An effective iterated greedy algorithm for multi-AGVs dispatching problem in material distribution

Xujin Zhang¹,a, Hongyan Sang¹,b*, Zhongkai Li¹,c, Junqing Li¹,d, Hengwei Guo¹,e
¹School of Computer Science, Liaocheng University, Shandong, China
*aemail: 1435152118@qq.com, bemail: 493056738@qq.com,cemail: lijunqing@lcu-cs.com, demail: 920563256@qq.com
*Corresponding author: beemail:sanghongyan@lcu-cs.com

Abstract: In this paper, the dispatching problem for multi-AGVs in the matrix manufacturing workshop from the practical factories is optimized. The goal is to reduce the transportation cost, including the cost of AGVs, distance cost and time cost. In order to solve the problem, we develop a mixed integer linear programming model. Then, an effective iterative greedy algorithm (IG) is proposed. The nearest-neighbor-based heuristic is used to generate high-quality initial solutions. An effective gradient unload method is designed. For the local search stage, three efficient operators are proposed. Finally, examples for experiment come from Foxconn electronic company. Comprehensive evaluation shows that the performance of the proposed IG is significantly better than some existing algorithms in the literature.

1. Introduction
Automatic guided vehicle (AGV) is an unmanned vehicle controlled by computer [1-2]. In recent years, AGV has been widely used in various production workshops [3]. In the matrix workshop, some AGVs transport raw materials to the machines. Optimizing AGV dispatching problem (AGVDP) can reduce the transportation cost of AGVs and enhance the workshop production efficiency. AGVDP has important research significance [4-5].

According to the literature review, there are few studies on AGVDP. At present, many factories adopt the first request first service (FCFS). A discrete artificial bee colony algorithm and a mixed integer linear programming model are proposed by Zou et.al. [6].

Compared with other algorithms, iterative greedy (IG) algorithm is an iterative search method. It is relatively simple, efficient, and easy to understand [7]. It has created state-of-the-art possibilities for many different problems, and used more and more in scheduling [8]. We propose an effective IG algorithm for AGVDP to reduce the transportation cost in this paper. All the experiments are based on real data from real factory.

The rest of the paper is organized as follows. In the second section, we describe the problem, give mixed integer linear programming model, and design an effective loading and unloading strategy. The third section presents the proposed IG algorithm in detail. The fourth section is the experiments. The fifth section gives summarizes.

2. Description of the Problem
2.1. **Introduction to the problem**

There are many workstations in the matrix production workshop. A workstation consists of buffer pool and computer numerical control (CNC) machines. The layout of the workshop is shown in Fig. 1. In the warehouse, some AGVs are waiting for instructions from the machine control system. If the inventory in the buffer reaches the specified lower limit, the machine will send a task request. Then, AGVs must transport the materials from the warehouse to those machines buffer pool before those latest delivery time. After the AGVs completes all tasks, they will return to the warehouse. For each AGV, the total weight of loaded raw materials can’t exceed AGV carrying capacity. In this paper, we divide a production cycle into several continuous stages (calculation stage and transportation stage). This means that the tasks collected in one cycle are transported in the next cycle.

![Figure 1 The matrix manufacturing workshop](image)

2.2. **Problem formulation**

The following assumptions are made:

1. All the machines can run normally without shutdown and failure.
2. Production materials are stored in the warehouse.
3. The starting and ending points of AGVs are in the warehouse.
4. The specifications of AGVs are the same.
5. Each machine can only be served by one AGV once in one stage.

The notation used is presented below.

- $i, j$: The identification of tasks.
- $p_i$: Location of task $i$.
- $x_i$: $p_i$ on X-axis.
- $y_i$: $p_i$ on Y-axis.
- $n$: Total number of tasks requested.
- $n'$: The maximum number of tasks which AGV can load.
- $k$: Current AGV (or AGV route).
- $k'$: Expected number of AGVs.
- $q_i$: Material demand of task $i$.
- $T_i$: The request time of task $i$.
- $T_i'$: The latest completion time of task $i$.
- $t_u$: Time for unloading raw materials.
- $t_m$: Unit time of raw material consumption.
- $S$: The maximum value of storing material in the buffer.
- $S_i'$: The value of the remaining material in the buffer when requested.
\( g \): Weight per unit of raw material.
\( c_g \): Unit cost of AGV driving distance.
\( c_a \): Cost of AGV running once.
\( c_{el} \): Penalty for early delivery within 100 seconds.
\( c_{en} \): Penalty for delivery over 100 seconds ahead of time.

Decision variables:
\( x_{ijk} \): If there is AGV route between task \( i \) and \( j \), it is 1, otherwise 0.
\( y_{ijk} \): If the early arrival time is less than 100 seconds, it is 1, otherwise 0.
\( m \): Total number of AGVs dispatched.
\( T_i \): Time when AGV arrives at the task \( i \).

Let's assume that the warehouse is the origin of the coordinates. \( E = \{ (i, j) \mid i, j \in V, i \neq j \} \) represents the set of all paths between tasks \( i \) and \( j \). The travel distance and travel time between two machines can be expressed as follows:

\[
d_{ij} = |x_i - x_j| + |y_i - y_j| \\
t_{ij} = d_{ij} / v
\]

The time of AGV from task (warehouse) \( i \) to \( j \) can be expressed as follows:

\[
T'_{ij} = T'_{ij} + t_i, \forall i \in V, j \in V \setminus \{0\}, i \neq j
\]

If \( n \) tasks make requests in this cycle, the number of AGVs that we expect can be expressed as:

\[
k' = \lceil n/n' \rceil, n' = 12
\]

AGVDP is optimized to get the minimum transportation cost from three aspects: shorten the total driving distance of AGV, ensure the delivery within the delivery time (reduce the time difference) and reduce the total amount of AGV. The formula is as follows.

\[
\min F(i, j, k) = c_t \sum_{k=1}^{m} \sum_{j=0}^{n} x_{ijk} d_{ij} + c_a \sum_{k=1}^{m} \sum_{j=0}^{n} x_{ijk} a_{ij} + c_{el} \sum_{k=1}^{m} \sum_{j=0}^{n} x_{ijk} y_{ijk} (T'_{ij} - T'_{ij}) + c_{en} \sum_{k=1}^{m} \sum_{j=0}^{n} x_{ijk} (1 - y_{ijk}) (T'_{ij} - T'_{ij})
\]

2.3. A cost saving gradient loading method

The starting point of each AGV is the warehouse, the closer the machine is to the warehouse, the higher the frequency of AGV passing through the machine. So, the easier these machines are to be served by different AGVs. In order to effectively reduce the transportation cost, we design a new gradient loading method. Through the actual investigation of the workshop, the appropriate parameters are set. When AGV arrives at task \( j \), the unloading amount of \( j \) can be expressed as follows:

According to the minimum bound of the buffer when a request is made by a task, the loading of the task request is satisfied by 60% enough to meet the needs of the next production cycle. If the distance between task \( j \) and depot is less than 120:

\[
q_j = \left( S - S' \right) \times 60\% + \left[ \left( T'_{ij} - T_{ij} \right)/t_u \right] \times g, \forall j \in V \setminus \{0\}
\]

Else when the distance more than 120:

\[
q_j = \left( S - S' \right) + \left[ \left( T'_{ij} - T_{ij} \right)/t_u \right] \times g, \forall j \in V \setminus \{0\}
\]

3. The Proposed Effective Iterated Greedy Algorithm

For solving the AGVDP, we propose an effective IG algorithm. The algorithm including solution representation, initial solution, destruction stage, construction stage, local search stage and acceptance stage [9].

3.1. Solution Representation
There are $n$ tasks, and a total of $m$ AGVs are required. The length of the solution is $(n + m - 1)$. Divide all tasks into $m$ AGV routes by $(m - 1)$ 0s. For example, there are 8 tasks and 3 AGVs. The order of transportation tasks of the first AGV is (2, 6, 9). The second AGV is (5, 3, 7, 1). The third AGV is (4). Then, the solution can be expressed as (2,6,9,0,5,3,7,1,0,4).

3.2. Initial Solution
Generating high-quality initial solution is very important to improve the performance of IG. We use a nearest-neighbor-based heuristic (NNH) algorithm based on the shortest distance [6]. Under the premise of satisfying the specified constraints, first select the task closest to the warehouse, and then select the task closest to the current task as the next delivery task, and so on. The specific procedure is shown in Fig. 2.

\begin{verbatim}
Procedure NNH
1: Let the AGV route $R = \emptyset$ and task $j = 0$
2: while $U$ is not empty do
3: for $i = 1$ to $n$
4: find the nearest task $i$ from $j$ in $U$
5: endfor
6: test to append task $i$ to route $R$
7: if $R$ meets the capacity and time constraints then
8: let task $j = i$ and delete $i$ from $U$
9: else
10: append $R$ to solution $\tau$, and empty $R$
11: add 0 at the end of $\tau$, and let $j = 0$
12: endif
13: endwhile
14: if route $R$ is unempty then
15: append route $R$ to solution $\tau$
16: endif
17: return $\tau$
\end{verbatim}

Figure 2 The procedure of NNH algorithm

3.3. Destruction
In the destruction stage, $\tau^d$ denotes the deleted tasks, $\tau^l$ denotes the remaining tasks. $d$ is the number of tasks which are needed deleted in the destruction stage. Randomly select a task on an AGV, then delete the task and the remaining tasks are saved in $\tau^d$. This process is performed $d$ times, save the remaining tasks in $\tau^l$.

3.4. Construction
The solution is divided into $\tau^d$ and $\tau^l$ in the destruction stage, these two parts will recombine into one solution in the construction stage. Take a task from $\tau^d$ and tentatively insert it into each position in $\tau^l$. On the premise of meeting all constraints, the location with the least transportation cost is selected to insert. Finally, the successfully inserted task is removed from $\tau^d$. The above steps are repeat until $\tau^d$ is empty.

3.5. Local search
In the local search phase, three efficient operators are proposed. The operators are as follows:

(1) Merge operation
There are $k'$ AGVs. First, sort the AGVs according to the number of loading tasks. Then, take the tasks on the last AGV and insert them into the AGV with more carrying capacity on the premise of meeting the time and capacity constraints, and repeat this process until the number of AGVs acceptable to the factory is met or cannot be merged again.

(2) Task exchange operation in the same AGV
Randomly selects an AGV from the solutions generated in the construction stage. Then two tasks are selected randomly to exchange on the premise of meeting the constraints.

(3) Task exchange operation on different AGVs
Two tasks are selected randomly from different AGVs in the solution $\tau$, and exchanged on the premise of meeting the constraints.

3.6. Acceptance criteria
In the final acceptance stage, a simple simulated annealing acceptance criterion is used to avoid IG falling into local optimum. The temperature $T$ of the criterion is shown in the following formula:

$$T = \left( \frac{\sum_{i=1}^{n} C_i^{\text{call}}}{5n} \right) / 2$$

$\sum_{i=1}^{n} C_i^{\text{call}}$ represents the total time of the call requests by $n$ tasks.

3.7. The complete procedure of IG algorithm
The pseudocode of the proposed IG is shown in Fig. 3.

```
Procedure IG
1: parameters: $d$
2: $\tau$ = The initial solution produced by NNH
3: Update best solution $\tau^{\text{best}}$
4: while The time is not up do
5: $\tau'$ and $\tau''$ = Destruction ($\tau$, $d$)
6: $\tau'$ = Construction($\tau'$, $\tau''$)
7: $\tau'$ = Localsearch($\tau'$)
8: $\tau$ = Acceptance criteria($\tau$, $\tau'$)
9: endwhile
10: return $\tau^{\text{best}}$
```

Figure 3 The presented IG algorithm

4. Result analysis
In this section, experiments are performed to test the proposed algorithm. The parameter calibration is carried out. A lot of practical examples from the actual factory (Foxconn Technology Group) are used for calibration and testing. Then, the proposed algorithm is compared to other algorithms in detail.

We use C++ to code, and use Visual Studio 2019 to run in windows 10 with 4GB memory. To make a fair comparison, the same stopping criteria for the above algorithm is set. The running time of CPU is preset as $\Delta t=5$s. The results use the relative percentage deviation (RPD). The formula is as follows:

$$\text{RPD} = \frac{C_i - C_{\text{best}}}{C_{\text{best}}}$$

$C_i$ represents the transportation cost in the current solution, and $C_{\text{best}}$ represents the minimum transport cost. The lower the RPD, the better the performance of the algorithm [10].

4.1. Experimental calibration
In the destruction stage of IG algorithm, the parameter $d$ needs to be calibrated. By literature review and continuous experiments, $d$ is determined at five levels: [3,4,5,6,7]. To get the optimal results, we have carried out a full algorithm calibration experiment. Ten different real examples are selected, and each example is run 10 times. Therefore, 400 times of this calibration experiment have carried out.

A multi-factor analysis of variance (ANOVA) technology is used to analyze calibration results, which has been widely used in the scheduling literature [11]. The Honest Significant Difference (HSD) experimental data is used for analysis at the 95% confidence level. According to the Fig. 4, we set the parameter $d$ is 5.
The IG algorithm has the best results in all effectiveness of the proposed IG algorithm.

Through the analysis of the results, it can conclude that the performance of IG algorithm is significantly better than other algorithms. IG algorithm can find the solution of minimum transportation cost for each problem. The results show that the IG algorithm with NNH strategy, cost saving loading strategy is better than DABC, HFOA, HGA and IHS algorithm in reducing the transportation cost of AGVDP.
5. Conclusion
At present, AGVDP has attracted extensive attention of researchers [14]. This paper aims to minimize transportation costs in AGVDP. We establish the mixed integer linear programming model, design an effective loading strategy to reduce the cost. The proposed strategy is applied to IG algorithm. According to the characteristics of the research problem, NNH is used to produce a high-quality solution, and the solution is optimized through the stages of destruction, reconstruction and local search. According to the experimental results, the effectiveness of the proposed IG algorithm is significantly better than other algorithms in dealing with AGVDP.

In the future, the proposed algorithm can be further improved to solve more complex problems. For example, the routing planning and AGVDP collision problem can be considered in AGV dispatching.

Acknowledgement
This work was supported by the National Science Foundation of China under Grant (61803192, 61773192), Shandong province colleges and universities youth innovation talent introduction and education program, Research fund project of Liaocheng university under Grant NO.318011922.

References
[1] Fazlollahtabar, H., Hassanli, S. (2018) Applied Intelligence, 48(2): 482-498.
[2] Fransen, K. J. C., Eekelen, J. A. W. M., Pogromsky, A., Boon, M. A. A., and Adan, I. J. B. F. (2020) A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems. Computers & Operations Research, 123.
[3] Patricio, R., and Mendes, A. (2020) Consumption Patterns and the Advent of Automated Guided Vehicles, and the Trends for Automated Guided Vehicles. Current Robotics Reports, 1(3): 145-149.
[4] Bechtsis, D., Tsolakis, N., Vlachos, D., and Iakovou, E. (2017) Sustainable supply chain management in the digitalisation era: The impact of Automated Guided Vehicles. Journal of Cleaner Production, 142: 3970-3984.
[5] Demesure, G., Defoort, M., Bekrar, A., Trentesaux, D., and Djemaï, M. (2015) Navigation Scheme with Priority-Based Scheduling of Mobile Agents: Application to AGV-Based Flexible Manufacturing System. Journal of Intelligent & Robotic Systems, 82(3): 495-512.
[6] Zou, W.-Q., Pan, Q.-K., Meng, T., Gao, L., and Wang, Y.-L. (2020) An effective discrete artificial bee colony algorithm for multi-AGVs dispatching problem in a matrix manufacturing workshop. Expert Systems with Applications, 161.
[7] Ruiz, R., Pan, Q.-K., and Naderi, B. (2019) Iterated Greedy methods for the distributed permutation flowshop scheduling problem. Omega, 83: 213-222.
[8] Mao J.-Y., Pan Q.-K., Zhong-H., Gao-L. (2020) An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with
preventive maintenance. Expert Systems with Applications, 114495.

[9] Ribas, I., Companys, R., and Tort-Martorell X. (2011) An iterated greedy algorithm for the flowshop scheduling problem with blocking. Omega, 39(3):293-301.

[10] Huang, J.-P., Pan, Q.-K., and Gao, L. (2020) An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. Swarm and Evolutionary Computation, 59.

[11] Sang, H.-Y., Duan, P.-Y. (2016) A Discrete Invasive Weed Optimization Algorithm for the No-Wait Lot-Streaming Flow Shop Scheduling Problems. International Conference on Intelligent Computing. Springer International Publishing, 517-526.

[12] Li, G., Zeng, B., Liao, W., Li, X., and Gao, L. (2018) A new AGV scheduling algorithm based on harmony search for material transfer in a real-world manufacturing system. Advances in Mechanical Engineering, 10(3), 2018.

[13] Euchi, J., and Sadok, A. (2021) Hybrid genetic-sweep algorithm to solve the vehicle routing problem with drones. Physical Communication, 44.

[14] Wang, C. L., and Li, S. W. (2018) Hybrid fruit fly optimization algorithm for solving multi-compartment vehicle routing problem in intelligent logistics. Advances in Production Engineering & Management, 13(4):466-478.