Local search heuristics for the identical parallel machine scheduling with transport robots

Alexander Yuskov^1 **and Yury Kochetov**^{2}

^{1} Novosibirsk State University
1, Pirogova str., Novosibirsk, Russia
a.yuskov@g.nsu.ru

^{2} Sobolev Institute of Mathematics
4 Acad. Koptyug avenue, Novosibirsk, Russia
jkochet@math.nsc.ru

ABSTRACT

We consider a new scheduling problem originating from order picking at an automatic warehouse. We have a finite set of orders. Each order consists of a set of items. All orders should process through the picking stage, where items are extracted from pallets and packed in boxes for customers. We have some identical parallel picking machines and some parking slots for pallets. Each machine can use the pallets from all parking slots. Each pallet should be brought from the automatic warehouse to the parking slot and returned by a transport robot. The fleet of robots is limited. Our goal is to process all orders with minimal makespan. To tackle the problem, we design a mathematical model and three local search heuristics based on the order permutations. A fast decoder procedure creates an active schedule for parallel machines under the robot fleet constraint. Computational experiments for semi-synthetic test instances with up to 345 orders, 30 robots, and 16 parallel machines indicate the high efficiency of the approach.

Keywords: machine scheduling, transport robot, local search, order picking.

2020 Mathematics Subject Classification: 90B35, 90C11, 90C59.

1 Introduction

Complicated warehouses take a significant part in the production process of different companies like online marketplaces, electronic manufacturers, etc. The general structure of such warehouses is common: a huge area, devoted to the storage of hundreds to thousands of items, and a department, which should be able to extract certain items from this storage and send them to customers quickly. The internal structure of such a warehouse consists of several areas, each of which serves its own purpose. The storage area presents a set of racks with shelves where raw materials are located. In automated warehouses, all the loading and unloading procedures in the storage area are performed by Automatic Guided Vehicles (AGVs). The picking process for customer's orders can take more than half of the overall operating cost

(Tompkins, White, Bozer and Tanchoco, 2010). It is essential to use resources efficiently in this stage.

Order serving depends on the internal structure of the warehouse and the requirements of the business and customers. Warehouse operations can be divided into four main stages: receiving goods, storing them in a storage area, picking from the storage area, and shipping to clients.

The first and last stages of this process are least covered in the literature. Trucks with goods arrive at the warehouse and need to be assigned to gates to be unloaded. Similarly, other trucks need to be loaded to send goods to customers. There are also cross-docking warehouses, where received goods are sent directly to the shipping docks. The most common problem that comes from these stages is the truck assignment. For example Tsui and Chang 1992, and Gue 1999 study the carrier-to-dock assignment and Oh, Hwang, Cha and Lee 2006 consider the order-truck assignment.

The primary function of most warehouses is keeping goods before shipping to clients. Thus, the problem of organizing the storage comes in the literature often. The items should be stored in a way to minimize collecting time before shipping. Each type of item can be assigned to a position separately (Malmborg, 1996), or the warehouse can be divided into multiple zones (Jane and Laih, 2005).

There are various ways to organize order picking. Order can be processed by workers or automated robots (Ouzidan, Sevaux, Olteanu, Pardo and Duarte, 2020). The collectors can move around the warehouse, collecting necessary items from the shelves (picker-to-parts) (Eisenstein, 2008), (Henn, Doerner, Strauss and Wäscher, 2003), or the necessary items can be delivered separately to the special picking station (parts-to-picker) (Ouzidan, Pardo, Sevaux, Olteanu and Duarte, 2021; Sarker and Babu, 1995). Each picker can serve only one order at a time (Eisenstein, 2008), or several orders can be picked simultaneously (Henn and Wäscher, 2012). Many researchers focus on batching problems. We need to group orders into batches so that orders in the same batch will be picked together (Van Den Berg, Van Der Hoff et al., 2001; Chen and Wu, 2005).

Transportation of items is mostly considered separately in routing and sequencing problems. In these problems, we need to determine the sequence of orders to minimize processing time or traveling distance, see for example (Van Den Berg and (noud) Gademann, 1999; Roodbergen and De Koster, 2001; Lee and Schaefer, 1997).

The sorting problem appears when multiple orders are picked together. Usually, order pickers place items on a circulation conveyor, and then items enter the assigned shipping lane if all items of the preceding are ready for shipping (Johnson, 1997; Meller, 1997). Some additional problems such as robot movement control (Burohman, Joelianto and Widyotriatmo, 2020) or manipulator control (Aparanji, Wali and Aparna, 2020) can also be applied to warehousing. A detailed literature review for various ways of organizing a warehouse can be found in (de Koster, Le-Duc and Roodbergen, 2007; Roodbergen and Vis, 2009; Gu, Goetschalckx and McGinnis, 2007; Davarzani and Norrman, 2015).

Although there are various papers about warehouse operations, most of them consider problems applied to a specific warehouse structure.

There are some other similar problems which are a generalization of the problem on parallel machines, for example, a multi-stage problem with limited buffer (Wardono and Fathi, 2004) and a problem with sequence-dependent set-up times (Kurz and Askin, 2001).
In this paper, we study a new scheduling problem for identical parallel picking machines with a limited number of parking slots for the AGVs. We know the set of orders (jobs) that have to be processed through the system. Each order consists of a set of items. Each item can be required for multiple orders. All items (raw materials) are in a storage area of the warehouse on pallets. Each pallet has a lot of identical items. They may be moved to the picking area and returned back. The transportation process is carried out by the AGVs. Each AGV can carry only one pallet at a time. We assume that the travel time from the storage to the picking area and back is the same for all pallets. The fleet of AGVs is limited. In this study, we do not consider the routing problem for AGVs. Thus, AGVs are managed as a resource. We assume that each AGV is unavailable when it performs a trip. As soon as the trip is completed, it returns to the fleet and can be used again immediately, no matter where it was located at the end of the previous trip. A similar rule is used in (Davydov, Kochetov, Tolstykh, Xialiang and Jiawen, 2021) where a two-stage flowshop scheduling problem with intermedia buffer is studied. But in contrast, we assume that pallets can stay at the parking slots without robots. In our problem, robots have more freedom for trips.
The order process is possible if and only if all required pallets are in the parking slots. The picking process is carried out by identical picking machines that work in parallel. They have access to all pallets in the parking slots. Each machine can process only one order at a time. We know the processing time for each order. Our goal is to find a schedule for this system with minimal total length. In other words, we deal with a makespan minimization problem.
To tackle the problem, we design a mixed integer nonlinear mathematical model and find a way to linearize it by introducing additional variables and constraints. Our preliminary computational experiments have shown that commercial solver Gurobi can find optimal solutions for small instances only. Thus, we develop three local search heuristics based on ideas of the parallel Tabu Search (TS) and Simulated Annealing (SA) (Gendreau and Potvin, 2019). We present feasible solutions as the permutations of the orders and apply a polynomial time decoding procedure to get feasible schedules for the parallel machines under the AGVs fleet constraint. The decoding procedure returns free pallets to the storage but tries to save some of them if we can use them again soon. The well-known swap and move neighborhoods are used for the local search. Randomized neighborhoods are applied in the Parallel Tabu Search heuristic to reduce the running time for the large-scale instances. To get the initial solution, we design a greedy algorithm based on an idea of minimal distance between the nearest orders. We construct the initial permutation step by step, taking into account the distance to the next order. We prefer the permutations where each two nearest orders need near the same pallets. We use each order as the first one in the permutation and select the best result for the initial solution.
Computational experiments were conducted on the semi-synthetic test instances obtained from a large electronic device manufacturer with up to 345 orders, 121 types of items, 30 transport robots, and 16 identical parallel machines. Each order requires at most 8 pallets. We note

that the most difficult instances have a small number of parking slots. Thus, we generate the benchmarks where the number of parking slots is equal to the maximal number of required pallets among all orders or exceeds it at most by 4. The proposed approach is compared to the four lower bounds with and without the robot fleet and parking slots constraints.
The main contribution of our work is the following:

- We introduce a new scheduling problem for an automated warehouse with identical parallel machines, limited parking slots, and transport robots.
- We present a mixed-integer linear program for this scheduling problem.
- We provide a polynomial time decoding algorithm for order permutations.
- We develop three local search heuristics to find near optimal solutions.
- We conduct computational experiments on the semi-synthetic test instances obtained from a large electronic device manufacturer to show the efficiency of the proposed approach.

The rest of the paper is organized as follows. In Section 2, we present a detailed mathematical formulation of the problem and show a way to linearize it. In Section 3, we design the decoding procedure and propose a general framework of the approach to tackle the problem. In Section 4, we discuss the results of the computational experiments. Section 5 concludes the paper.

2 Mathematical model

Let us introduce the following notations to present the mathematical model formulation: n is the number of orders, m is the number of parallel identical picking machines, C is the number of transport robots, L is the number of types of items, P is the number of parking slots. For order i, we denote by L_i the set of required items and by p_i its processing time. We assume that the traveling time from the storage to the parking slots and the traveling time from the parking slots to the storage is the constant W. Moreover, we need two additional constants: an upper bound for the length of the schedule $M = \sum_{i=1}^{n} p_i$ and an upper bound for the number of robot trips $T = \sum_{i=1}^{n} |L_i|$.

2.1 A mixed-integer non-linear model

The model is formulated as the well-known problem for the identical parallel machine scheduling with additional constraints for transport robots and parking slots for the pallets. For the classical parallel machine scheduling, we introduce the following variables:
$s_i \geq 0$ is the starting time of order i,

$$x_{ij} = \begin{cases} 1, & \text{if orders } i \text{ and } j \text{ are performed on the same machine} \\ & \text{and the order } i \text{ immediately precedes order } j \\ 0, & \text{otherwise} \end{cases}$$

$$f_i = \begin{cases} 1, & \text{if order } i \text{ is the first on its machine} \\ 0, & \text{otherwise} \end{cases}.$$

To control the robot movements and available parking slots, we introduce positive variables τ_k which indicate the time moment of the k-th change event on the parking slots. Without loss of generality, we can control the size of the robot fleet and the number of occupied parking slots in these event points only. Let us introduce the auxiliary variables:

$$d_{kl} = \begin{cases} 1, & \text{if an item } l \text{ is delivering at the event point } k \\ 0, & \text{otherwise} \end{cases}$$

$$r_{kl} = \begin{cases} 1, & \text{if an item } l \text{ is removing at the event point } k \\ 0, & \text{otherwise} \end{cases}$$

$$b_{ki} = \begin{cases} 1, & \text{if order } i \text{ starts after event point } k \\ 0, & \text{otherwise} \end{cases}$$

$$e_{ki} = \begin{cases} 1, & \text{if order } i \text{ finishs before event point } k \\ 0, & \text{otherwise} \end{cases}$$

$$\alpha_{k_0k} = \begin{cases} 1, & \text{if } \tau_k + W > \tau_{k_0} \\ 0, & \text{otherwise} \end{cases}$$

$$\beta_{k_0k} = \begin{cases} 1, & \text{if } \tau_k + 2W > \tau_{k_0} \\ 0, & \text{otherwise} \end{cases}.$$

Now we can present our problem as mixed-integer nonlinear program as follows:

$$\min \max_{i=1,\ldots,n} (s_i + p_i) \tag{2.1}$$

$$s_i \geq s_j + p_j - M(1 - x_{ji}), \quad \forall i, j = 1, \ldots, n, \tag{2.2}$$

$$\sum_{i=1}^{n} x_{ij} \geq 1 - f_j, \quad \forall j = 1, \ldots, n, \tag{2.3}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1, \quad \forall i = 1, \ldots, n, \tag{2.4}$$

$$\sum_{i=1}^{n} f_i \leq m, \tag{2.5}$$

$$\tau_{k+1} \geq \tau_k + 1, \quad \forall k = 1, \ldots, T-1, \tag{2.6}$$

$$s_i \geq \tau_k - M(1 - b_{ki}), \quad \forall k = 1, \ldots, T; i = 1, \ldots, n, \tag{2.7}$$

$$s_i + p_i \leq \tau_k + M e_{ki}, \quad \forall k = 1, \ldots, T; i = 1, \ldots, n, \tag{2.8}$$

$$\sum_{k=1}^{T} b_{ki} d_{kl} - \sum_{k=1}^{T} e_{ki} r_{kl} \geq 1, \quad \forall i = 1, \ldots, n; l \in L_i, \tag{2.9}$$

$$\sum_{k=1}^{k_0} (d_{kl} - r_{kl}) \geq 0, \quad \forall k_0 = 1, \ldots T; l = 1, \ldots, L, \tag{2.10}$$

$$\sum_{k=1}^{k_0} \sum_{l=1}^{L} (d_{kl} - r_{kl}) \leq P, \quad \forall k_0 = 1, \ldots T, \tag{2.11}$$

$$\tau_k + W \leq \tau_{k_0} + M\alpha_{k_0k}, \quad \forall k_0 = 2, \ldots, T; k = 1, \ldots k_0 - 1, \tag{2.12}$$

$$\tau_k + W \geq \tau_{k_0} + 1 - M(1 - \alpha_{k_0k}), \quad \forall k_0 = 2, \ldots, T; k = 1, \ldots k_0 - 1, \tag{2.13}$$

$$\tau_k + 2W \le \tau_{k_0} + M\beta_{k_0k}, \quad \forall k_0 = 2,\dots,T; k = 1,\dots k_0-1, \tag{2.14}$$

$$\sum_{l=1}^{L} d_{k_0l} + \sum_{k=1}^{k_0-1}\sum_{l=1}^{L} d_{kl}\alpha_{k_0k} + \sum_{k=1}^{k_0-1}\sum_{l=1}^{L} r_{kl}(1-\alpha_{k_0k})\beta_{k_0k} \le C, \quad \forall k_0 = 1,\dots,T, \tag{2.15}$$

$$\sum_{k=k_0+1}^{T}\sum_{l=1}^{L} d_{kl}\alpha_{kk_0} + \sum_{l=1}^{L} r_{k_0l} + \sum_{k=1}^{k_0-1}\sum_{l=1}^{L} r_{kl}\alpha_{k_0k} \le C, \quad \forall k_0 = 1,\dots,T. \tag{2.16}$$

Objective function (2.1) defines the length of the schedule which we want to minimize. Constraints (2.2) determine the starting times of orders on each machine. Constraints (2.3) state that every order except the first one on each machine must follow another order. Similarly, the constraint (2.4) says that each order can be followed by at most one other order. The inequality (2.5) shows the number of available machines to perform the orders. Constraints (2.6) specify the order of event points. Constraints (2.7) and (2.8) define the relationships between order starting times and event points. Moreover, we can compute the 0-1 variables b_{ki} and e_{ki} by these inequalities. Constraints (2.9) say that for each order, all its items must be in parking slots during the order processing. We compute how many times robots bring the corresponding pallets and return them. For each item, constraints (2.10) guarantee that the number of its pallets in the parking slots is non-negative at each event point. Constraints (2.11) show the upper bound of parking slots available in each event point. The inequalities (2.12)-(2.14) define the auxiliary 0-1 variables α_{k_0k} and β_{k_0k}. Inequalities (2.15) and (2.16) control the number of robots which moving at each event point. In inequalities (2.15), we calculate the number of robots coming to the parking slots at the event point τ_{k_0}, number of robots coming to the parking slots in time interval $(\tau_{k_0} - W, \tau_{k_0})$, and number of robots starting to the storage in time interval $(\tau_{k_0} - 2W, \tau_{k_0} - W)$ and require at most C robot trips. In other words, we calculate the number of robot trips immediately after the event point $\tau_{k_0} - W$. In inequalities (2.16), we calculate the number of robots starting to the parking slots in time interval $(\tau_{k_0-W}, \tau_{k_0})$, number of robots starting to the storage at the event point τ_{k_0}, and number of robots starting to the storage in time interval $(\tau_{k_0} - W, \tau_{k_0})$. Thus, we calculate the number of robot trips immediately after the event point τ_{k_0} and again require at most C robot trips.

2.2 Linearization

To linearize the model, we introduce new additional variables:

$$\xi_{kil} = b_{ki}d_{kl}, \qquad \eta_{kil} = e_{ki}r_{kl},$$

$$\chi^1_{k_0k} = \alpha_{k_0k}\sum_{l=1}^{L} d_{kl}, \qquad \chi^2_{k_0k} = \beta_{k_0k}(1-\alpha_{k_0k})\sum_{l=1}^{L} r_{kl},$$

$$\zeta_{k_0k} = \alpha_{k_0k}\sum_{l=1}^{L} r_{kl}, \qquad \theta_{k_0k} = \alpha_{kk_0}\sum_{l=1}^{L} d_{kl}.$$

We include new constraints to ensure that the values of these variables are correct.

$$\xi_{kil} \leq b_{ki}, \quad \forall k = 1,...,T; i = 1,...,n; l \in L_i, \tag{2.17}$$

$$\xi_{kil} \leq d_{kl}, \quad \forall k = 1,...,T; i = 1,...,n; l \in L_i, \tag{2.18}$$

$$\eta_{kil} \geq e_{ki} + r_{kl} - 1, \quad \forall k = 1,...,T; i = 1,...,n; l \in L_i, \tag{2.19}$$

$$\chi^1_{k_0k} \geq \sum_{l=1}^{L} d_{kl} - L\left(1 - \alpha_{k_0k}\right), \quad \forall k = 1,...,T; k_0 = 1,...,k-1, \tag{2.20}$$

$$\chi^2_{k_0k} \geq \sum_{l=1}^{L} r_{kl} - L\alpha_{k_0k} - L\left(1 - \beta_{k_0k}\right), \forall k = 1,...,T; k_0 = 1,...,k-1, \tag{2.21}$$

$$\zeta_{k_0k} \geq \sum_{l=1}^{L} r_{kl} - L\left(1 - \alpha_{k_0k}\right), \quad \forall k = 1,...,T; k_0 = 1,...,k-1, \tag{2.22}$$

$$\theta_{k_0k} \geq \sum_{l=1}^{L} d_{k_0l} - L\left(1 - \alpha_{k_0k}\right), \quad \forall k = 1,...,T; k_0 = 1,...,k-1. \tag{2.23}$$

Taking into account new variables, we rewrite some constraints as follows:

$$\sum_{k=1}^{T} \xi_{kil} - \sum_{k=1}^{T} \eta_{kil} \geq 1, \quad \forall i = 1,\ldots,n; l \in L_i, \tag{2.9'}$$

$$\sum_{l=1}^{L} d_{k_0l} + \sum_{k=1}^{k_0-1} \chi^1_{k_0k} + \sum_{k=1}^{k_0-1} \chi^2_{k_0k} \leq c, \quad \forall k_0 = 1,\ldots,T, \tag{2.15'}$$

$$\sum_{k=k_0+1}^{T} \theta_{k_0k} + \sum_{l=1}^{L} r_{k_0l} + \sum_{k=1}^{k_0-1} \zeta_{k_0k} \leq c, \quad \forall k_0 = 1,\ldots,T. \tag{2.16'}$$

We will use the linearized model to obtain exact solutions and lower bounds by commercial software (Gurobi).

3 Metaheuristics

In this section, we present three metaheuristics: tabu search, simulated annealing, and their hybrid algorithm based on the orders permutations. We design a polynomial time decoding procedure to determine the starting time for each order under the fleet robot constraint. We create an active schedule and assign the current order to the first available machine when all necessary items for it are in the parking slots.

3.1 Decoding procedure

To schedule a current order from a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, we need to define which pallets with items (or items for short) must be delivered, assign the trips for robots, and return some useless items to the storage if there are no free parking slots for new items. If we can return more items than necessary, we have to decide which items to return first. In the decoding procedure, we return free items but try to save some of them if we can use them again soon. To present the procedure, we introduce the following auxiliary structures and parameters:

- C_u is the current finish time for machine u. Initially, all elements are zeros;
- T_l is the current maximum finish time of orders with item l. Initially, all elements are zeros;
- q is the queue of robot release events. It contains pairs of release time and number of robots. We use the following operations for it:
  - $q.push(t, c)$ to add pair of releasing c robots at time t to the queue;
  - $q.pop()$ to get the first pair in the queue and remove it;
  - $q.first()$ to get the first pair in the queue;
- B is the current set of items at the parking slots;
- r is the current number of available robots;
- t is the current time.

The pseudo-code for the decoding procedure is presented as Algorithm 1. To process the current order j from the given permutation, we need to deliver all items from the set $L_j \setminus B$ to the parking slots. If we do not have free robots for it (line 5), we have to wait until some robots are released. If there are free parking slots, we decide how many items we can deliver (line 10) and plan the event of releasing these robots (line 11). If all parking slots are occupied (line 12), we need to remove some items. We create a set R_a of items that we can return to storage (line 13). We include in this set all items which will be free at the arrival time of new robots $t+W$. If we do not have such items, we calculate a time when the first item (one or several) will be free, the time T_{min} (line 13). If the set R_a includes some items, we try to save some of them if we can use them again soon. To this end, we sort the set R_a by decreasing of distances. For each item in this set, we find the position of the next order with the same item in the given permutation π and calculate the difference with the current position i. In other words, we place items that are needed for the nearest orders at the end and items for later orders or are already useless at the beginning. We update the number of available robots (lines 16-19) and define the maximal number of items for delivering (line 21), update the queue and remove c useless items from the parking slots (lines 22-24). We repeat the calculations until all items from the set L_j will be at the parking slots. In such a case, we assign the first available machine for the current order j (line 30) and define the completion time for it (line 32). Finally, we update the maximal finish time for each item from the current order (lines 33-35). The makespan of the schedule constructed is the result of the decoding procedure. It is easy to see that the time complexity of the procedure is $O(n^2P^2 + nm)$.
Below we consider a small instance with $n = 9$, $m = 3$, $P = 3$, $C = 2$ and $p_1 = 10, p_2 = 8, p_3 = 2, p_4 = 5, p_5 = 6, p_6 = 6, p_7 = 4, p_8 = 2, p_9 = 1$. $L_1 = \{1\}, L_2 = \{2\}, L_3 = \{2\}, L_4 = \{3\}, L_5 = \{4\},\ L_6 = \{5\}, L_7 = \{6\}, L_8 = \{6, 7\}, L_9 = \{6, 7, 8\}$ and show the schedule for permutation $\pi = \{1, 2, \ldots, 9\}$ in Figure 1. The upper plot of this Gantt chart shows the schedule of orders on machines, the middle one shows the states of the parking slots, and the lower one shows the robot trips. As we can see, the makespan is 18 for this feasible solution.
The optimal solution for this problem with makespan equals 17 is shown in Figure 2.

```
Data: order permutation $\pi$
1  $C_u \leftarrow 0,$ for $u = 1, \ldots, m$; $T_l \leftarrow 0,$ for $l = 1, \ldots, L$; $q \leftarrow \emptyset$; $B \leftarrow \emptyset$; $r \leftarrow C$; $t \leftarrow 0$;
2  for $i \leftarrow 1$ to $n$ do
3      $j = \pi(i)$;
4      while $L_j \setminus B \neq \emptyset$ do
5          if $r = 0$ then
6              $(t, r_{new}) \leftarrow q.pop()$;
7              $r \leftarrow r + r_{new}$;
8          end
9          if $|B| < P$ then
10             $c \leftarrow \min(r, P - |B|, |L_j \setminus B|)$;
11             $q.push(t + W, c)$;
12         else
13             $R_a \leftarrow \{l \in B \mid T_l = \min_{v \in B} I_v \vee T_l \leq t + W\}$;
14             $T_{min} = \min\{T_l, l \in R_a\}$;
15             to sort the set $R_a$ by the distance to next order in $\pi$ for each item ;
16             while $q \neq \emptyset \wedge q.first().time \leq T_{min} - W$ do
17                 $(t, r_{new}) \leftarrow q.pop()$;
18                 $r \leftarrow r + r_{new}$;
19             end
20             $t \leftarrow \max(t, T_{min} - W)$;
21             $c \leftarrow \min(r, |R_a|, |L_j \setminus B|)$;
22             $R \leftarrow$ subset of the set $R_a$ containing the first $c$ items;
23             $B \leftarrow B \setminus R$;
24             $q.push(t + 2W, c)$;
25         end
26         $D \leftarrow$ subset of the set $L_j \setminus B$ containing $c$ items;
27         $r \leftarrow r - c$;
28         $B \leftarrow B \cup D$;
29     end
30     $u(j) \leftarrow \operatorname{argmin}(C_u)$;
31     $s_j \leftarrow \max\left(C_{u(j)}, t + W\right)$;
32     $C_{u(j)} \leftarrow s_j + p_j$;
33     for $l \in L_j$ do
34         $T_l \leftarrow \max(T_l, s_j + p_j)$;
35     end
36 end
37 return $C_{max}(\pi) = max_u C_u$;
```

Algorithm 1: Decoding procedure

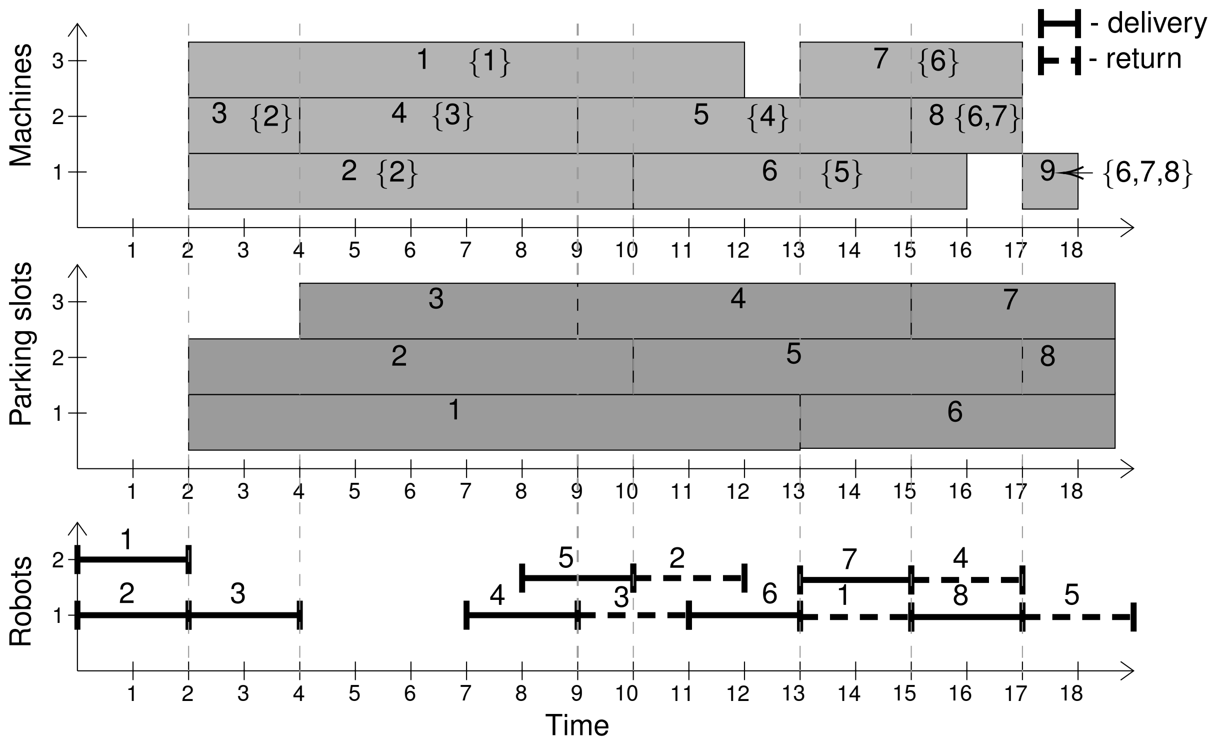

Figure 1: Schedule built by decoding procedure

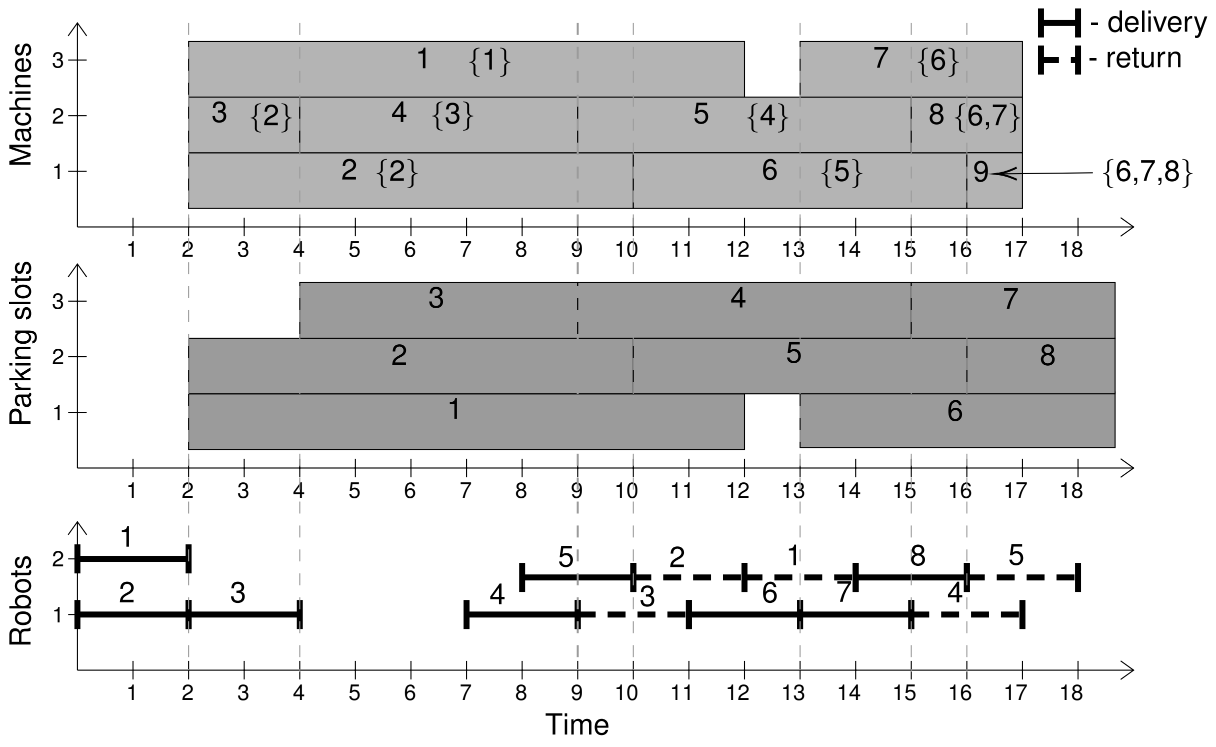

Figure 2: Optimal schedule

Note that this optimal solution cannot be found by order permutations. We need the item permutations to this end. Nevertheless, we apply the order permutations to reduce the search space and try to find the best solution in this area. We guess that the item permutations is extremely large space for local search heuristics.

3.2 Tabu search

Trajectory local search methods and, in particular, Tabu Search (TS) show high performance for solving NP-hard combinatorial optimization problems (Talbi, 2009). They are successfully applied to the facility location problems (Mladenović, Brimberg, Hansen and Moreno-Pérez, 2007), vehicle routing (Erzin, Mladenovic and Plotnikov, 2017; Erzin and Plotnikov, 2015), bilevel programming (Lavlinskii, Panin and Plyasunov, 2015; Panin, Pashchenko and Plyasunov, 2014; Iellamo, Alekseeva, Chen, Coupechoux and Kochetov, 2015), etc. The convergence conditions and bounds for effectiveness of these methods are obtained in (Aarts, Korst and van Laarhoven, 1997; Brimberg, Hansen and Mladenovic, 2010; Kochetov, 2011; Bernhard and Jens, 2012). Below we propose a variant of the TS algorithm to find the best permutation for the problem (2.1)-(2.16).

Denote by $N(\pi)$ a set of permutations which can be obtained from π by one of swap or move operations. In the swap operation, two elements swap their positions in the permutation. In the move operation, one element is moved to a new position. It is easy to see that arbitrary permutation, in particular the optimal permutation, can be obtained from another one by at most n steps to neighboring solutions. As a tabu list ($Tabu$), we use the pairs of permutation elements after their swapping or the element with its previous position after moving to a new position. The length of the tabu list is a constant TL. The list $Tabu$ contains only the information about the last TL modifications of the current permutation. We forbid TL neighboring permutations, which can be expressed in the pairs of numbers: swapping two elements again or returning an element to its previous position. As a randomized neighborhood $N_K(\pi, Tabu)$, we use a random subset of K non-forbidden elements from the neighborhood $N(\pi)$. We pick up these K elements with uniform distribution. The pseudo-code of the TS algorithm is presented in Algorithm 2.

To create the initial permutation, we design a greedy algorithm based on an idea of the minimal distance between the nearest orders. At each step of the algorithm, we select a new order which requires the minimal new items from storage. We use each order as the first one and select the best result for the initial permutation. The parameter k_{stop} controls the running time of the TS algorithm. We terminate the calculations as the incumbent permutation π^* does not change during k_{stop} iterations of the local search. Parameter K controls the running time of each iteration. The large values allow us to intensify the search in the current region of the feasible domain but increase the efforts for each iteration. The small values reduce the running time and allow us to diversify the search. In our computational experiments, we try to find an optimal balance between these opportunities.

```
Create an initial permutation $\pi$, $\pi^* \leftarrow \pi,\ k \leftarrow 1$;
while $k < k_{stop}$ do
    Generate the neighborhood $N_K(\pi, Tabu)$;
    $\pi' \leftarrow \text{argmin}\{C_{max}(\pi'') \mid \pi'' \in N_K(\pi, Tabu)\}$;
    Update the list Tabu and put $\pi \leftarrow \pi'$;
    if $C_{max}(\pi') < C_{max}(\pi^*)$ then
        $\pi^* \leftarrow \pi'; k \leftarrow 1$;
    else
        $k \leftarrow k + 1$;
    end
end
return the best found permutation $\pi^*$;
```

Algorithm 2: Randomized Tabu Search(K, TL, k_{stop})

3.3 Simulated annealing

Simulated annealing (SA) (Kirkpatrick, Gelatt and Vecchi, 1983) also belongs to the class of trajectory methods and is very popular for two reasons. First, it is quite simple and successful in many applications. Second, it is interesting to mathematicians due to its asymptotic convergence properties under some mild constraints for the set of neighboring solutions. Unfortunately, the convergence requires an exponential number of iterations, making SA impractical to find the optimal solution. Nevertheless, it can be used for sophisticated models, like (2.1)-(2.16), as an approximation algorithm with many iterations to compare with other algorithms. The pseudo-code of the SA algorithm is presented in Algorithm 3.

```
Create an initial permutation $\pi$, $\pi^* \leftarrow \pi,\ \alpha \leftarrow \alpha_{max}$ ;
while $\alpha > \alpha_{\min}$ do
    Choose a permutation $\pi' \in N(\pi)$ randomly;
    $\Delta \leftarrow C_{max}(\pi') - C_{max}(\pi)$;
    if $\Delta \leq 0$ then
        $\pi \leftarrow \pi'$
    else
        $\pi \leftarrow \pi'$ with probability $\exp(-\Delta/\alpha)$;
    end
    $\alpha \leftarrow \alpha \cdot \beta$;
    if $C_{max}(\pi') < C_{max}(\pi^*)$ then $\pi^* \leftarrow \pi'$ ;
end
return the best found permutation $\pi^*$;
```

Algorithm 3: Simulated Annealing($\alpha_{\min}, \alpha_{max}, \beta$)

In each step of the SA, we move to a neighboring permutation selected at random with uniform distribution if it is better than the current one. Otherwise, we can move with a probability that depends on the temperature (parameter α). The SA starts with high temperature α_{max}

and terminates with the low temperature α_{min}. The *cooling schedule* is determined by the parameter $\beta < 1$.

3.4 Hybrid TS-SA algorithm

Now we present a hybrid randomized local search algorithm (HTS) with intensification and diversification stages. We apply the randomized TS algorithm and return to the best-found permutation π^* if we cannot improve this solution during k_{restart} iterations. To intensify the search in this region, we enlarge the number of neighboring permutations K (line 13). To diversify the search, we apply the SA algorithm with low temperature and return the initial value of the parameter K (lines 15-16). The HTS algorithm terminates after the R_{stop} intensification and diversification stages. The pseudo-code of the hybrid algorithm is presented in Algorithm 4.

```
1  Create an initial permutation $\pi$, $\pi^* \leftarrow \pi$, $R \leftarrow 0$, $k \leftarrow 0$, $Intensification \leftarrow true$;
2  while $R < R_{stop}$ do
3      Generate the neighborhood $N_K(\pi, Tabu)$;
4      $\pi' \leftarrow \text{argmin}\{C_{max}(\pi'') \mid \pi'' \in N_K(\pi, Tabu)\}$;
5      Update the list Tabu and put $\pi \leftarrow \pi'$;
6      if $C_{max}(\pi') < C_{max}(\pi^*)$ then
7          $\pi^* \leftarrow \pi'$; $k \leftarrow 0$;
8      else
9          $k \leftarrow k + 1$;
10     end
11     if $k > k_{restart}$ then
12         if $Intensification$ then
13             $\pi \leftarrow \pi^*$; enlarge the number of neighbors $K$;
14         else
15             Start the SA from the current permutation $\pi$; $s \leftarrow$ the last permutation of the SA;
16             Reduce the number of neighbors $K$;
17         end
18         $R \leftarrow R + 1$;  $k \leftarrow 0$;  $Intensification \leftarrow \neg Intensification$;
19     end
20 end
21 return  the best found permutation $\pi^*$;
```

Algorithm 4: Hybrid Tabu Search algorithm$(R_{\text{stop}}, k_{\text{restart}})$

The most time-consuming step of the algorithm is to find the best non-forbidden neighboring permutation (line 4). To accelerate the search, we can do it in parallel. We use all available cores of the personal computer for it in the TS and the HTS algorithms. For the SA, it is not used.

4 Computational experiments

We conduct the computational experiments on the semi-synthetic test instances obtained from a large electronic device manufacturer with up to 345 orders, 121 types of items, 30 transport robots, and 16 identical parallel machines. Each order requires at most 8 pallets. Detailed parameters of the instances can be found in the Appendix. We use 20,000,000 iterations as the stopping criterion for the SA and 3000 iterations without improvement for the TS. The running time for large-scale instances on AMD Ryzen 5 3600 3.6GHz with 12 logical cores is about 400 seconds for the SA, 150 seconds for the parallel TS, and 500 seconds for the HTS. In our preliminary computational experiments, we tune the control parameters of the meta-heuristics. As a result, we apply the following values for the TS: the length of tabu list $TL = 3n$, the number of neighboring permutations K is at most 10^4, the number of iterations k_{stop} without improvement of the incumbent permutation π^* is 2000. For the SA, we use α_{max} as a sufficiently large number for moving in arbitrary neighboring solution. To this end, we define α_{max} as the maximum difference between lengths of solutions in the neighborhood for the initial permutation. The cooling schedule is determined in such a way to perform approximately $k_{max} = 2 \cdot 10^7$ iterations and put $\beta = (-\alpha_{max} \log 10^{-2})^{-1/k_{max}}$, $\alpha_{min} = -1/\log 10^{-2}$. For the HTS, we use $R_{stop} = 20$, $k_{restart} = 500$, default neighborhood size K is $2 \cdot 10^3$, extended neighborhood size is 10^4. The SA runs 1000 iterations with $\alpha_{max} = 100$.

Table 1 shows the results of computational experiments. For each test instance, we run the algorithms 10 times and show the best (TS_b, SA_b, HTS_b), the worst (TS_w, SA_w, HTS_w), and the average (TS_a, SA_a, HTS_a) results. The best heuristic results are presented in bold for each instance. Moreover, we include the lower bound (LB) as the best lower bound provided by the Gurobi solver for four models: the whole model (2.1)-(2.16), the simplified model without constraints for robots, the model without constraints for the parking slots, and the model without constraints for robots and parking slots. We terminate Gurobi after an hour of calculations for each model and return the final lower bounds obtained. It is slightly better than LP bounds.

Table 1: Total experimental results

n	TS$_b$	TS$_w$	TS$_a$	SA$_b$	SA$_w$	SA$_a$	HTS$_b$	HTS$_w$	HTS$_a$	LB
14	**2698**	2698	2698.0	**2698**	2698	2698.0	**2698**	2698	2698.0	2329
21	**2339**	2339	2339.0	**2339**	2339	2339.0	**2339**	2339	2339.0	**2339**
21	**2666**	2666	2666.0	**2666**	2666	2666.0	**2666**	2666	2666.0	2339
21	**2442**	2483	2453.9	**2442**	2442	2442.0	**2442**	2442	2442.0	**2442**
25	**2098**	2140	2116.9	**2098**	2098	2098.0	**2098**	2098	2098.0	1805
27	2476	2538	2504.2	**2455**	2476	2466.2	**2455**	2455	2455.0	2128
27	**2775**	2782	2777.8	**2775**	2775	2775.0	**2775**	2775	2775.0	**2775**

n	TS$_b$	TS$_w$	TS$_a$	SA$_b$	SA$_w$	SA$_a$	HTS$_b$	HTS$_w$	HTS$_a$	LB
45	**1794**	1794	1794.0	**1794**	1794	1794.0	**1794**	1794	1794.0	**1794**
45	3499	3772	3609.7	3487	3577	3535.3	**3479**	3500	3488.1	3473
54	1996	2059	2022.2	1982	2052	2003.1	**1954**	1968	1959.7	1629
68	4095	4293	4212.6	**3965**	4215	4066.1	3970	4005	3984.0	3814
76	2972	3002	2987.8	2903	2979	2943.8	**2856**	2891	2881.3	2054
77	**4925**	4926	4925.8	**4925**	4926	4925.8	**4925**	4925	4925.0	4689
89	3029	3513	3170.3	3057	3133	3091.4	**3005**	3029	3017.3	2266
89	4544	4864	4672.3	4387	4613	4473.8	**4350**	4445	4403.6	3808
91	6589	6652	6615.4	**6581**	6672	6597.1	**6581**	6582	6581.9	6076
101	2428	2539	2476.3	2428	2477	2451.6	**2413**	2428	2416.5	2398
101	**7408**	7409	7408.9	**7408**	7408	7408.0	**7408**	7408	7408.0	**7408**
101	5186	5235	5199.5	**5185**	5187	5185.8	**5185**	5186	5185.1	5181
125	5115	5484	5284.5	**4883**	5031	4955.0	4946	5072	4997.5	3106
127	**7551**	7553	7552.1	**7551**	7552	7551.3	**7551**	7551	7551.0	7208
150	7876	8299	8026.7	**7732**	7904	7804.9	7750	7835	7776.4	7649
177	5361	5536	5436.5	**5132**	5285	5217.6	5273	5387	5345.8	3821
178	10153	10855	10530.3	**9971**	10507	10099.6	9978	10116	10055.7	9834
218	5328	5454	5420.0	**4992**	5119	5058.8	5145	5251	5188.4	4499
218	9299	9592	9437.6	**9116**	9355	9247.2	9222	9342	9286.3	8269
319	7717	8050	7812.9	**7310**	7611	7494.4	7637	7780	7719.2	6454
345	27298	27370	27323.9	**27293**	27305	27294.6	**27293**	27294	27293.5	**27293**

We can see that 6 test instances are easy for all heuristics, and the optimal solutions are discovered. For other instances, we can observe that the SA shows slightly better results than the HTS. In many cases, the results coincide. For the worst instance for the HTS ($n = 319$) the relative deviation between the SA and the HTS: $\varepsilon = 100\%(SA_b - HTS_b)/SA_b$ is at most 4.5%. It is interesting to note that the last test instance ($n = 345$) is easy to solve. We find the optimal solution and can prove the optimality. If we compare it with the previous one ($n = 319$), we see the same number of robots, 6 parking slots only, and a small number of machines ($m = 4$). We guess that 8 packing slots for 16 machines is a tough combinatorial case, and the case of 6 parking slots for 4 machines has more freedom and is easy from this point of view.

5 Conclusion

We have considered a new scheduling problem for identical parallel picking machines with limited parking slots and fleet transport robot constraints. It is an NP-hard combinatorial optimization problem originating from order picking at an automatic warehouse. We design the mixed-integer linear program and apply the commercial software Gurobi to find lower and upper bounds for makespan. Our computational experiments have shown that it is useful for small test instances only. Therefore, we have developed metaheuristics based on the order permutation decoding procedure. It is a fast polynomial time algorithm to create an active schedule under the fleet robot constraint. We have adopted the Tabu Search and Simulated Annealing frameworks and developed a Hybrid TS-SA algorithm to tackle the problem. Computational experiments were conducted on the semi-synthetic test instances obtained from a large electronic device manufacturer with up to 345 orders, 121 types of items, 30 transport robots, and 16 identical parallel machines. Each order requires at most 8 pallets. The proposed approach is compared to the lower bound with and without the robot fleet and parking slots constraints. Unfortunately, this lower bound is weak for large-scale instances. Thus, we apply the asymptotically exact SA metaheuristic with a huge number of iterations to compare results for the HTS algorithm. We have found the optimal solutions for 6 test instances and observed small deviations between the SA and the HTS. For future research, it is interesting to modify the decoding procedure and combine the order permutations with item permutations.

Acknowledgment

This work was supported by Russian Science Foundation grant 21-41-09017.

References

Aarts, E. H. . L., Korst, J. H. M. and van Laarhoven, P. J. M. 1997. *Simulated annealing*, Princeton University Press, pp. 91–120.
URL: *https://doi.org/10.1515/9780691187563-007*

Aparanji, V. M., Wali, U. V. and Aparna, R. 2020. Multi-layer auto resonance network for robotic motion control, *International Journal of Artificial Intelligence* **18**(1): 19–44.

Bernhard, K. and Jens, V. 2012. *Combinatorial Optimization : Theory and Algorithms.*, Vol. 21 of *Algorithms and Combinatorics*, 5 edn, Springer.

Brimberg, J., Hansen, P. and Mladenovic, N. 2010. Attraction probabilities in variable neighborhood search, *4OR* **8**(2): 181–194.

Burohman, A. M., Joelianto, E. and Widyotriatmo, A. 2020. Collision-free stable polygon formation of multi-agent robots, *International Journal of Artificial Intelligence* **18**(1): 163–176.

Chen, M.-C. and Wu, H.-P. 2005. An association-based clustering approach to order batching considering customer demand patterns, *Omega* **33**(4): 333–343.
URL: *https://www.sciencedirect.com/science/article/pii/S030504830400088X*

Davarzani, H. and Norrman, A. 2015. Toward a relevant agenda for warehousing research: literature review and practitioners' input, *Logistics Research* **8**(1).
URL: *http://dx.doi.org/10.1007/s12159-014-0120-1*

Davydov, I., Kochetov, Y., Tolstykh, D., Xialiang, T. and Jiawen, L. 2021. Hybrid variable neighborhood search for automated warehouse scheduling, *(submitted)Optimization Letters* .

de Koster, R., Le-Duc, T. and Roodbergen, K. J. 2007. Design and control of warehouse order picking: A literature review, *European Journal of Operational Research* **182**(2): 481–501.

Eisenstein, D. D. 2008. Analysis and optimal design of discrete order picking technologies along a line, *Naval Research Logistics* **55**(4): 350–362.

Erzin, A., Mladenovic, N. and Plotnikov, R. 2017. Variable neighborhood search variants for min-power symmetric connectivity problem, *Computers & Operations Research* **78**: 557–563.
URL: *https://www.sciencedirect.com/science/article/pii/S0305054816301174*

Erzin, A. and Plotnikov, R. 2015. Using vns for the optimal synthesis of the communication tree in wireless sensor networks, *Electronic Notes in Discrete Mathematics* **47**: 21–28. The 3rd International Conference on Variable Neighborhood Search (VNS'14).
URL: *https://www.sciencedirect.com/science/article/pii/S1571065314000468*

Gendreau, M. and Potvin, J.-Y. (eds) 2019. *Handbook of Metaheuristics*, Vol. 272 of *International Series in Operations Research & Management Science*, 3 edn, Springer International Publishing, Cham.
URL: *https://link.springer.com/10.1007/978-3-319-91086-4*

Gu, J., Goetschalckx, M. and McGinnis, L. F. 2007. Research on warehouse operation: A comprehensive review, *European Journal of Operational Research* **177**(1): 1–21.

Gue, K. R. 1999. Effects of trailer scheduling on the layout of freight terminals, *Transportation Science* **33**(4): 419–428.

Henn, S., Doerner, K. F., Strauss, C. and Wäscher, G. 2003. Metaheuristics for the Order Batching Problem in Manual Order Picking Systems, *BuR – Business Research* **3**(1): 82–105.

Henn, S. and Wäscher, G. 2012. Tabu search heuristics for the order batching problem in manual order picking systems, *European Journal of Operational Research* **222**(3): 484–494.
URL: *http://dx.doi.org/10.1016/j.ejor.2012.05.049*

Iellamo, S., Alekseeva, E., Chen, L., Coupechoux, M. and Kochetov, Y. 2015. Competitive location in cognitive radio networks, *4OR* **13**(1): 81–110.

Jane, C.-C. and Laih, Y.-W. 2005. A clustering algorithm for item assignment in a synchronized zone order picking system, *European Journal of Operational Research* **166**(2): 489–496.
URL: *https://www.sciencedirect.com/science/article/pii/S0377221704001201*

Johnson, M. E. 1997. The impact of sorting strategies on automated sortation system performance, *IIE transactions* **30**(1): 67–77.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. 1983. Optimization by simulated annealing, *Science* **220**(4598): 671–680.
URL: *https://science.sciencemag.org/content/220/4598/671*

Kochetov, Y. 2011. Facility location: discrete models and local search methods, *in* V. Chvatal (ed.), *Combinatorial Optimization. Methods and Applications*, IOS Press, pp. 97–134.

Kurz, M. E. and Askin, R. G. 2001. Heuristic scheduling of parallel machines with sequence-dependent set-up times, *International Journal of Production Research* **39**(16): 3747–3769.
URL: *https://www.tandfonline.com/doi/full/10.1080/00207540110064938*

Lavlinskii, S. M., Panin, A. A. and Plyasunov, A. V. 2015. A bilevel planning model for public–private partnership, *Automation and remote control* **76**(11): 1976–1987.

Lee, H. F. and Schaefer, S. K. 1997. Sequencing methods for automated storage and retrieval systems with dedicated storage, *Computers & Industrial Engineering* **32**(2): 351–362.
URL: *https://www.sciencedirect.com/science/article/pii/S0360835296002987*

Malmborg, C. J. 1996. Storage assignment policy tradeoffs, *International Journal of Production Research* **34**(2): 363–378.

Meller, R. D. 1997. Optimal order-to-lane assignments in an order accumulation / sortation system, *IIE Transactions* **29**(4): 293–301.
URL: *https://doi.org/10.1080/07408179708966335*

Mladenović, N., Brimberg, J., Hansen, P. and Moreno-Pérez, J. A. 2007. The p-median problem: A survey of metaheuristic approaches, *European Journal of Operational Research* **179**(3): 927–939.
URL: *https://www.sciencedirect.com/science/article/pii/S0377221706000750*

Oh, Y., Hwang, H., Cha, C. N. and Lee, S. 2006. A dock-door assignment problem for the Korean mail distribution center, *Computers and Industrial Engineering* **51**(2): 288–296.

Ouzidan, A., Pardo, E. G., Sevaux, M., Olteanu, A.-L. and Duarte, A. 2021. BVNS Approach for the Order Processing in Parallel Picking Workstations, *Lecture Notes in Computer Science*, Springer Nature Switzerland AG, pp. 176–190.
URL: *http://link.springer.com/10.1007/978-3-030-69625-2_14*

Ouzidan, A., Sevaux, M., Olteanu, A. L., Pardo, E. G. and Duarte, A. 2020. On solving the order processing in picking workstations, *Optimization Letters* .
URL: *https://doi.org/10.1007/s11590-020-01640-w*

Panin, A. A., Pashchenko, M. G. and Plyasunov, A. V. 2014. Bilevel competitive facility location and pricing problems, *Automation and Remote Control* **75**(4): 715–727.

Roodbergen, K. J. and De Koster, R. 2001. Routing methods for warehouses with multiple cross aisles, *International Journal of Production Research* **39**(9): 1865–1883.

Roodbergen, K. J. and Vis, I. F. 2009. A survey of literature on automated storage and retrieval systems, *European Journal of Operational Research* **194**(2): 343–362.
URL: *http://dx.doi.org/10.1016/j.ejor.2008.01.038*

Sarker, B. R. and Babu, P. S. 1995. Travel time models in automated storage/retrieval systems: A critical review, *International Journal of Production Economics* **40**(2-3): 173–184.

Talbi, E.-G. 2009. *Metaheuristics: from design to implementation*, Vol. 74, John Wiley & Sons.

Tompkins, J. A., White, J. A., Bozer, Y. A. and Tanchoco, J. M. A. 2010. *Facilities Planning*, John Wiley & Sons.

Tsui, L. Y. and Chang, C. H. 1992. An optimal solution to a dock door assignment problem, *Computers and Industrial Engineering* **23**(1-4): 283–286.

Van Den Berg, J. P. and (noud) Gademann, A. J. 1999. Optimal routing in an automated storage/retrieval system with dedicated storage, *IIE Transactions (Institute of Industrial Engineers)* **31**(5): 407–415.

Van Den Berg, J. P., Van Der Hoff, H. H. et al. 2001. An order batching algorithm for wave picking in a parallel-aisle warehouse, *IIE transactions* **33**(5): 385–398.

Wardono, B. and Fathi, Y. 2004. A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities, *European Journal of Operational Research* **155**(2): 380–401.
URL: *https://linkinghub.elsevier.com/retrieve/pii/S0377221702008731*

Appendix A. Test data parameters

Table 2: Test data parameters

n	m	C	P	W	$\min p_i$	$\max p_i$	$\min \|L_i\|$	$\max \|L_i\|$
14	6	30	6	714	230	1044	1	6
21	16	30	8	702	216	1211	1	5
21	6	30	5	709	216	1630	1	5
21	6	30	3	720	216	355	1	3
25	6	30	4	704	216	320	1	4
27	6	30	6	735	216	682	1	5

n	m	C	P	W	$\min p_i$	$\max p_i$	$\min \lvert L_i \rvert$	$\max \lvert L_i \rvert$
27	4	30	8	735	216	682	1	5
45	16	30	8	742	223	989	1	4
45	6	30	4	742	223	989	1	4
54	16	30	8	721	216	571	1	5
68	6	20	5	743	216	933	1	5
76	16	30	8	736	216	627	1	6
77	6	20	5	746	216	1630	1	5
89	16	30	8	716	216	794	1	6
89	8	20	6	716	216	794	1	6
91	6	20	6	744	216	1630	1	6
101	16	30	8	728	216	877	1	5
101	4	30	6	728	216	877	1	5
101	6	20	5	728	216	877	1	5
125	16	30	8	729	216	933	1	6
127	6	20	8	733	216	1630	1	8
150	6	20	6	742	216	982	1	6
177	16	30	8	731	216	1253	1	8
178	6	20	6	739	216	1128	1	6
218	16	30	8	729	216	1630	1	5
218	8	20	5	729	216	1630	1	5
319	16	30	8	731	216	1211	1	6
345	4	30	6	718	216	1630	1	6