Heuristic Algorithm for the Co-Operative Picking with AGVs in Warehouse

Takayoshi Yokota

Abstract: The performance of order picking in a warehouse is crucial in modern logistics. Automated guided vehicles (AGVs) have been developed recently and are in operation in many warehouses. Human pickers still play an important role in picking items that are different in shape and size and putting them on AGVs. Hence, co-operation among human pickers and AGVs is important. This co-operation should be effective in enabling every AGV and human to work efficiently. This paper proposes an algorithm that generates a sub-optimal co-operation schedule for both AGVs and human pickers. The effectiveness of the algorithm was evaluated through computer simulations involving an actual warehouse represented as a two-dimensional lattice-network model.

Key Words: AGV, picking, logistics, warehouse, min-max strategy.

1. Introduction

In recent years, order picking in warehouses has improved, especially with the advent of internet-based business to customer (B2C) shopping and other business to business (B2B) logistics. With B2B logistics, each order usually consists of several items, for instance, 10 to 20 items, in which each item is stored in a different position in a warehouse even though items are optimally arranged by a thorough statistical study of the correlation among them. Human pickers pick items from warehouse shelves and put them on an appropriate automated guided vehicle (AGV). AGVs are autonomous vehicles that operate in accordance with a pre-planned route to pick up items, store them, and carry them to the packing area. In this paper, the author assumes that AGVs cannot pick items by themselves, requiring human pickers to pick items and place them on the AGV, because the warehouse stores various types of items with different sizes and shapes that cannot be picked up automatically by AGVs. The author proposes an algorithm that generates a schedule of sub-optimal co-operation of AGV’s and human pickers. AGVs’ routes, i.e., the schedule of visiting shelves in the warehouse, are assumed to be optimized with minimum trip length criteria. Such scheduling of AGVs has been described in the publication [1].

Due to thousands of orders that are made comes every day, in which multiple kinds of items can be required in each order, the algorithm needs to be suitably fast. The schedule for AGVs and human pickers must be generated within an hour on the basis of this amount of orders. However, because of the limited nature of the exhaustive search in previous work, the proposed algorithm sometimes outperformed the solution obtained by the limited exhaustive search in a number of instances. In this paper, the author categorizes the exhaustive search algorithms into three levels and formulates the sizes of solution spaces. The author also introduces an optimization algorithm based on simulated annealing for comparison. The results are shown in Section 3.1. Here, the roles of AGV and human pickers are described below.

(a) Role of AGVs

Start from the start point and visit shelves in the warehouse in accordance with the scheduled route. Wait for a human picker to pick items of the order and place them on the AGV. Move to another shelf on the scheduled plan. If there are no items left to be picked, go to the packing area and then to the start area.

(b) Role of human pickers

Start from the start point and visit shelves in the warehouse in accordance with the human pickers schedule plan where an AGV is scheduled to come. If the AGV arrives or already waiting, pick the items and place them on the AGV. Move to another shelf in accordance with the human picker scheduled plan.

Algorithms for handling logistics in warehouses vary widely depending on the type and amount of items stored, the manner in which they are shipped, and the size of the warehouses. The author focuses on order picking for this study. The author assumes that, in order picking, picking and shipping are targeted for each order. An AGV collects items that are listed in...
one order of a set of orders by traveling around a warehouse with the help of human pickers. It is said that 50% of the total time consumed in order picking is the traveling time in a warehouse [3]. Therefore, batching in which several orders are combined to reduce traveling time in a warehouse is often considered [3]–[6]. Another approach is called zone picking [7],[8]. Several papers discuss collision avoidance and route planning of AGVs [9]–[11]. Online rescheduling of picking agents was discussed in [12]; however, no co-operation with human pickers is considered. To the best of the author’s knowledge, there is no literature referring to the optimization of the co-operation between AGVs and human pickers. In the author’s previous publication [2], a new heuristic algorithm to generate co-operative picking with AGVs in warehouse was proposed and evaluated and compared with a limited exhaustive search algorithm. In this paper, the author categorizes the exhaustive search algorithms into three levels and formulates the sizes of solution spaces. The limited exhaustive search of the previous paper [2] is now called the Level 1 exhaustive search, and its size of the solution space is far smaller than those of the Level 2 and 3 exhaustive searches. Details of each level will be described in Section 3.1.

For research into the collaborative operation in the warehouse, Kiva [8],[13],[14] is well-known. In the Kiva system, the picking of items is automatically performed by autonomous mobile robots who carry pods (i.e., movable shelves) in which desired items are stored. The robots carry the pod to the human picker who picks the desired item(s) from the pod. In this system, the human picker does not have to walk to the pod. The pod is brought to them and then returned to the warehouse. Items are stored in pods of uniform shape so that the robot can lift and carry them all the way to a human picker. In this way, the human picker does not have to walk around and can just wait for the robot to bring the pod. Hence, with the Kiva system, no one has to walk around the warehouse. This situation is highly sophisticated. However, there are still many cases in which items are very different in shape, size, and weight and can not easily be stored in uniform shape movable pods such as those used in the Kiva system. In such cases, pods must be fixed and remain still, and human pickers have to walk up to the pod as well as support the AGVs. As long as human pickers are necessary, the co-operation between AGVs and human pickers must be in an optimal manner. The optimization of the schedule for each AGV and each human picker is the focus of this paper.

2. Order Picking Based on AGVs and Pickers
2.1 Definition of Problem

In this section, we formulate order picking on the basis of AGVs and human pickers. Figure 1 illustrates the problem definition. Suppose there are many retail shops throughout a country or the world, and they all send orders to their local center every day. The local center aggregates these orders and then sends batched orders to a warehouse. In modern warehouses, many types of items are stored on shelves. Assume each AGV is assigned a batched order from the local center that takes care of orders from local retail shops. The problem is how to assign the batched orders to AGVs to maintain high efficiency. Items of interest in the warehouse vary in shape, and human pickers have to put them on AGVs. Therefore, co-operation between AGVs and human pickers is necessary. Hereafter, batched orders are simply referred to as orders, and human pickers are referred to as pickers. In the warehouse, these orders are assigned to AGVs, and each AGV moves around the warehouse to gather items in orders. Pickers move around and put items on the AGVs. In this case, there are three optimization problems, i.e., the optimum assignment of orders to AGVs, the optimum routing of AGVs, and the optimum assignment of pickers to AGVs. In a previous study, a fast sub-optimal routing algorithm was developed that minimizes the trip length of AGVs sub-optimally in real time [1], which is not the focus of this paper. This paper focuses on the optimum assignment of orders to AGVs and the optimum assignment of pickers to AGVs. The objective of the optimization is to minimize the task completion time, which is to minimize the maximum completion time of picking. All AGVs and pickers start at the same position, and once all items are picked and all AGVs return to the goal position, the task is completed. Figure 2 shows an example of a schedule of three AGVs represented in time and 2-D position. The x and y axes represent the positions of the items to be picked in the warehouse, and the t axis represents time. In this situation, the author assumes the AGVs cannot pick items and need pickers to collect the items and put them on the AGVs. It is desirable that pickers are waiting for an AGV to come to the appropriate shelf. To achieve this, the author simultaneously optimizes the AGVs picking schedule and picker assigning schedule, which is a combinational problem with a solution space that may easily explode even for small cases.

The requirement for pickers can be formulated as the demand for pickers as follows:

\[
\text{demand}(i,t) = \sum_{k=1}^{K_i} p_{i,k} \cdot \delta_{\tau_{i,k}}, \quad t = 1, 2, ..., M, \tag{1}
\]

where \(p_{i,k}\) is the 2-D position vector of the \(i\)th AGV when it stops at the \(k\)th destination for picking, \(M\) is the number of AGVs, and \(K_i\) is the number of destinations of the \(i\)th AGV. This equation activates the demand position vector when \(t = \tau_{i,k}\) holds, and \(\text{demand}(i,t)\) forms a time series of demand position vectors. The notations \(\tau_{i,k}\) is the scheduled arrival time of the \(i\)th AGV to its \(k\)th destination for picking, and \(\delta_{\tau_{i,k}}\) is a Kronecker delta function defined as follows:

![Fig. 1 Problem definition: function of warehouse.](image-url)
Fig. 2 Problem definition: order picking and its schedule.

\[
\delta_{t,\tau_i, k} = \begin{cases} 
1, & \text{if } t = \tau_i, k, \\
0, & \text{if } t \neq \tau_i, k.
\end{cases}
\] 

(2)

The author assumes that the time axis is digitized and \( t \) and \( \delta_{t,\tau_i, k} \) take discrete values. The total demand \( \text{total demand}(t) \) for pickers is given by the following equation:

\[
\text{total demand}(t) = M \sum_{i=1}^{\text{demands}} \sum_{k=1}^{K_i} p_{i,k} \cdot \delta_{t,\tau_i, k}.
\] 

(3)

Equation (3) represents the time series of the points illustrated in Fig. 3, which is a synthesis of the three diagrams in Fig. 2. A demand of 1 in Fig. 3 (b) means that there is a demand for a picker to put items on an AGV. If no demand for pickers exists, demand is 0. Note that any delay in the schedule due to the lack of supporting pickers is not taken into account at this stage, and the time series of Fig. 3 (a) is the ideal schedule that may be affected by disruption. The 3-D demand representation in Fig. 3 (a) is reduced to a 2-D time chart such as that shown in Fig. 3 (b), which is easier to handle. The position information can be easily associated with each demand for a picker.

Figure 4 shows an example of the demands of three AGVs for pickers in an actual case. If we denote the picking completion time of each AGV as \( TC_i \) and their maximum value \( TC_{\text{MAX}} \) as Eq. (4), the objective of the optimization is to minimize the maximum value \( TC_{\text{MAX}} \) for both the AGVs picking schedule and pickers assignment schedule.

\[
TC_{\text{MAX}} = \max\{TC_i, i = 0, 1, 2, \ldots, M - 1\}
\] 

(4)

2.2 Equalization of Burden of Orders on Each AGV

With the proposed algorithm, the burden of orders on each AGV is equalized as evenly as possible. The burden means the travel length or the travel time of each AGV required to pick items. If the travel length of an AGV is extremely longer than those of other AGVs, the task completion time will be determined solely by the AGV with the longest travel length. Therefore, equalization of burden, i.e., travel length, is desirable. The author first carries out preliminary equalization by the following simple step: each order is assigned to the AGV with the least burden. This step is repeated until no order is left unassigned. The burden is estimated by the time required for the AGV to move to the ordered item in the warehouse. The sub-optimum route for each order is previously obtained using the fast sub-optimum routing algorithm [1], and the total travel time for each order is known. The sub-optimum routing algorithm is a heuristic traveling salesman problem algorithm in which the order of picking items is arranged to minimize the total travel path length in the Manhattan distance of each AGV. The results of the preliminary equalization are then optimized using a more elaborate optimization algorithm based on simulated annealing. Both preliminary and optimum results of equalization obtained from the simulated annealing algorithm for five AGVs are listed in Table 1.

In this case, a batch of 100 orders was distributed as evenly as possible to five AGVs. Each of the 100 orders consisted of 2 to 23 items, and the average was 12.7 items. The number of
items in each order is shown in Fig. 5. During the preliminary equalization, the maximum deviation was 76 steps, while the simulated annealing optimization reduced it to two steps and reached a strict optimum solution. Steps mean the number of steps for the AGV to travel. One step corresponds to 0.5 m. The author modeled an actual warehouse for the evaluation of the proposed algorithm. The warehouse was modeled using a two-D lattice network as illustrated in Fig. 6. The size of each square is $0.5 \times 0.5$ m, and that of the warehouse is about $19 \times 17.5$ m. AGVs and pickers start from and return to the same position marked as “Start and Goal” in Fig. 6.

2.3 Assignment of Pickers to AGVs

The total number of demand events $Q$ in Fig. 3 (b) is given by

$$Q = \sum_{i=1}^{M} K_i.$$  \hspace{1cm} (5)

The author assumes that we have $N$ pickers and the combinatorial number of assigning $N$ pickers to $Q$ demands is $N^Q$. This number easily increases explosively as $N$ and $Q$ increase. For instance, 10 pickers and 10 AGVs with 10 destinations, the number increases to $10^{10}$. Even a probabilistic optimization algorithm, such as the genetic algorithm, may not be feasible for this huge search space. Hence, we propose a heuristic algorithm that can be used in real time. The author adopted the following strategies to narrow the search space:

**Strategy 1)** Assign a picker to an AGV that is closest to the picker rather than those that are not close.

**Strategy 2)** A picker should be assigned to an AGV who is most behind in the schedule.

These two simple strategies greatly reduce the size of the search space and computation time. The proposed algorithm for assigning pickers to AGVs is summarized in Algorithm 1. After equalization of burdens of orders on each AGV, a sub-optimum schedule for each AGV is obtained. From these schedules, the time chart illustrated in Fig. 3 (b) is synthesized. Then the assigning of a picker and time chart updating are carried out in the while-do loop of the proposed algorithm as shown in Algorithm 1.

In the proposed algorithm, pickers are assigned in accordance with Step 7 of Algorithm 1. The Level 1 and 2 exhaustive search algorithm and the simulated annealing algorithm are described in Section 3.1. In these algorithms, pickers are assigned to AGVs in accordance with an assignment sequence pattern that minimizes the task completion time by executing an exhaustive simulation for a given AGV schedule (picking sequence).

3. Evaluation of Proposed Algorithm

In this section, we describe some of the evaluation results of the proposed algorithm. The author conducted several computer simulations of order picking problems in which AGVs and pickers co-operate using the warehouse layout shown in Fig. 6.

3.1 Simple Evaluation for Small Cases

Before we applied the algorithm to real-world order picking problems of hundreds of orders, we conducted a simple evaluation for small problems ($N = 2$, $M = 3$, $Q = 3, 6, 9, 12, 15, 18$, and 20) to determine the quality of the solutions obtained with the proposed algorithm and compared it with that of the solutions obtained with an exhaustive search algorithm and a stochastic optimization algorithm. A well-known stochastic optimization algorithm, the genetic algorithm (GA), is not readily applicable to our problem because of the difficulty in representing AGV schedules

Algorithm 1 Assign a picker to an AGV

1: Input order list.
2: Distribute orders as evenly as possible to $AGV_i, i = 1, 2, ..., M$
3: Generate ideal schedule $S_i$ for each $AGV_i, i = 1, 2, ..., M$
4: Create a time chart by synthesizing ideal schedules ideal schedule $S_i$ for each $AGV_i, i = 1, 2, ..., M$
5: while Demand for picker exists in the time chart do
6: Look at the time chart and select the demand from an AGV which is most behind the ideal schedule
7: Assign a free picker who can move to the AGV fastest.
8: if no picker is available and cannot be assigned to the AGV then add delay to the corresponding time chart
9: end if
10: if the picking finishes then free the picker, delete the demand from the time chart
11: end if
12: end while
as genes. This is because the visiting sequence in the AGV schedules, such as those shown in Fig. 3, is not independent of each other and mutually highly dependent, and the crossover operation in the GA may generate a significantly different solution from the parent solutions. Hence, both the development and evaluation of a GA for our problem is our future research. Simulated annealing is a simple algorithm that considers neighboring solutions from the current one randomly avoiding local optima by accepting worse solutions probabilistically, which is controlled by a temperature parameter. The algorithm is described as Algorithm 2. In this particular algorithm, the neighboring solution is randomly generated by choosing two different positions in the current solution (i.e., AGV picking schedule) and exchanging them as shown in Fig. 7.

When considering an exhaustive search, we define three levels of exhaustiveness as described below:

- **Level 1 exhaustive search**: This level of exhaustiveness maintains the picking sequence in the time chart of Fig. 3 (b).
- **Level 2 exhaustive search**: This level of exhaustiveness maintains the picking sequence in each AGV in the time chart of Fig. 3 (a).
- **Level 3 exhaustive search**: This level of exhaustiveness does not consider the original picking sequences of Fig. 3 (b) and generates arbitrary sequences. However, the items that each AGV must pick are kept constant.

These levels of exhaustiveness are illustrated in Figs. 8, 9, and 10. For these exhaustive searches, the combinational numbers, or the sizes of the solution spaces, may easily increase explosively even for small numbers of AGVs, pickers, and items in orders. For simplicity if the number of items of orders for each AGV is assumed to be equal, that is $Q_e$, the number of AGVs is $M$, and the number of pickers is $N$, then the sizes of the solution spaces are given by $N^{(MQ_e)}$ for Level 1, $(MQ_e)!N^{MQ_e}$ for Level 2, and $(MQ_e)!N^{MQ_e}$ for Level 3. The examples of solution sizes when $N = 2$ are shown in Table 2. Even for a small set of numbers of $M, N$, and $Q_e$, the size of the search space increases explosively, and the author decided to evaluate the proposed algorithm by comparing it with the solutions obtained by Level 1 and 2 exhaustive searches. The Level 3 exhaustive search requires an enormous amount of time and is computationally difficult even for small cases. A GA or simulated annealing algorithm may overcome this problem. In this paper, a simulated annealing algorithm was compared. Items for each AGV were as evenly distributed as possible. For instance, 6, 6, and 6 items when $Q = 18$, and 6, 7, and 7 items

![Fig. 7 Generation of neighboring solution in simulated annealing. Each box means the visiting place of each AGV, and the numbers in the boxes mean the AGV ID. This simulated annealing algorithm maintains the picking sequence for each AGV in the time chart of Fig. 3 (a).](image)

![Fig. 8 Level 1 exhaustive search.](image)

![Fig. 9 Level 2 exhaustive search.](image)

![Fig. 10 Level 3 exhaustive search.](image)
when \( Q = 20 \). Note that \( Q \) is the total number of items and not \( Q_e \) in the example shown in Table 2. The small orders were subsets of the real-world orders shown in Fig. 5. For the case \( Q = 20 \), for example, the selection of pickers was in accordance with the exhaustive generation of bit patterns consisting of 23 bits from 000000h to 7FFFFFh. The combination number increased to \( 2^{23} \approx 8 \times 10^6 \). The Level 1 exhaustive search took 10,851 s (about 3 h) by using a PC with an Intel core i7 processor operating at 3.6 GHz and Visual C++ 2019 release mode. The warehouse layout is shown in Fig. 6, the walking velocity of pickers was 1.0 m/s and the velocity of the AGVs was set to 0.7 m/s.

The Level 1 exhaustive search algorithm does not cover all the search space because of the dynamic nature of the time chart. Determining the demand event that should be handled next is another combinational optimization problem. For simplicity, in the Level 1 exhaustive search, pickers were assigned each time a demand event occurred on the time chart, which is dynamically updated depending on the picker’s arrival situation. If a picker is late, the AGV must wait until the picker arrives, causing a delay in the time chart. There may be a better solution if pickers are not assigned in accordance with the order of demand events on the time chart. This is why the Level 1 exhaustive search algorithm is referred to as the limited exhaustive search. Next, the Level 2 exhaustive search algorithm was applied, and its results are also summarized in Table 3.

Figure 11 shows the results of the evaluation for small orders. Because of the combinational explosion, the evaluation was limited to the case in which the total number of items was 20. The proposed algorithm provided a comparably good solution in the case of \( Q = 20 \). The proposed algorithm, the Level 1 exhaustive search, and the simulated annealing algorithm could provide solutions up to \( Q = 20 \). The Level 2 exhaustive search algorithm could provide solutions up to \( Q = 15 \).

For the simulated annealing algorithm, the number of iterations was 10,000 for all cases, and the cooling parameter \( r = 0.999 \) and the initial temperature \( T_0 = 100.0 \) s. The per cent gap between solutions (proposed algorithm v.s. best known) was calculated. The completion time of the best-known solutions is emphasized by shadows in Table 3. The proposed algorithm provided a comparably good solution such as in the cases of \( Q = 18 \) and 20. Even in the case of \( Q = 20 \), it required only 82 ms while the Level 1 exhaustive search took 10,815 s (about 3 h) and the simulated annealing algorithm took 20,614 s (about 5 h). The completion time obtained with the proposed algorithm was comparable for \( Q = 18 \) and 20 (within the range of about 2% additional completion time) and exceeded 10% for the cases below \( Q = 15 \) with the exception of the extremely small case of \( Q = 3 \).

### Table 3. Quality comparison of solutions for small orders.

| \( Q \): Number of items | 3   | 6   | 9   | 12  | 15  | 18  | 20  |
|--------------------------|-----|-----|-----|-----|-----|-----|-----|
| Completion time (s) (Level 1 exhaustive search) | 41.4| 63.0| 91.0| 102.6| 116.1| 124.0| 141.8|
| Completion time (s) (Level 2 exhaustive search) | 41.4| 61.2| 83.0| 100.0| N/A | N/A | N/A |
| Completion time (s) (simulated annealing) | 41.4| 61.2| 83.0| 95.0 | 110.0| 121.8| 130.0|
| Completion time (s) (proposed algorithm) | 42.1| 78.6| 92.0| 102.4| 123.1| 124.1| 132.8|
| % gap between solutions (proposed v.s. best known) | 1.7 | 28.4| 10.8| 17.8 | 18.8 | 1.9 | 2.1 |
| CPU time with Level 1 exhaustive search (s) | 0.09| 0.55| 4.97| 36.1 | 352.1 | 1,085.1| 10,815.1|
| CPU time with Level 2 exhaustive search (s) | 0.10| 0.11| 20.6| 2484.6| 149,001.1| N/A | N/A |
| CPU time with simulated annealing (s) | 0.31| 0.85| 6.2 | 57.3 | 506.3 | 4,643.8| 20,614.8|
| CPU time with proposed algorithm (s) | 0.024| 0.037| 0.041| 0.059 | 0.079 | 0.068 | 0.082 |

### Fig. 11. Picking completion time for small orders.
Fig. 12 Comparison of completion time of each AGV.

Table 4 Comparison of completion time in seconds of each AGV.

| AGV | case0 | case1 | case2 | case3 | case4 | case5 | case6 | case7 | case8 | case9 | ave. | stdev |
|-----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------|-------|
| AGV0 | 3856.4 | 3872.1 | 3859.7 | 3839.2 | 3879.2 | | | | | | | |
| AGV1 | 3865.0 | 3888.9 | 3921.6 | 3894.0 | 3994.1 | | | | | | | |
| AGV2 | 3851.1 | 3837.7 | 3859.2 | 3850.4 | 3829.5 | | | | | | | |
| AGV3 | 3850.2 | 3854.7 | 3881.6 | 3866.9 | 3843.4 | | | | | | | |
| AGV4 | 3822.9 | 3857.4 | 3866.1 | 3838.9 | 3817.9 | | | | | | | |
| ave. | 3852.6 | 3861.2 | 3866.3 | 3862.3 | 3854.7 | | | | | | | |
| stdev | 23.6 | 25.2 | 31.1 | 29.6 | 30.2 | | | | | | | |

Fig. 13 Comparison of walking time of each picker.

Table 5 Comparison of walking time in seconds of pickers.

| PICK0 | PICK1 | PICK2 | PICK3 |
|-------|-------|-------|-------|
| case0 | 2139.0 | 2107.0 | 2118.5 | 2090.0 |
| case1 | 2215.0 | 2157.0 | 2130.0 | 2069.0 |
| case2 | 2062.5 | 2127.0 | 2133.0 | 2058.0 |
| case3 | 2135.0 | 2066.0 | 2129.0 | 2114.0 |
| case4 | 2052.0 | 2071.0 | 2129.5 | 2076.5 |
| case5 | 2125.0 | 2085.0 | 2184.0 | 2146.5 |
| case6 | 2124.0 | 2126.0 | 2129.0 | 2103.0 |
| case7 | 2110.0 | 2073.0 | 2125.0 | 2084.0 |
| case8 | 2090.0 | 2110.0 | 2134.0 | 2033.0 |
| case9 | 2064.0 | 2086.0 | 2112.0 | 2032.0 |
| ave. (s) | 2120.7 | 2086.0 | 2132.3 | 2080.6 |
| stdev (s) | 58.4 | 29.7 | 19.4 | 35.5 |

Fig. 14 Comparison of wait time of each picker.

Table 6 Comparison of wait time in seconds of each picker.

| PICK0 | PICK1 | PICK2 | PICK3 |
|-------|-------|-------|-------|
| case0 | 45.7 | 51.5 | 46.5 | 44.2 |
| case1 | 44.1 | 49.6 | 68.9 | 40.0 |
| case2 | 40.1 | 43.6 | 63.8 | 45.5 |
| case3 | 41.9 | 49.2 | 46.4 | 46.0 |
| case4 | 70.9 | 55.5 | 38.8 | 66.2 |
| case5 | 48.6 | 52.6 | 50.6 | 35.6 |
| case6 | 42.4 | 39.1 | 47.0 | 41.3 |
| case7 | 35.1 | 69.7 | 59.9 | 47.3 |
| case8 | 45.9 | 48.3 | 60.7 | 30.6 |
| case9 | 38.3 | 47.3 | 44.8 | 57.6 |
| ave. (s) | 45.3 | 50.6 | 52.7 | 45.5 |
| stdev (s) | 9.8 | 8.1 | 9.8 | 10.3 |

timum solutions were not available because the combination number was too large. The author argues that sub-optimum solutions or feasible solutions can be obtained in a very short time. Obtaining feasible solutions even though they are not exactly optimum in a short computation time is much more desirable than obtaining an exact solution by waiting for hundreds of years. The walking velocity of pickers was 1.0 m/s and the velocity of AGVs was set to 0.7 m/s. Figure 12 shows the completion time of each AGV for these ten cases. Table 4 lists the results. There was very little deviation in these cases, i.e., tens of seconds, which is quite small compared to the average completion time of 3,855 s. The ratio of deviation to average completion time was less than 1%. Figure 13 shows the results of the walking time of pickers for the ten random cases. Again, very little deviation in these cases was observed, although no explicit strategy to equalize the walking time of pickers was introduced. Table 5 lists the walking time of pickers. The deviation was also tens of seconds, which is quite small compared to the average picker walking time of about 2,100 s. The ratio of deviation in the average picker walking time was about 3%. Figure 14 shows the wait time of each picker. This time, deviation was not very small; however, the magnitude was almost 1 min and negligible compared with the walking time of about 2,100 s (i.e., more than 30 min). Table 6 lists the wait time of pickers for the ten cases, and Fig. 15 shows the results of the time required in searching for items by each picker. The author assumed that each picker needs 5 s to find an item on a shelf and put it on an AGV. This value was determined by careful observation of the pickers at this warehouse. This value may vary in actual situations. In this situation, however, it was simply set to a fixed value. The variance in searching time does not significantly affect the total completion time because the magnitude of the standard deviation of searching time is the order of seconds while the completion time is the order of thousand seconds. Table 7 lists the time required in searching for items on a shelf. The deviation was tens of seconds, which is quite
Fig. 15 Comparison of searching time for items by each picker.

Table 7 Comparison of searching times in seconds for items by each picker.

| case0 | PICK0 | PICK1 | PICK2 | PICK3 |
|-------|-------|-------|-------|-------|
| 1685.0 | 1710.0 | 1695.0 | 1735.0 |
| 1635.0 | 1705.0 | 1690.0 | 1795.0 |
| 1740.0 | 1675.0 | 1660.0 | 1750.0 |
| 1680.0 | 1735.0 | 1695.0 | 1750.0 |
| 1725.0 | 1720.0 | 1680.0 | 1700.0 |
| 1640.0 | 1770.0 | 1685.0 | 1730.0 |
| 1705.0 | 1705.0 | 1695.0 | 1720.0 |
| 1715.0 | 1710.0 | 1675.0 | 1725.0 |
| 1700.0 | 1680.0 | 1650.0 | 1795.0 |
| 1720.0 | 1690.0 | 1670.0 | 1745.0 |

ave. (s) 1694.5
stdev (s) 35.0

Fig. 16 Completion time for different number of pickers.

3.3 Computational Time Evaluation

Computational time was evaluated by running several settings of the problems. The number of pickers varied from one to five while the number of AGVs was five. Ten cases of 100 random batched orders were simulated. The CPU time was about 100 ms for all cases and not significantly affected by the number of pickers. This is because the proposed algorithm does not depend much on the number of pickers. In Algorithm 1, the CPU time is consumed mainly in the while loop, which is executed for the number of AGVs. In the while loop, the processing time for updating the time chart is dominant and consumes almost constant time. The updating of the time chart by using the quick sort algorithm is done every time in the while loop. This is the reason why the proposed algorithm is almost linear to the number of AGVs. Figure 17 shows the results of CPU time when the number of AGVs increased. The coefficient of determination i.e., $R^2$, was 0.9988 and was almost linear to the number of AGVs. Therefore, the proposed algorithm is sufficiently fast to solve real-world problems of this size in less than 1 s.

4. Conclusion

The author proposed a fast and feasible algorithm for generating a co-operation schedule of AGVs and pickers. The algorithm introduces two heuristic rules to reduce the size of the search space. The proposed algorithm was first compared with two levels of exhaustive search algorithms for cases with items less than 20, three AGVs, and two pickers. The proposed algorithm provided a comparably good solution such as in the case of $Q = 18$ and 20. Even in the case of $Q = 20$, it required only 82 ms while the Level 1 exhaustive search took 10,815 s (about 3 h), and the simulated annealing algorithm took 20,614 s (about 5 h). The completion time obtained with the proposed algorithm was comparable for $Q = 18$ and 20 (within the range of about 2% additional completion time) and exceeded 10% for the cases below $Q = 15$ with the exception of extremely small case of $Q = 3$.

As a real-world case, a set of problems consisting of 10 cases of 100 random batched orders were simulated. The author evaluated the performance of the proposed algorithm by changing

![Fig. 15 Comparison of searching time for items by each picker.](image1)

![Fig. 16 Completion time for different number of pickers.](image2)

![Fig. 17 CPU time of the proposed algorithm when the number of AGVs increased.](image3)
the number of AGVs and pickers. It was observed that the standard deviation of the completion time was small, and fairly equalized completion time among AGVs was observed. The ratio of deviation to average completion time was less than 1%. Walking time deviation among pickers was also small. The ratio of deviation in the average picker walking time was about 3%. Although the objective function is the task completion time and does not explicitly evaluate the evenness of the time, the proposed algorithm worked to equalize the time implicitly.

The CPU time of the proposed algorithm is almost linear to the number of AGVs in a warehouse, and the CPU time is not significantly affected by the number of pickers.

For real-world cases, the proposed algorithm is a promising algorithm which can generate sub-optimal schedules for AGVs and pickers with almost minimum task completion time in a very short time. This will enable co-operative picking with AGVs in warehouses.

References

[1] T. Tamura and T. Yokota: Fast optimization algorithm for conveyance orders and paths of conveyance robots in warehouses, Proc. 6th International Conference on Transportation and Logistics, A-24, Sept. 2016.

[2] T. Yokota: Min-max-strategy-based optimum co-operative picking with AGVs in warehouse, Proc. SICE Annual Conference 2019, pp. 236–242, 2019.

[3] Y.-C. Ho, T.-S. Su, and Z.-B. Shi: Order-batching methods for an order-picking warehouse with two cross aisles, Computers and Industrial Engineering, Vol. 55, No. 2, pp. 321–347, 2008.

[4] N. Gadrmann and S. van de Velde: Order batching to minimize total travel time in a parallel-aisle warehouse, IIE Transactions, Vol. 37, No. 1, pp. 63–75, 2005.

[5] S. Henn, S. Koch, K.F. Doerner, C. Strauss, and G. Wischner: Metaheuristics for the order batching problem in manual order picking systems, Business Research, Vol. 3, No. 1, pp. 82–105, 2010.

[6] A.J.R.M. (Noud) Gademan, J.P. van den Berg, and H.H. van der Hoff: An order batching algorithm for wave picking in a parallel-aisle warehouse, IIE Transactions, Vol. 33, No. 5, pp. 385–398, 2001.

[7] C.-C. Janea and Y.-W. Laih: A clustering algorithm for item assignment in a synchronized zone order picking system, European Journal of Operational Research, Vol. 166, No. 2, pp. 489–496, 2005.

[8] J.-T. Li and H.-J. Liu: Design optimization of Amazon robotics, Automation, Control and Intelligent Systems, Vol. 4, No. 2, pp. 48–52, 2016.

[9] K.C.T. Vivaldini, J.P.M. Galdames, T.B. Pasqual, R.M. Sobral, R.C. Araújo, M. Becker, and G. Caurin: Automatic routing system for intelligent warehouses, IEEE International Conference on Robotics and Automation, Vol. 1, pp. 1–6, 2010.

[10] E. Gawrilow, E. Köhler, R.H. Möhring, and B. Stenzel: Dynamic routing of automated guided vehicles in real-time, Mathematics-Key Technology for the Future, pp. 165–177, Springer, 2008.

[11] K.C.T. Vivaldini, M. Becker, and G.A.P. Caurin: Automatic routing of forklift robots in warehouse applications, Proc. 20th International Congress of Mechanical Engineering, COBO09-0334, 2009.

[12] J.I.U. Rubrico, T. Higashi, H. Tamura, and J. Otam: Online rescheduling of multiple picking agents for warehouse management, Robotics and Computer-Integrated Manufacturing, Vol. 27, No. 1, pp. 62–71, 2011.

[13] P.R. Wurman, R. D’Andrea, and M. Mounts: Coordinating hundreds of cooperative, autonomous vehicles in warehouses, Proc. 19th National Conference on Innovative Applications of Artificial Intelligence, Vol. 2, pp. 1752–1759, 2007.

[14] P.R. Wurman, R. D’Andrea, and M. Mounts: Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Magazine, Vol. 29, No. 1, pp. 9–20, 2008.

Takayoshi Yokota (Member)

He is a professor of Tottori University since April 2012. He received his B.S., M.S., and Ph. D. from Tokyo Institute of Technology in 1979, 1981, and 1984. He joined the Hitachi Research Laboratory of Hitachi Ltd. in 1984. He was a professor at the faculty of Kyoto University in April 2009. He is a professor of the Graduate School of Engineering at Tottori University since April 2012. His current research interests include analysis, modeling, and spatio-temporal information processing of intelligent transport systems (ITS) related data. He is also a member of IEEE, IEICE, and IPSJ.