity. The task is scheduled one by one according to their sequence in the activity list, and then a feasible schedule of the solution is generated by the schedule generation scheme.

A major disadvantage of the activity list is that a single schedule can be represented by different activity lists, which results in a larger solution space and the structure of the solution does not necessarily contain information about the quality of the associated schedule [20]. Therefore, we designed an additional code called "auxiliary code" to represent the schedule of a solution. The auxiliary code is automatically generated when the schedule of a solution is obtained, and it indicates which task is performed by which robot. It is like a label of a solution and not involved in other operations. Using auxiliary code to distinguish solutions avoids the situation where different active lists produce the same schedule and classifies the solutions with the same allocation tasks but different sequence into one cluster, which increases the diversity of the population.

The sequence of the auxiliary code is fixed to the task number, and each bit indicates the task is performed by which robot. When the activity list of a solution is obtained, the schedule generation scheme uses this activity list as the input to generate a feasible schedule. Then the makespan and the auxiliary code of each solution can be obtained according to its schedule. As an illustration, there is a problem which contains three picking stations and four robots. Figure 4 shows two solutions with different activity lists but having the same auxiliary code. In solution1, the first bit of its auxiliary code means task 1 is assigned to robot 3 and the second bit means task 2 is assigned to robot 4, and so forth like this until the last bit. The robot 1 of solution 1 should perform task 7, task 11, and task 5 in order, and the robot 1 of solution 2 performs the same tasks but with a different order. These two solutions are similar. When the population is filled with these similar solutions, the quality of the algorithm will decline. The auxiliary codes of the two solutions are the same, so we only keep the solution that has the less makespan, which reduces the number of similar solutions in the population and increases the diversity of the population. Moreover, the auxiliary code of each solution is unique in the population. If there are some solutions with the same auxiliary code, we only keep the solution with the best makespan.

3.2. Schedule Generation Scheme. While the activity list of a solution is obtained, the schedule generation scheme is performed to generate a feasible schedule of the solution by converting all the permutations of activities to their starting times. Two schedule generation schemes, namely, serial and parallel, play key roles in almost all heuristics presented for the RCPSP [16]. The serial schemes are activity oriented which select one activity in each decision stage and schedule it as early as possible while precedence and resource feasibility are satisfied. On the contrary, the parallel schemes are time-oriented which schedule all the eligible activities at current decision time and then increase the decision times until all the activities are scheduled. These two schedule generation

CalculateSameShelf(r) /*judge whether the shelf of the latest task in robot r is the same as the new task j
Get the latest task s_l^r in robot r
IF $k_j = k_{s_l^r}$ *AND* $AK_{k_j} = \mathrm{FT}_{s_l^r}$ /* the two tasks have the same shelf and the shelf is used by j immediately after s_l^r
  IF $S_p \supseteq (s_l^r \cup j),\ \exists p \in P$ /* s_l^r is the predecessor task of j in the same station
    IF $s_l^r = j - 1$ /* s_l^r is the direct predecessor task of j in the same station
      $\mathrm{FT}_{s_l^r} \coloneqq \mathrm{FT}_{s_l^r} - T_5^{s_l^r}, \mathrm{ST}_j \coloneqq \mathrm{FT}_{s_l^r}, \mathrm{FT}_j \coloneqq \mathrm{ST}_j + T_4^j + T_5^j$, selectRobot $\coloneqq r$
      RETURN TRUE
    ELSE /* s_l^r is the indirect predecessor task of j in the same station
      Get the travel time TT that robot travel from the exit of current station to the entrance of the current station
      IF $\mathrm{FT}_{s_l^r} - T_5^{s_l^r} + TT \geq \mathrm{ST}_j + T_2^j$ /* the modified time of task j arrives at the station satisfies the constraint (8) (9)
        $\mathrm{FT}_{s_l^r} \coloneqq \mathrm{FT}_{s_l^r} - T_5^{s_l^r}, \mathrm{ST}_j \coloneqq \mathrm{FT}_{s_l^r}, \mathrm{FT}_j \coloneqq \max\{\mathrm{ST}_j + TT + T_3^j, \mathrm{FT}_{j-1} - T_5^{j-1}\} + T_4^j + T_5^j$, selectRobot $\coloneqq r$
        RETURN TRUE
  ELSE /* j and s_l^r are required by different stations
    Get the travel time TR that robot travel from the exit of the current station to the entrance of the target station
    IF j is the first task in the station
      OR $\mathrm{FT}_{s_l^r} - T_5^{s_l^r} + TR \geq \mathrm{ST}_j + T_2^j$ /* the modified time of task j arrives at the station satisfies the constraint (8) (9)
    $\mathrm{FT}_{s_l^r} \coloneqq \mathrm{FT}_{s_l^r} - T_5^{s_l^r}, \mathrm{ST}_{J_i} \coloneqq \mathrm{FT}_{s_l^r}$, selectRobot $\coloneqq r$
    $\mathrm{FT}_j \coloneqq \mathrm{ST}_j + TR + T_3^j + T_4^j + T_5^j$ (J_i is the first task) or $\mathrm{FT}_j \coloneqq \max\{\mathrm{ST}_j + TR + T_3^j, \mathrm{FT}_{j-1} - T_5^{j-1}\} + T_4^j + T_5^j$
    RETURN TRUE
ELSE
  RETURN FALSE

Pseudocode 1: Pseudocode of judgment rules and modified calculated method.

schemes have been modified to solve the RCPSPTT by Krüger and Scholl [17].

However, for the robot allocation problem in RMFS, the most critical difference from the standard RCPSPTT is the specific operational process when delivering a shelf which is required by multiple tasks. For example, if a shelf is required by a new task after its picking process of the current task is finished, the shelf can be directly shipped to the station of this new task when satisfying all constraints, instead of being shipped back to the shelf's storage location first and then the target station. The schedule generation scheme should tackle this special operational process which will greatly reduce the makespan of the system. According to the generation process of schedule, at each decision stage, we need to judge whether the latest scheduled tasks of each robot required the same shelf with the new task. When the shelf is the same, and all constraints are satisfied, the new task can be allocated to this robot, and the finishing time of the latest scheduled task in this robot should be modified according to the changed travel path. In this way, the time-oriented parallel scheme cannot be used, so we design a modified serial schedule generation scheme.

The algorithm starts with an initialization of the instance, and the activity list is as the input of schedule generation scheme. Tasks are scheduled one by one according to their sequence on the activity list. Suppose one task j, which is provided by the activity list, is scheduled at the current stage. For the selected task j, the required resource "shelf" is determined; we should choose the resource "robot" to ship the shelf to the target station. Different robots will lead to different resource transfer times. To compute the earliest transfer-feasible finishing time of task j with different robots, we designed a modified calculation method according to the different situations. We first choose one robot r and judge whether the shelf required by the latest task i of robot r is the same as task j. If they are the same, there are three situations: task i is the direct predecessor task of task j in the same picking station, task i is the indirect predecessor task of task j in the same picking station, and task i and task j are required by two different picking stations. In these situations, the robot does not need to return to the storage location of task i but instead goes directly to the station of task j if all the constraints are met. The calculations of the finishing time of task j on these situations are different. If some constraint cannot be met, the finishing time of task j is calculated in the normal way (i.e., the robot returns to the storage location of task i before performing the new task j.) The pseudocode of the modified calculation method of finishing time is shown in Pseudo-code 1.

Parameters

$\mathrm{FT}_{s_0^r}$ the available time of robot $r \in R$ at its starting position s_0^r

AK_k the available time of shelf $k \in K$

When the finishing time of task j with each robot is obtained, we need a resource transfer priority rule to determine which robot to be allocated to perform task j. The transfer priority rule "minFT" is used which selects the robot with the earliest finishing time of the task j. Then, the resources involved are allocated and updated. After all the tasks on the activity list have been assigned, a robots' allocation schedule is generated,

Initialize: $S^r \coloneqq \varnothing$ $(\forall r \in R)$; $\mathrm{ST}_j \coloneqq 0$, $\mathrm{FT}_j \coloneqq$ a large postive number $(\forall j \in J)$; $AK_k \coloneqq 0$ $(\forall k \in K)$;
S_p $(\forall p \in P)$ is predetermined; Let AL be the activity list to be transformed into a schedule
FOR $i \coloneqq 1$ to M
Let $j \in J$ be the task at position i in AL
selectRobot $\coloneqq 0$
FOR $r \coloneqq 1$ to N
IF $S^r \neq \varnothing$
IF CalculateSameShelf$(r) = TRUE$
BREAK
ELSE
Get the latest task s_l^r in robot r
$\mathrm{tempST}_j \coloneqq 0$, $\mathrm{tempFT}_j \coloneqq 0$
IF j is the first task in the station
$\mathrm{tempST}_j \coloneqq \max\{\mathrm{FT}_{s_l^r} + \Delta_{s_l^r,j}, \mathrm{ST}_j, AK_{k_j}\}$, /∗satisfy the constraints (10) (11)
$\mathrm{tempFT}_j \coloneqq \mathrm{ST}_j + T_2^j + T_3^j + T_4^j + T_5^j$
ELSE
$\mathrm{tempST}_j \coloneqq \max\{\mathrm{FT}_{s_l^r} + \Delta_{s_l^r,j}, \mathrm{ST}_j, AK_{k_j}\}$, /∗satisfy the constraints (8)-(11)
$\mathrm{tempFT}_j \coloneqq \max\{\mathrm{ST}_j + T_2^j + T_3^j, \mathrm{FT}_{j-1} - T_5^{j-1}\} + T_4^j + T_5^j$
ELSE/∗$S^r = \varnothing$
$\mathrm{tempST}_j \coloneqq 0$, $\mathrm{tempFT}_j \coloneqq 0$
IF j is the first task in the station
$\mathrm{tempST}_j \coloneqq \max\{\mathrm{FT}_{s_0^r} + \Delta_{s_0^r,j}, \mathrm{ST}_j, AK_{k_j}\}$, /∗satisfy the constraints (10) (11)
$\mathrm{tempFT}_j \coloneqq \mathrm{ST}_j + T_2^j + T_3^j + T_4^j + T_5^j$
ELSE
$\mathrm{tempST}_j \coloneqq \max\{\mathrm{FT}_{s_0^r} + \Delta_{s_0^r,j}, \mathrm{ST}_j, AK_{k_j}\}$, /∗satisfy the constraints (8)-(11)
$\mathrm{tempFT}_j \coloneqq \max\{\mathrm{ST}_j + T_2^j + T_3^j, \mathrm{FT}_{j-1} - T_5^{j-1}\} + T_4^j + T_5^j$
IF $\mathrm{tempFT}_j < \mathrm{FT}_j$
$\mathrm{ST}_j \coloneqq \mathrm{tempST}_j$, $\mathrm{FT}_j \coloneqq \mathrm{tempFT}_j$, selectRobot $\coloneqq r$
NEXT r
$S^{\mathrm{selectRobot}} \coloneqq S^{\mathrm{selectRobot}} \cup j$
$AK_{k_j} \coloneqq \mathrm{FT}_j$ /∗ update the available time of the shelf used in task j
IF $M_p - j > B_p$ $(j \in S_p;\ p \in P)$
$\mathrm{ST}_{j+B_p} \coloneqq \max\{\mathrm{FT}_j - T_5^j - T_2^{j+B_p}, \mathrm{ST}_{j+B_p}\}$ /∗ Add constraint (9) to the task $j + B_p$, and update its starting time
IF j is not the last task in the station
$\mathrm{ST}_{j+1} \coloneqq \max\{\mathrm{FT}_j - T_2^j, \mathrm{ST}_{j+1}\}$ /∗ Add constraint (8) to the task $j + 1$, and update its starting time
Next i

Pseudocode 2: Pseudocode of the designed schedule generation scheme.

and the makespan of the system is obtained. The pseudocode of the designed schedule generation scheme is shown in Pseudo-code 2.

3.3. Building-Block Based Crossover. The delivery tasks in robots allocation problem are constrained by precedence relationships, so the precedence-based permutation crossover operator is more suitable to solve such problems. Commonly used operators are uniform crossover [21], kX crossover [22], and MBX crossover [16]. The uniform crossover is too disruptive to exploit enough useful information in the parents. kX crossover which is also called k-point crossover first selects k contiguous blocks of the donator parent, then the genes not in the blocks are reserved, and the genes in the blocks are reordered based on their order in the receiver parent. The one-point [23] and two-point crossover [24] all belong to kX operator and are widely used for solving RCPSP. The disruptiveness of the kX operator is decreased as the value of k increases.

The MBX crossover improved the standard two-point crossover. Different from the standard version, the genes in the selected block of donator are reserved. The blocks of the receiver are expanded to the domain containing all these genes, and the genes not in the block of the receiver are also reserved. The genes in the block of the receiver but not in the block of the donator are rearranged on both sides of the block of the donator according to their precedence relations with the genes in the block of the donator. There are some genes which are called free activities that do not have any precedence relations with the gene in the reserved block. They are rearranged on both sides of the reserved block with the same probability. The MBX performs better than the standard two-point crossover because it can exploit more promising characteristics by inheriting the contiguous

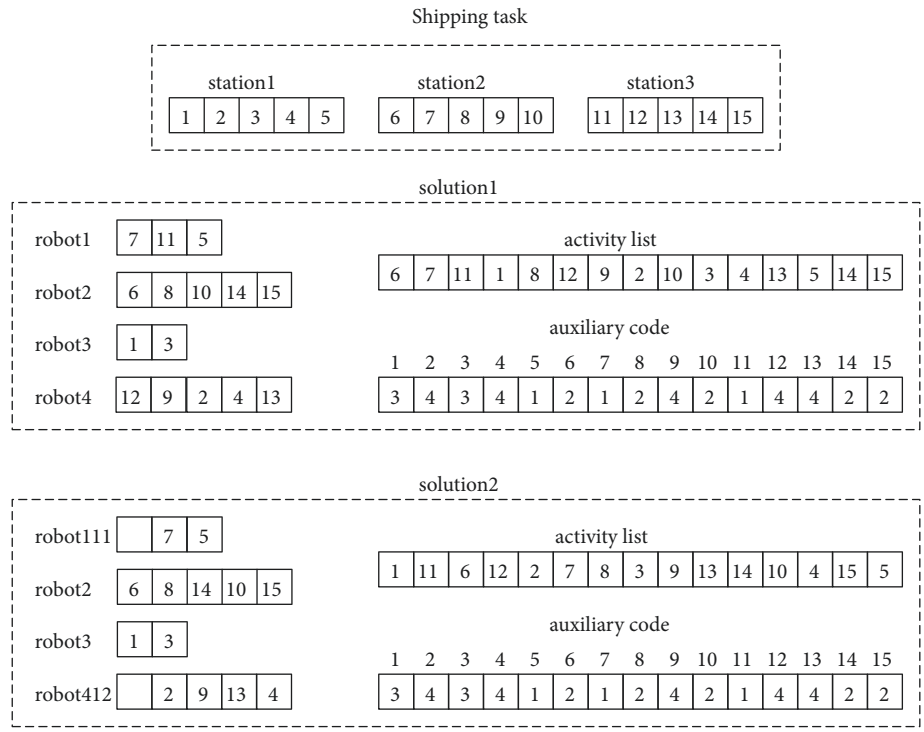

Figure 4: Two solutions to the problem with three picking stations and four robots.

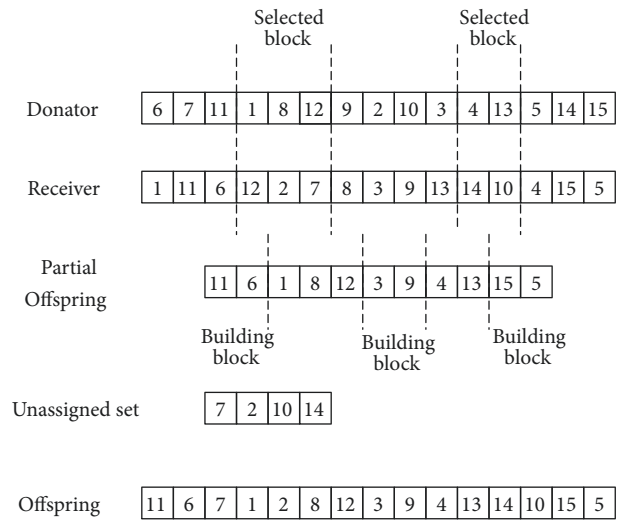

Figure 5: The working process of a two-point building-block based crossover.

genes from both parents. Kadri and Boctor [8] modified the MBX by randomly rearranging the free activities to all possible positions before or after the reserved block while satisfying precedence constraints. It makes the operator more disruptive and yields better results than MBX. They use this operator to solve the RCPSPTT.

We designed a building-block based crossover which exploits the promising information not only by inheriting the contiguous genes but also by the building-block genes. The building-blocks contain the genes both in the unselected parts with the same boundary in donator and receiver regardless of whether they are contiguous. The genes not in the building-blocks are rearranged to all possible positions while satisfying precedence constraints. By introducing more destructiveness while maintaining more reasonable inheritance, the operator balanced diversification and intensification. Moreover, the designed operator is based on k-point crossover which can flexibly adjust the value of k to improve the performance, while MBX is limited in two-point structure.

The building-block based crossover works as follows:

(i) First, randomly select k contiguous block of genes from the donator parent. The genes in the selected block of donator are reserved. We call the parts between each two selected blocks as unselected blocks.

(ii) Second, find the building-blocks of each unselected blocks; i.e., find the genes both in the unselected blocks with the same boundary of donator and receiver, and reorder them based on their order in the receiver parent. The genes not in the building-blocks but in the corresponding unselected blocks of donator are added in the unassigned set one by one according to their sequence in the donator parent.

(iii) Then, the building-blocks and reserved blocks are combined in the offspring solution based on the order of blocks in donator parent.

(iv) Finally, the genes in the unassigned set are rearranged one by one to all possible positions while satisfying precedence constraints. The offspring solution is obtained.

The working process of a two-point building-block based crossover is shown in Figure 5, and the precedence relation is the same as Figure 4. The selected blocks in donator are {1, 8, 12} and {4, 13} and the building block found in receiver is {11, 6}, {3, 9}, and {15, 5}. The unassigned set is {7, 2, 10, 14}. The gene {7} can be inserted to the partial offspring solution between {6} and {8} according to the precedence constraints and will randomly choose a position. In this example, we choose the position before {1}. Other genes in the unassigned set will be inserted in the same way one by one.

3.4. Mutation. Once the crossover operator is performed, the mutation operator is applied to each offspring solution to generate a new one. The offspring solution and the newly generated solution are all retained until the final step of selecting new solutions pool for the next generation. The mutation is based on remove and reinsert, and the probability is P_m. We draw a random number x_i for each gene in the offspring. If $x_i \leq P_m$, the gene will be removed and added into the unassigned set. Then, the genes in the unassigned set will be reinserted to any possible position in the partial solution while meeting the constraints.

4. Results and Discussion

To evaluate the performance of the designed algorithm, we generate some instances of robots allocation problem based on the actual working flow of the RMFS. The proposed algorithm is compared with the rule-based method first-come-first-served (FCFS) used in the practical system, the random activity selection rule (RAND) used in the stage of initialization population, the genetic algorithm using standard two-point crossover [24], and the genetic algorithm using modified MBX [8]. The experiment is programmed in C ++ and runs on a 64-bit win7 system using an Intel Core i7 2.9 GHz CPU with 8GB of memory.

4.1. Instances Generation. The experiment uses a layout similar to Figure 2. The picking stations are located on the left side of the storage area. The robots are shared by all picking stations and drive at a constant speed. We assume that, after finishing the picking process, the robot ships the shelf back to its previous storage location and waits there for the next task. The tasks are generated with two rules: completely random and ABC categorization random. The completely random rule selects each shelf with the same probability when generating a task. The ABC categorization is a zoning strategy which divides the storage area into A, B, and C three zones, and the shelves in different zones are selected with a different probability. We generate 10 instances based on each rule. The total number of shelves is 1800. There are 3 stations with 5 buffers, and each station contains 40 tasks. The tasks are performed with 6, 15, 24, and 42 robots in each instance. The starting position of the robots is randomly generated as the instance is generated. The picking time for each task is randomly generated in units of 10 second. The parameters of RMFS are shown in Table 1.

4.2. Experiments Results. The rule-based method FCFS assigns a task to the earliest idle robot when all the robots are busy. If more than one robot are idle at decision time, FCFS random selects an idle robot. The RAND is iterated the same number of times as other algorithms to make the comparison. The two-point crossover (2-Point) is the standard version. In the modified MBX (MMBX) and the proposed BBX, the genes in the selected reserved blocks are limited to 1/5 of the solution. We use two versions of BBX: two-point based (BBX2) and four-point based (BBX4). All the genetic algorithms (GAs) use the proposed mutation operator. Mutation probability is 0.1, the size of the population is 50, and the number of generations is 50. The initial population for the four GAs is the same. All the algorithms run 10 times for each instance. Since the robot allocation problem is NP-hard and cannot get the optimal solution with a large number of tasks, we use a method similar to the critical path method to relax some constraints to get the lower bound of each instance as the comparison criteria. In each instance, we suppose each shelf has a dedicated robot for transportation (e.g., if there are 116 shelves in 120 tasks, there are 116 robots responsible for handling them,) so that the constraint of the resource "robot" is relaxed and the resource transfer time is neglected. We can get the lower bound (LB) while satisfying the constraints of the resource "shelf" and "station buffer." Table 2 shows the average results of all the algorithms in the completely random instances, and Table 3 shows the average results of all the algorithms in the ABC zoning instances. In the tables, "Ran" represents the instances set of random tasks, "ABC" represents the instances set of ABC zoning based tasks, "MS" represents the results of the average makespan of the system, "DS" represents the results of the average resource transfer time of all the robots, "DEV%" represents the percentage deviation from the results of makespan with the "LB," and "Best" represents the number of times each algorithm achieves the best solution of makespan among others in 10 instances. The best results about "MS," "DS," and " DEV%" obtained for each instances set are highlighted in bold.

As shown in Table 2, the GAs are much better than FCFS in all instances, and the gaps are more obvious with the increase of the available robots. The reason why FCFS is similar in different sets of instances may be that due to the finish-to-start precedence constraints, the tasks are assigned only after their direct predecessor tasks have already arrived at the picking stations, and the number of picking stations is always less than the number of robots. Although the random operator is worse than the GAs, it has also achieved satisfactory results, which proves that the designed schedule generation scheme has much effectiveness. The RAND is significantly worse than the 2-Point in the case of fewer numbers of robots. However, due to the sufficient quantity of robots and the high efficiency of the schedule

Table 1: Parameters of RMFS.

Parameter	Value
Number of shelves in the vertical of shelf block	2
Number of shelves in the horizontal of a shelf block	5
Number of cross aisles	10
Number of aisles	20
Number of shelves	1800
Length of the side aisle	5m
Unit length	1m
Number of picking stations	3
Buffer of picking stations	5
Picking time	[10s,50s]
Number of robots	6, 15, 24, 42
Velocity of robots	1m/s
The time that the robot lifts and lowers the shelf	1s
Number of storage location per zone	A: 20%, B: 30%, C: 50%
Selected probability of shelf	A: 70%, B:25%, C:5%
Number of tasks in each station	40

Table 2: Average results of the completely random instances.

Average result	Ran-6				Ran-15			
	MS	DS	DEV%	Best	MS	DS	DEV%	Best
FCFS	5882	5822.4	286.7	0	5559.7	5784	265.5	0
RAND	4854.3	4114.2	219.1	0	2132.3	3952.6	40.17	0
2-Point	4656.3	4015.5	206.1	0	2117.3	4133.7	39.19	0
MMBX	4599.6	3750.2	202.4	0	2073.9	3863.6	36.33	0
BBX2	4593	3790.1	201.9	0	2075.7	3881.7	36.45	0
BBX4	**4561.3**	**3707.7**	**199.8**	**10**	**2062.3**	**3790.8**	**35.57**	**10**
	Ran-24				Ran-42			
FCFS	5627.7	5884.9	270	0	5631.7	5907	270.2	0
RAND	1546.1	4348.51	1.63	5	1528.6	**3983.1**	0.49	9
2-Point	1547.3	4513.9	1.72	3	1529	3992.6	0.51	9
MMBX	1537.8	4266.2	1.09	5	1528.1	4020.2	0.45	9
BBX2	1538.2	4304.5	1.12	5	1528.5	4008.8	0.48	9
BBX4	**1537.1**	**4218.6**	**1.05**	**10**	**1528**	4029.9	**0.45**	**10**

Table 3: Average results of the ABC zoning instances.

Average result	ABC-6				ABC-15			
	MS	DS	DEV%	Best	MS	DS	DEV%	Best
FCFS	5603.4	5694.9	272.5	0	5283.2	5696.2	251.3	0
RAND	4614.8	3951.8	206.81	0	2038.7	3938.89	35.54	0
2-Point	4536.2	3801.3	201.59	0	2028.6	4101.5	34.87	0
MMBX	4463.9	3534	196.78	0	1975.9	3774.8	31.37	1
BBX2	4468.2	3533.8	197.07	1	1982.4	3858.8	31.8	0
BBX4	**4437.6**	**3498**	**195.03**	**9**	**1968.4**	**3737.3**	**30.87**	**9**
	ABC-24				ABC-42			
FCFS	5319.4	5707.7	253.7	0	5320.9	5739.2	253.8	0
RAND	1529.8	3994.1	1.71	7	**1514.2**	3763.1	**0.67**	**10**
2-Point	1531	4185.2	1.79	6	**1514.2**	3763.1	**0.67**	**10**
MMBX	1524.7	3948.2	1.37	7	**1514.2**	3763.1	**0.67**	**10**
BBX2	1525.5	3981.9	1.42	7	**1514.2**	3762.1	**0.67**	**10**
BBX4	**1524**	**3912.3**	**1.32**	**10**	**1514.2**	**3761.7**	**0.67**	**10**

Table 4: Average results comparison about BBX with different parameters.

(population, generation)		(10,10)			(50,50)
Aver Results	MS	T(s)	Gap	MS	T(s)
Ran6	4831.39	1	5.92%	4561.3	8
Ran15	2130.67	1	3.31%	2062.3	15
Ran24	1547.56	1	0.68%	1537.1	20
Ran42	1528.96	2	0.06%	1528	30
ABC6	4586.1	1	3.35%	4437.6	8
ABC15	2032.94	1	3.28%	1968.4	15
ABC24	1528.96	1	0.33%	1524	20
ABC42	1514.2	2	0	1514.2	30

generation scheme, the RAND is slightly better than the 2-Point GA in the case of a large number of robots. The MMBX and the BBX2 are better than the standard 2-point operator in all instances while the BBX2 is almost the same with the MMBX. It can be easily explained that although the specific operations of the two operators are different, the characteristics of destructiveness and inheritance are similar in the BBX2 and the MMBX. The BBX4 performs best for both makespan and travel time in almost all instances, which proves the improvement effect of increasing the number of cut points.

Furthermore, due to the lack of available robots, it takes a longer time to complete the tasks in the instances of 6 robots, and the percentage deviations of all the algorithms are large. In the instances of 15 robots, the results of all the GAs have been greatly improved, but the performance between algorithms is also obviously different. Then, in the instances of 24 robots, the gaps between different GAs are significantly reduced, and they can approach close to the optimal results. In the instances of 42 robots, the GAs have almost achieved the best solutions which are only slightly larger than LB, which is because the number of robots is saturated for performing the current number of tasks for 3 picking stations. As can be seen, in the cases of an insufficient number of robots, the scheduling scheme has a great impact on system performance, so the difference between the algorithms is more obvious. It demonstrates that the practical RMFS system has such a characteristic that when the number of robots in the system is sufficient or saturated, the scheduling scheme has little effect on the makespan of the system while the robots' path planning and coordination problem is more critical in these cases. However, due to the cost and traffic problems in the practical system, the number of robots is not the more the better. So, determining the right number of robots for an RMFS is also a key area of research, which is out of this paper's scope.

In Table 3, since the tasks are more concentrated and more of them are close to the picking stations, the optimal scheduling can be obtained with fewer robots. In the instances of 24 robots, all the algorithms can almost obtain the optimal solutions, and they have achieved these goals in the instances of 42 robots. Same as the previous discussion, the BBX4 outperforms other operators in all the sets of instances again. The 2-point crossover is the worst one, and MMBX is slightly better than BBX2. The results demonstrate that the proposed algorithm is much effective, in which the designed serial schedule generation scheme can generate good enough solutions firstly, and then BBX can do a further optimization based on that.

It should be noted that the deviation used above is concerning a lower bound instead of the optimal solution. Even for the classical algorithms of RCPSP, the average deviations from the lower bound are relatively large (between 8% and 40 %). Since the addition of the transfer times between the activities in RCPSPTT, the deviation is getting more obvious. Moreover, in the studies of RCPSPTT, the experiment instances are always generated from the classic dataset PSPLIB. Krüger and Schroll [17] introduced the extended instances from the instances in which the optimal solutions are known and the transfer time Δ_{ij} is bounded by the gap between the end of activity i and the start of activity j in the optimal solutions. Poppenborg and Knust [18] randomly generated the transfer time with limit boundaries to ensure that the transfer times are mostly shorter than the processing times. However, their results are still much larger than the lower bound. In our experiments, the location of the required shelves in RMFS is randomly generated, so that the scale of the transfer time is not limited. For this reason, the transfer time may be much longer than the processing time of tasks, which will lead to a larger deviation between the results and the LB in the instances with a small capacity of resource "robots." It can be assumed that the deviations for the optimal solution should be much smaller. Although the resulting LB is a weak lower bound, the effectiveness of the designed serial schedule generation scheme can also be proved, which can approach the optimal solution in the instances with a larger capacity of resource "robots."

Moreover, we do a computation study only using BBX4 operators with all the sets of instances in which the size of the population is 10 and the number of generation is 10. The average result obtained by BBX4 with new parameters is compared with the above results. In Table 4, "(population, generation)" represents the size of the population and the number of generation, "Gap" is the deviation is the percentage deviation of "MS," and "T" represents the average operation time of the algorithms in seconds. The results shown in Table 4 indicate that the runtime is greatly reduced without losing too much performance.

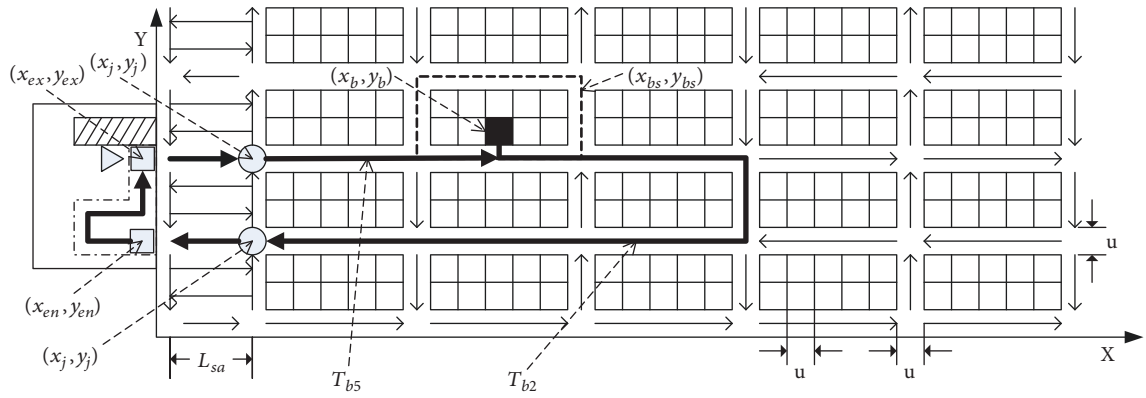

Figure 6: Travel time calculation.

The designed algorithm is much effective and rapidly. It is worth noting that when the number of robots is too much, the GAs can reach the optimal solution very soon. However, they do not end early because of the termination condition.

5. Conclusions

This paper deals with the robots allocation problem in a robotic mobile fulfilment system. We analyze the special workflow of RMFS in detail and model this problem as a resource-constrained project scheduling problem with transfer times (RCPSPTT). Then, a dedicated serial schedule generation scheme is designed considering the specific operational process when shipping a shelf which is required by multiple tasks. For further optimization, we propose a genetic algorithm with a new designed building-block based crossover operator to search various activity permutation as the input of schedule generation scheme, which introduces more destructiveness while maintaining adequate, reasonable inheritance. The experiment results show that the designed algorithm is more effective than the rule-based assignment method which is commonly used in the practical system and outperform other algorithms in all instances.

In future work, more constraints should be considered in this problem such as congestion and battery capacity. Determining the right number of robots for an RMFS is also a key area of research. Moreover, we can use the designed algorithm as the calculation basis to solve other resource allocation problems of RMFS.

Appendix

Since the robot allocation problem depends on their driving behaviour, we need to calculate the travel times of robots in steps 1, 2, and 5. We assume that the picking stations are on the left side of the storage area and the calculation process is as follows.

There are Q_{ca} cross aisles and Q_a aisles in RMFS. N_{cs} is the number of shelves in the vertical of a shelf block and N_{as} is the number of shelves in the horizontal of a shelf block. To simplify the expression, we set the width of an aisle, the width of a cross-aisle, and the width of a shelf all equal to unit length u. The length of the side aisle is L_{sa}. The entrance coordinates of picking station are (x_{en}, y_{en}) and the exit coordinates of picking station are (x_{ex}, y_{ex}). Each shelf has three coordinates:

the relative coordinates in the block,

$$\begin{gathered}(x_{bs}, y_{bs}) \\ x_{bs} \in [1, N_{as}], \\ y_{bs} \in [1, N_{cs}]\end{gathered} \tag{A.1}$$

the relative coordinates of its block,

$$\begin{gathered}(x_b, y_b) \\ x_b \in [1, Q_a], \\ y_b \in [1, Q_{ca} - 1]\end{gathered} \tag{A.2}$$

and the absolute coordinates in the global coordinate system.

$$\begin{gathered}(x_s, y_s) \\ x_s = L_{sa} + (N_{as} \cdot u + u) \cdot (x_b - 1) + (x_{bs} - 1) \cdot u, \\ y_s = u + (N_{cs} \cdot u + u) + (y_{bs} - 1) \cdot u\end{gathered} \tag{A.3}$$

The location coordinates of the current robot are (x_v, y_v) and velocity of robots is v. The time the robot lifts and lowers the shelf is T_l.

In step 1, the travel distance for the unloaded robot is simply the Manhattan distance. The travel time is $T_1 = (|x_v - x_s| + |y_v - y_s|)/v$. In step 2 and step 5, since the loaded robot is travelling along the aisles with a single direction, calculating the travel times is more complicated. We use the parity of the two relative coordinates of the shelf to distinguish different situations. The graphic explanation is shown in Figure 6.

In step 2, the robot first drives in the storage area and then enters the entrance of the picking station through the high-speed area. The travel time of the robot in the storage area is T_{b2}, and the coordinates of the intersection where the robot enters the high-speed area from the storage area are (x_j, y_j) $x_j = L_{sa}$.

(i) y_b is odd and y_{bs} is odd:

if $y_b = 1$ (i.e., the shelf is at the bottom block), $y_j = y_s + 2u$.

x_b is odd, $T_{b2} = (u + (N_{as} - x_{bs}) \cdot u + u + N_{as} + u + N_{cs} + (x_b + 1) \cdot (u + N_{as}))/v$;
x_b is even, $T_{b2} = (u + (N_{as} - x_{bs}) \cdot u + u + N_{cs} + x_b \cdot (u + N_{as}))/v$

else, $T_{b2} = (u + (N_{as} - x_{bs}) \cdot u + u + N_{cs} + x_b \cdot (u + N_{as}))/v$

x_b is odd, $y_j = y_s - 2u - N_{cs}$
x_b is even, $y_j = y_s + N_{cs} - u - u$

(ii) y_b is odd and y_{bs} is even:

$y_j = y_s + u$, $T_{b2} = (u + x_{bs} \cdot u + (x_b - 1) \cdot (u + N_{as}))/v$

(iii) y_b is even and y_{bs} is odd:

$y_j = y_s - u$, $T_{b2} = (u + x_{bs} \cdot u + (x_b - 1) \cdot (u + N_{as}))/v$

(iv) y_b is even and y_{bs} is even:

$T_{b2} = (u + (N_{as} - x_{bs}) \cdot u + u + N_{cs} + x_b \cdot (u + N_{as}))/v$

x_b is odd, $y_j = y_s + N_{cs} - u - u$
x_b is even, $y_j = y_s + 2u + N_{cs}$

So, the travel time of the robot in step 2 is $T_2 = T_{b2} + (|x_{en} - x_j| + |y_{en} - y_i|)/v$

In step 5, the robot leaves from the exit of the picking station and drives to the storage location of the carrying shelf through the high-speed area. Note that we directly use the coordinates of the shelf to represent the coordinates of the storage location. The travel time of the robot in the storage area is T_{b5}, and the coordinates of the intersection where the robot leaves the high-speed area to the storage area are also (x_j, y_j) $x_j = L_{sa}$.

(i) y_b is odd and y_{bs} is odd:

$y_j = y_s - u$, $T_{b5} = (u + x_{bs} \cdot u + (x_b - 1) \cdot (u + N_{as}))/v$

(ii) y_b is odd and y_{bs} is even:

if $y_b = Q_{cb}$ (i.e., the shelf is at the top block), $y_j = y_s - u$.

x_b is odd, $T_{b5} = (u + (N_{as} - x_{bs}) \cdot u + u + N_{as} + u + N_{cs} + (x_b + 1) \cdot (u + N_{as}))/v$;
x_b is even, $T_{b5} = (u + (N_{as} - x_{bs}) \cdot u + u + N_{cs} + x_b \cdot (u + N_{as}))/v$

else, $T_{b5} = (u + (N_{as} - x_{bs}) \cdot u + u + N_{cs} + x_b \cdot (u + N_{as}))/v$

x_b is odd, $y_j = y_s + 2u + N_{cs}$
x_b is even, $y_j = y_s - (N_{cs} - u) - u$

(iii) y_b is even and y_{bs} is odd:

$T_{b5} = (u + (N_{as} - x_{bs}) \cdot u + u + N_{cs} + x_b \cdot (u + N_{as}))/v$

x_b is odd, $y_j = y_s + N_{cs} - u + u$
x_b is even, $y_j = y_s - 2u - N_{cs}$

(iv) y_b is even and y_{bs} is even:

$y_j = y_s + u$, $T_{b5} = (u + x_{bs} \cdot u + (x_b - 1) \cdot (u + N_{as}))/v$

So, the travel time of the robot in step 5 is $T_5 = T_{b5} + (|x_{en} - x_j| + |y_{en} - y_i|)/v$.

Data Availability

The experiment instances are generated based on the actual flow of the RMFS system. All data included in this study are available upon request by contact with the corresponding author.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] T. Lamballais, D. Roy, and M. B. M. De Koster, "Estimating performance in a Robotic Mobile Fulfillment System," *European Journal of Operational Research*, vol. 256, no. 3, pp. 976–990, 2017.

[2] B. Zou, Y. Gong, X. Xu, and Z. Yuan, "Assignment rules in robotic mobile fulfilment systems for online retailers," *International Journal of Production Research*, vol. 55, no. 20, pp. 6175–6192, 2017.

[3] J. J. Enright and P. R. Wurman, "Optimization and coordinated autonomy in mobile fulfillment systems," in *Proceedings of the AAAI Conference on Automated Action Planning for Autonomous Mobile Robots*, pp. 33–38, 2011.

[4] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in *Proceedings of the AAAI-07/IAAI-07 Proceedings: 22nd AAAI Conference on Artificial Intelligence and the 19th Innovative Applications of Artificial Intelligence Conference*, pp. 1752–1760, Canada, July 2007.

[5] Z. Yuan and Y. Y. Gong, "Bot-in-time delivery for robotic mobile fulfillment systems," *IEEE Transactions on Engineering Management*, vol. 64, no. 1, pp. 83–93, 2017.

[6] N. Boysen, D. Briskorn, and S. Emde, "Parts-to-picker based order processing in a rack-moving mobile robots environment," *European Journal of Operational Research*, vol. 262, no. 2, pp. 550–562, 2017.

[7] B. Zou, X. Xu, Y. Gong, and R. De Koster, "Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system," *European Journal of Operational Research*, vol. 267, no. 2, pp. 733–753, 2018.

[8] R. L. Kadri and F. F. Boctor, "An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: the single mode case," *European Journal of Operational Research*, vol. 265, no. 2, pp. 454–462, 2018.

[9] K. Nonobe and T. Ibaraki, "Formulation and tabu search algorithm for the resource constrained project scheduling problem," in *Essays and Surveys in Metaheuristics (Angra dos Reis, 1999)*, vol. 15 of *Oper. Res./Comput. Sci. Interfaces Ser.*, pp. 557–588, Kluwer Acad. Publ., Boston, Mass, USA, 2002.

[10] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version," *European Journal of Operational Research*, vol. 149, no. 2, pp. 268–281, 2003.

[11] M. D. Mahdi Mobini, M. Rabbani, M. S. Amalnik, J. Razmi, and A. R. Rahimi-Vahed, "Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem," *Soft Computing*, vol. 13, no. 6, pp. 597–610, 2009.

[12] A. Gonzalez-Pardo, J. Del Ser, and D. Camacho, "Comparative study of pheromone control heuristics in ACO algorithms for solving RCPSP problems," *Applied Soft Computing*, vol. 60, pp. 241–255, 2017.

[13] R.-M. Chen, "Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7102–7111, 2011.

[14] V. Valls, F. Ballestín, and S. Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 185, no. 2, pp. 495–508, 2008.

[15] Z. Cai and X. Li, "A hybrid genetic algorithm for resource-constrained multi-project scheduling problem with resource transfer time," in *Proceedings of the 2012 IEEE International Conference on Automation Science and Engineering: Green Automation Toward a Sustainable Society, CASE 2012*, pp. 569–574, Republic of Korea, August 2012.

[16] R. Zamani, "A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 229, no. 2, pp. 552–559, 2013.

[17] D. Krüger and A. Scholl, "A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times," *European Journal of Operational Research*, vol. 197, no. 2, pp. 492–508, 2009.

[18] J. Poppenborg and S. Knust, "A flow-based tabu search algorithm for the RCPSP with transfer times," *OR Spectrum*, vol. 38, no. 2, pp. 305–334, 2016.

[19] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: an update," *European Journal of Operational Research*, vol. 174, no. 1, pp. 23–37, 2006.

[20] D. Debels, B. de Reyck, R. Leus, and M. Vanhoucke, "A hybrid scatter search/electromagnetism meta-heuristic for project scheduling," *European Journal of Operational Research*, vol. 169, no. 2, pp. 638–653, 2006.

[21] C. Bierwirth, D. C. Mattfeld, and H. Kopfer, "On permutation representations for scheduling problems," in *Parallel Problem Solving from Nature—PPSN IV*, vol. 1141 of *Lecture Notes in Computer Science*, pp. 310–318, Springer, Berlin, Germany, 1996.

[22] L. Djerid, M.-C. Portmann, and P. Villon, "Performance analysis of permutation cross—over genetic operators," *Journal of Decision Systems*, vol. 5, no. 1-2, pp. 157–177, 1996.

[23] L. Davis, "Job shop scheduling with genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 136–140, 1985.

[24] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Naval Research Logistics (NRL)*, vol. 45, no. 7, pp. 733–750, 1998.