Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

A SIMULATION OPTIMIZATION METHOD FOR SCHEDULING AUTOMATED GUIDED VEHICLES IN A STOCHASTIC WAREHOUSE MANAGEMENT SYSTEM

Gongbo Zhang
Xiaotian Liu
Yijie Peng

Guanghua School of Management
Peking University
5 Yiheyuan Road
Beijing 100871, P. R. CHINA

ABSTRACT

We consider the problem of scheduling automated guided vehicles (AGVs) in a stochastic warehouse management system. This problem was studied in the Case Study Competition of the 2022 Winter Simulation Conference. We propose a simulation optimization method that simultaneously optimizes dispatching and route planning for AGVs to enhance the system performance. Experimental results on two warehouse system simulation scenarios demonstrate that the proposed method outperforms the default method.

1 INTRODUCTION

In a stochastic warehouse management system, stock-keeping units (SKUs) are stored on storage racks, and orders arrive randomly at the warehouse. To meet the surging logistics demand, automated guided vehicles (AGVs) are commonly used in intelligent warehouses as the primary means of transportation, fulfilling the horizontal movement of SKUs between racks and delivery locations. However, challenges arise in ensuring the efficiency of AGVs due to the high volume of orders and limited space for AGV movement. AGVs are also widely used in manufacturing plants and automated container terminals (Corréa et al. 2007; Zhen et al. 2020; Hu et al. 2023). This study aims to develop dispatching and route planning algorithms for a stochastic warehouse management system. These algorithms have tuning parameters that generate multiple designs of the system, each with an unknown mean performance that can be estimated using Monte Carlo simulation. A simulation optimization method is proposed to allocate a fixed simulation budget for finding the optimal design.

The literature on AGV scheduling is extensive. See Vis (2006), Fazlollahtabar and Saidi-Mehrabad (2015), and De Ryck et al. (2020) for thorough overviews. The scheduling of AGVs typically involves dispatching and route planning, where the former assigns orders to AGVs under resource constraints, and the latter determines the travel paths to fulfill the movement of SKUs specified by corresponding orders. These two problems were initially studied separately, but recent works have simultaneously considered them to improve the overall system efficiency (Corréa et al. 2007; Nishi et al. 2011). Beyond these two problems, recent advances in AGV scheduling include determining AGV fleet size (Vis et al. 2001), optimizing vehicle speed (Adamo et al. 2018), and managing vehicle battery (Kabir and Suzuki 2019). However, in this work, our focuses are on dispatching and route planning problems.

The AGV dispatching problem typically assumes that both the number of orders and AGV fleet size are known. Dispatching methods can be categorized into static and dynamic paradigms, where the latter reassigns orders to better-suited AGVs if available. If the number of orders and AGV fleet size are relatively small, the dispatching problem formulated as a mixed-integer linear programming problem can be solved exactly using a branch and bound algorithm or other tree-based algorithms (De Ryck et al. 2020). However, for larger numbers of orders and AGV fleet size, the dispatching problem becomes a constrained NP-hard optimization problem that can only be solved using heuristic methods such as neighborhood search and genetic algorithms (Lee et al. 2010). The AGV routing problem is similar to the traditional vehicle routing problem (VRP), but it often assumes that AGVs travel at a constant speed within a limited grid space and try to avoid obstacles, vehicle collisions and deadlocks. Existing literature on collision-free routing can be categorized into static, time-window based (Adamo et al. 2018), and dynamic paradigms. In the static paradigm, an AGV occupies an entire path before passing through it, and sticking at a certain square unit for an AGV increases the makespan of some orders. In the dynamic paradigm, any path can be adjusted based on time-varying traffic conditions, e.g., finding the shortest path for each AGV and then adjusting it to find an alternative path when a collision occurs. Other dynamic collision-free routing algorithms include generating k-shortest paths (Lee et al. 1998), using a graph-representation method (Fransen et al. 2020), or adopting a space-time routing framework (An et al. 2020). A recent advance in the AGV routing problem includes reinforcement learning methods. For example, Hu et al. (2020) propose a real-time scheduling method based on deep reinforcement learning, and Hu et al. (2023) use a multi-agent reinforcement learning method to solve an integer programming model. In this work, our focus is on finding desirable paths for all AGVs using a constrained tree-based search method that avoids collisions between AGVs.

The stochastic warehouse management system is modeled as a discrete event dynamic system (DEDS). There are tuning parameters that can take on various values and affect system performance. We consider a simulation budget allocation problem for optimizing parameters. Simulation budget allocation is actively studied for optimizing complex DEDSs that are computationally intensive to simulate. The goal of a classic simulation budget allocation problem is to select the best design of a stochastic model with the largest (or smallest) mean from a finite number of designs with unknown means. The mean of a random output of the model often does not have an analytical form but can be estimated by simulation. However, simulation is typically expensive, so a limited simulation budget needs to be intelligently allocated to improve sampling efficiency. Peng and Zhang (2022) conduct comprehensive numerical comparisons of fixed-budget sampling policies, including optimal computing budget allocation (OCBA) (Chen et al. 2000), expected improvement (EI) (Ryzhov 2016), knowledge gradient (KG) (Frazier et al. 2008), and asymptotically optimal allocation procedure (AOAP) (Peng et al. 2018). However, as the number of parameters and their alternative values increases, the total number of combinations increases rapidly, posing a significant challenge to the traditional implementation of these policies. In this work, we use parallel computing to increase computational efficiency.

In this work, we adopt a discrete event simulation model used in the Case Study Competition of the 2022 Winter Simulation Conference (2022 WSC). We propose a rule-based dynamic dispatching algorithm and a route planning algorithm that uses breadth-first search (BFS) to generate collision-free paths in a search space with constraints. To further optimize system performance, we study a simulation budget allocation problem for finding the best tuning parameter combination in the dispatching and route planning algorithms. Each combination represents a system design, and we use an efficient sampling policy called AOAP to intelligently allocate the simulation budget. Our budget allocation scheme is implemented in a parallel computing environment with a central processing unit (CPU) that contains multiple cores. The designs are distributed among different processors to reduce execution time. We demonstrate the efficiency of our proposed method through numerical experiments on two warehouse management system simulation scenarios. The rest of the paper is organised as follows. Section 2 describes the discrete-event simulation model. The dispatching and route planning algorithms are proposed in Section 3. Section 4 presents the numerical results. The last section concludes the paper.

2 PROBLEM DESCRIPTION

In this section, we outline the discrete-event simulation model adopted in our work. The AGV traffic network is represented as a grid consisting of fixed-size traversable square units, where different warehouse layouts can be created by blocking certain units. Figure 1 shows a configuration of the grid. Orders arrive at the system following a Poisson process, and the picking rack and delivery location of an order are specified upon arrival. The randomness in the simulation model arises from the random arrival of orders. AGVs move along adjacent square units in the limited grid space to pick or deliver SKUs or park at berths if no order is assigned to them.

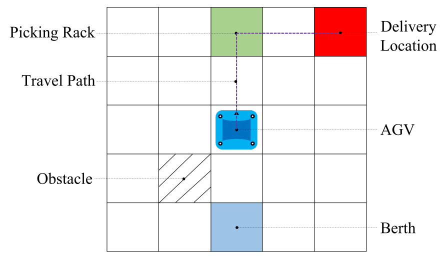


Figure 1: Grid-based AGV traffic network.

An AGV parks at its berth before it is matched with an order. Then it travels to the picking rack and picks the SKUs. After loading, it travels to the delivery location and delivers the SKUs. If the AGV is assigned an order before unloading, it will travel to pick up the next order. If the AGV has no more order assigned, it will travel back to its berth and wait for new orders. The speed of an AGV is 1 square unit per second, and the loading and unloading durations are 3 seconds and 4 seconds, respectively. There are four statuses for an AGV, including: (1) picking–when it is moving to the picking rack, (2) delivering–when it has finished loading SKUs and is moving to the delivery location, (3) parking–when it has finished unloading and has no order assigned, and (4) idle–when it is parking at its berth or has finished unloading but still has new assigned orders. The following assumptions are introduced in the simulation model:

- Orders are independent of each other.
- Each square unit can hold only one AGV at a time. An AGV can reserve multiple square units for route planning, but it can only be located in one unit at a time.
- Each order is assigned to one AGV, which can be assigned multiple orders. An AGV can transfer SKUs specified by only one order at a time.
- All AGVs have the same function, and their maximum battery capacity ensures the energy consumption of the assigned orders.
- The simulation model ignores the acceleration and deceleration of AGVs during movement.
- Each AGV has its own berth and must return to it when its status is ‘parking’.

The AGV dispatching module is invoked when the system receives an order or the status of an AGV becomes ‘parking’. All available orders in the system, except those being transported by the AGVs, can be assigned to any AGVs except those in the ‘parking’ status. That is, an order may be reassigned if the AGV has not started moving to pick up the SKUs. After the assigned AGV reaches the picking rack, the SKUs will start loading and subsequently be transported to the delivery location, where the SKUs will be unloaded from the vehicle.

The AGV routing module is invoked after dispatching, loading, and unloading. First, the AGV obtains its full path from its current location to the destination location, e.g., from the picking rack to the delivery location. Next, the partial route request module is invoked to find the first partial route, which is a section of the full path adjacent to the current location. If all the square units in the requested partial route are unoccupied by other AGVs, the AGV starts to move; otherwise, it waits for the partial route to be released at its current location. If multiple AGVs are waiting to pass through the same partial route, the one with the earliest request time has priority. After a random time, which follows a uniform distribution over the duration of the first partial route, the partial route request module is again invoked to find the next partial route. The current partial route is released after the AGV reaches its last square unit. The requesting process repeats until the AGV reaches its final destination. Once the AGV completes its full path, it performs loading, unloading, or parking, depending on the specific operation requirements.

A metric to evaluate the performance of the warehouse management system is the adjusted average order cycle time (ACT), i.e., $\mathrm{ACT} = \mathbb{E}[(T_f^{\widetilde{c}}(N_f) + \widetilde{c}N_u)/N]$, where $T_f^{\widetilde{c}}(N_f)$ is the total cycle time for finished orders, N_f is the number of finished orders, N_u is the number of unfinished orders, $N = N_u + N_f$ is the total number of generated orders, and $\widetilde{c}$ is a penalty parameter for unfinished orders due to deadlocks. The simulation model runs for 1 hour, with the penalty time $\widetilde{c}$ set at $\frac{1}{5}$ of the time it takes for an AGV to travel the entire grid. Our objective is to develop dispatching and route planning algorithms for the corresponding modules of the simulation model to minimize the ACT in a given warehouse management system simulation scenario.

3 A NEW SCHEDULING METHOD

In this section, we first propose dispatching and collision-free route planning algorithms for the warehouse system. Then, we adopt a simulation budget allocation policy to further improve the system performance. The structure of the simulation model and the input information for each module affect the design of these algorithms.

3.1 Dispatching

The dispatching attempts to match all available orders in the system with an available AGV. All available orders include those that have never been assigned and those that have been assigned but have not yet started to pick up. The inputs of the dispatching algorithm are dynamic attributes of orders and AGVs, including: the label and arrival time of each order; the label, berth, paths, request partial routes, request time, time to travel through the partial routes, assigned orders, and status of each AGV. It outputs sequences of orders assigned to each available AGV.

We propose a rule-based dispatching algorithm to allocate available orders to the nearest AGV within an acceptable range of distance. At the ℓ-th invocation of the algorithm, we have a set of available orders $I^{(\ell)}$ and a set of all AGVs except those in the ‘parking’ status $J^{(\ell)}$. The algorithm begins by dividing the $I^{(\ell)}$ into two subsets: $I_1^{(\ell)}$ and $I_2^{(\ell)}$, based on the time interval between the current dispatching time and the order arrival time. Orders in $I^{(\ell)}$ with an interval time greater than 300 seconds are classified as $I_1^{(\ell)}$, while other orders are classified as $I_2^{(\ell)}$. For the orders in $I_1^{(\ell)}$, the range that can be assigned is the entire grid Ω, whereas for the orders in $I_2^{(\ell)}$, the range that can be assigned is a fixed number of square units r, which can be adjusted as a tuning parameter to optimize the system performance.

We then calculate the distance $d_{ij}^{(\ell)}$ between each order $i \in I^{(\ell)}$ and each AGV $j \in J^{(\ell)}$. We use the Manhattan distance to calculate the distance between two square units $(x_1, y_1), (x_2, y_2) \in \Omega$, which is the sum of the absolute differences of their coordinates, i.e., $|x_1 - x_2| + |y_1 - y_2|$. The algorithm prioritizes the assignment of orders in $I_1^{(\ell)}$, followed by orders in $I_2^{(\ell)}$. Specifically, orders that arrive earlier are assigned first. For each $i \in I^{(\ell)}$, we select the AGV with the shortest distance as the assigned AGV, i.e., $j^* = \arg\min_j d_{ij}^{(\ell)}$. If a closer AGV is found during the ℓ-th invocation of the algorithm, orders that were previously assigned to an AGV in the $(\ell-1)$-th invocation are reassigned. Algorithm 1 shows the pseudocode for calculating $d_{ij}^{(\ell)}$ for each $i \in I^{(\ell)}$ in the dispatching algorithm. Here, c is the maximum number of orders that a vehicle can be assigned at the same time, which can be adjusted as a tuning parameter to optimize the system performance.

Algorithm 1: Distance between an order and an AGV

```
Input: $i \in I^{(\ell)}$, $J^{(\ell)}$, $r$, $c$
Output: $d_{ij}^{(\ell)}$
1  for $j \in J^{(\ell)}$ do
2      if $j \in J^{(\ell)}$ has no order in the $(\ell-1)$-th dispatching then
3          Calculate the distance between $i$'s picking rack and $j$'s current location;
4      else
5          if $i$ had been assigned to $j$ in the $(\ell-1)$-th dispatching then
6              if $i$ is the first order to be picked up by $j$ then
7                  Calculate the distance between $i$'s picking rack and $j$'s current location;
8              else
9                  Calculate the distance required for $j$ to complete all previous orders of $i$ assigned in
                   the $(\ell-1)$-th dispatching and reach $i$'s picking rack;
10             end
11         else
12             if the total number of orders in $j$ in the $(\ell-1)$-th dispatching is less than $c$ then
13                 if $j$ has new orders in the $\ell$-th dispatching then
14                     if the total number of orders in $j$ in the $(\ell-1)$-th and the $\ell$-th dispatching is
                       less than $c$ then
15                         Calculate the distance required for $j$ to complete all assigned orders in the
                           $(\ell-1)$-th and $\ell$-th dispatching and reach $i$'s picking rack;
16                     end
17                 else
18                     Calculate the distance required for $j$ to complete all assigned orders in the
                       $(\ell-1)$-th dispatching and reach $i$'s picking rack;
19                 end
20             else
21                 $d_{ij}^{(\ell)} = \infty$;
22             end
23         end
24     end
25     if $i \in I_1^{(\ell)}$, or the calculated distance of $i \in I_2^{(\ell)}$ is smaller than $r$ then
26         $d_{ij}^{(\ell)}$ = the calculated distance;
27     else
28         $d_{ij}^{(\ell)} = \infty$;
29     end
30 end
```

3.2 Route Planning

The route planning algorithm calculates routes for AGVs from their starting point to their destination. The algorithm takes three inputs: the entire grid Ω, the starting square unit, and the destination square unit. It outputs a path composed of square units. The partial route request algorithm reserves a partial route to avoid collisions for a moving AGV. This algorithm takes dynamic attributes of AGVs and grids as inputs, which include the label and occupancy of each square unit. The partial route request algorithm outputs a partial route of the whole path. We propose a collision-free route planning algorithm that uses BFS (Lee 1961) to search in a constrained search space that maintains a safety zone between AGVs. BFS is a search algorithm that explores a tree data structure in a breadth-first order, starting at the tree root and exploring all nodes at the present depth before moving on to the next depth level. By constructing a tree from the starting point and considering all possible moves, BFS can find the shortest path between two square units in an unweighted grid.

We consider two collision situations: opposite collision and same square unit collision, as shown in Figure 2. In opposite collision, AGV1 and AGV2 move head-to-head on two adjacent square units. In same square unit collision, AGV3 and AGV4 move toward the same square unit, while in another instance of same square unit collision, AGV6 moves to the square unit occupied by AGV5.

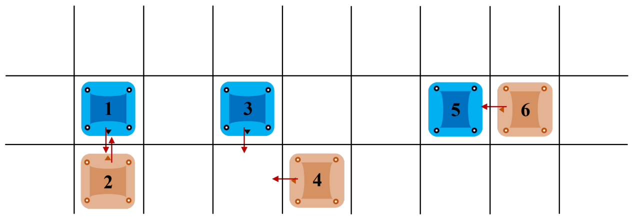


Figure 2: The collision situations of AGVs.

To develop a collision-free routing algorithm, we incorporate a constraint set into the search space of BFS. Suppose that at time t_r^j, the routing module is invoked to find a path for vehicle j, and let $\widetilde{I}^{(t_r^j)}$ denote all AGVs in motion at that time. In addition, suppose that vehicle $i \in \widetilde{I}^{(t_r^j)}$ arrives at its destination at time $\bar{t}_o^i$. We collect the complete paths of all AGVs in $\widetilde{I}^{(t_r^j)}$ to determine their locations on the grid at times $t_r^j, t_r^j+1, \cdots, \max_{i \in \widetilde{I}^{(t_r^j)}} \bar{t}_o^i$. Based on their future locations, we develop a constraint set that ensures a safe distance between vehicle j and other AGVs, thereby achieving collision avoidance. These constraints rely on two tuning parameters, s_1 and s_2, which can be adjusted to optimize the system performance. Specifically, suppose that vehicle $i \in \widetilde{I}^{(t_r^j)}$ locates in a square unit at time $t_o^i \in [t_r^j, \bar{t}_o^i)$. Then one of the constraints is set as vehicle j cannot move into that square unit in the time region $\widetilde{R}^{(t_o^i)} \triangleq \left[t_o^i - s_1, t_o^i + s_1 + 1\right]$. Since the sojourn time of vehicle i at its destination at time $\bar{t}_o^i$ is unknown to the route planning module, another constraint is set as vehicle j cannot move into the destination of i in the time region $\widetilde{R}^{(\bar{t}_o^i)} \triangleq \left[\bar{t}_o^i, \bar{t}_o^i + 4 + s_2\right]$, with 4 being the unloading duration mentioned in Section 2. The constraint set includes all constraints $\widetilde{R}^{(t_o^i)}$ and $\widetilde{R}^{(\bar{t}_o^i)}$, $t_o^i \in [t_r^j, \bar{t}_o^i)$, $\forall i \in \widetilde{I}^{(t_r^j)}$. For example, when $s_1 = 0$, the constraint $\widetilde{R}^{(t_o^i)} = \left[t_o^i, t_o^i + 1\right]$ ensures that vehicle j does not move into the same square unit as other AGVs at time t_o^i, as shown in the situation of AGV3 and AGV4 in Figure 2. It also ensures that vehicle j does not pass through other AGVs at time $t_o^i + 1$, as shown in the situation of AGV1 and AGV2 in Figure 2. If the constraint set is too large, BFS may fail to provide a solution, while if the constraint is too small, vehicle collisions may not be avoided entirely. However, determining the optimal set of constraints is challenging due to the limited input information available from simulation models. In the partial route request algorithm, the AGV requests 1 square unit from its remaining route as its next partial route.

3.3 Simulation Budget Allocation

We consider a simulation budget allocation problem of finding the tuning parameter combination (r, c, s_1, s_2) in the dispatching and route planning algorithms, which leads to the minimal ACT. Each combination represents a system design. Suppose that there are k designs and their corresponding unknown ACTs are denoted as μ_i, $i = 1, \cdots, k$, which can be estimated by independent simulation replications $X_{i\ell}$, $\ell \in \mathbb{Z}^+$. In simulation budget allocation problem, it is often assumed that $X_{i\ell}$, $X_{j\ell}$, $i \neq j$, follow independent normal distributions with unknown means and known variances, i.e., $X_{h,\ell} \sim N\left(\mu_h, \sigma_h^2\right)$, $h = 1, \cdots, k$. The normal assumption is justified by the central limit theorem in our work since the ACT, which is the average system time of certain orders arrived within 1 hour, is calculated as an average of many random variables. The central issue in simulation budget allocation is how to allocate simulation replications intelligently to improve sampling efficiency. A popular metric to measure sampling efficiency is the probability of correct selection (PCS), i.e., $\text{PCS} = \Pr\{\bigcap_{i=2}^{k} \mu_{\langle 1\rangle_T} < \mu_{\langle i\rangle_T}\}$, where $\langle i\rangle_T$ are indices ranked in ascending order by estimated performance after allocating a total of T simulation replications. There are numerous existing sampling procedures in the research of simulation budget allocation, and we adopt the AOAP which is derived under the Bayesian framework. A conjugate prior distribution $N(\mu_i^{(0)}, (\sigma_i^{(0)})^2)$ is introduced to quantify the uncertainty of μ_i, and its posterior distribution is given by $N(\mu_i^{(t)}, (\sigma_i^{(t)})^2)$, where

$$(\sigma_i^{(t)})^2 = \left(\frac{1}{(\sigma_i^{(0)})^2} + \frac{t_i}{\sigma_i^2}\right)^{-1}, \quad \mu_i^{(t)} = (\sigma_i^{(t)})^2 \left(\frac{\mu_i^{(0)}}{(\sigma_i^{(0)})^2} + \frac{t_i m_i^{(t)}}{\sigma_i^2}\right),$$

where t is the total number of allocated replications, t_i is the number of allocated replications to design i, $m_i^{(t)} = \sum_{\ell=1}^{t_i} X_{i,\ell} \big/ t_i$ is the sample mean. The dynamic decisions in simulation budget allocation problem can be formulated as an allocation and selection policy, and the optimal allocation and selection policy satisfies a Bellman equation. The AOAP is derived in an approximate dynamic programming paradigm, which finds a suitable value function approximation for the Bellman equation. Let $\mathscr{E}_t$ be the information set obtained after allocating t simulation replications, which is completely determined by the posterior hyper-parameters, and let $A_{t+1}(\mathscr{E}_t)$ be the allocation decision based on collected information. Then the AOAP is given by

$$A_{t+1}(\mathscr{E}_t) = \arg\max_{i=1,\cdots,k} \widehat{V}_t(\mathscr{E}_t; i),$$

where $\langle 1\rangle_t = \min_{i=1,\cdots,k} \mu_i^{(t)}$,

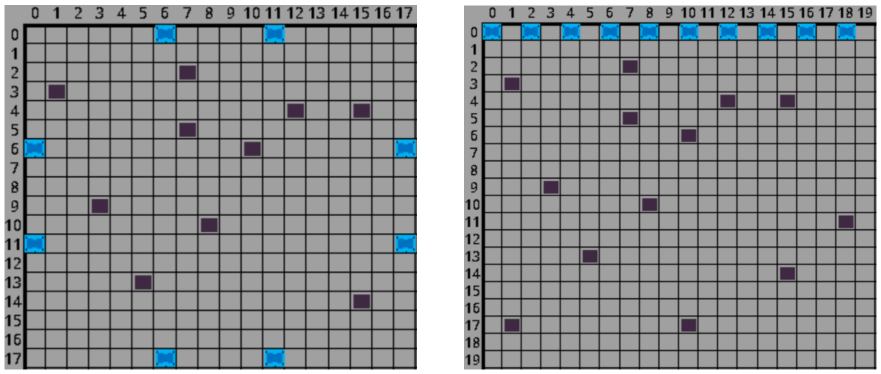
$$\widehat{V}_t(\mathscr{E}_t; 1) = \min_{j\neq 1} \frac{\left(\mu_{\langle 1\rangle_t}^{(t)} - \mu_{\langle j\rangle_t}^{(t)}\right)^2}{\left(\sigma_{\langle 1\rangle_t}^{(t+1)}\right)^2 + \left(\sigma_{\langle j\rangle_t}^{(t)}\right)^2},$$

$$\widehat{V}_t(\mathscr{E}_t; j) = \min\left\{ \frac{\left(\mu_{\langle 1\rangle_t}^{(t)} - \mu_{\langle j\rangle_t}^{(t)}\right)^2}{\left(\sigma_{\langle 1\rangle_t}^{(t)}\right)^2 + \left(\sigma_{\langle j\rangle_t}^{(t+1)}\right)^2}, \min_{\ell\neq 1,j} \frac{\left(\mu_{\langle 1\rangle_t}^{(t)} - \mu_{\langle \ell\rangle_t}^{(t)}\right)^2}{\left(\sigma_{\langle 1\rangle_t}^{(t)}\right)^2 + \left(\sigma_{\langle \ell\rangle_t}^{(t)}\right)^2} \right\}, \quad j = 2, \cdots, k.$$

When the number of designs becomes large, the limited computational power of a single processor can restrict one from solving the problem within a reasonable amount of time. With the rapid development of computing technology, parallel computing becomes increasingly prevalent and accessible to ordinary users. In this work, we do not develop parallel budget allocation procedures but instead implement the AOAP in a parallel computing environment with m processors. We equally allocate the k designs and T simulation budget to the m processors. In each processor, the AOAP sequentially allocates T/m simulation replications, allocating one simulation replication at each step, to find the design with the best estimated performance among k/m designs. Subsequently, the best design is selected from the m designs determined by the m processors by comparing their estimated performances. By doing so, the promising top designs have a greater chance of surviving until the end of the simulation, and the computational time required to find the best design can be significantly reduced. In general, if a scheduling method of AGVs is designed for multiple scenarios, then the problem of finding the optimal tuning parameter combination can be formulated as a context-dependent simulation budget allocation problem, where the goal is to correctly select the best design for each scenario. For a review on the context-dependent simulation budget allocation problem, please refer to Keslin et al. (2022).

4 NUMERICAL EXPERIMENTS

In this section, we conduct numerical experiments to demonstrate the effectiveness of our proposed scheduling method. We compare our method against a default approach provided in the WSC 2022. The default dispatching algorithm is the same as that in Section 3.1 with $r = 5$ and $c = 3$, except for some differences in how the distances between AGVs and orders are determined. Specifically, in Line 15 of Algorithm 1, the default algorithm calculates the distance between i's picking rack and the delivery location of the last order in the ℓ-th dispatching, and in Line 18, the default algorithm calculates the distance between i's picking rack and the delivery location of the last order in the $(\ell-1)$-th dispatching. The default route planning algorithm uses the A^* search algorithm (Hart, Nilsson, and Raphael 1968) to generate routes for each AGV, finding the shortest path based on the cost of the path and an estimate of the cost required to extend the path all the way to the destination. Moreover, the default partial route request algorithm requests 3 square units from its remaining route as its next partial route. In case there are fewer than 3 square units left, the algorithm requests the remaining number of square units as its next partial route. Figure 3 illustrates the layouts of two warehouse management system simulation scenarios that are considered in the numerical experiments. In Scenarios 1 and 2, AGVs move in an 18×18 grid and a 20×20 grid, respectively. The inputs of the simulation model include the inter-arrival times of orders, which follow exponential distributions with mean parameters of 70 and 120 orders per hour in Scenarios 1 and 2, respectively. Scenario 2 is more challenging than Scenario 1 due to the larger quantity of orders, larger AGV fleet size, denser berths, and more obstacles.



(a) Scenario 1 (b) Scenario 2

Figure 3: Layouts of two warehouse management system simulation scenarios: (a) Scenario 1; (b) Scenario 2. The blue square units represent berths, and the purple square units represent obstacles.

We report some performance metrics of the system in addition to the ACT in the numerical results, including: the average ratio of finished orders $R_f = \mathbb{E}\left[N_f/N\right]$; the average waiting time for orders to be picked up by AGVs on picking racks W_{order}; the average waiting time caused by congestion for loaded AGVs during delivery W_{loaded}; and the average waiting time caused by congestion for empty AGVs during movement W_{empty}. We conduct two numerical experiments under both simulation scenarios. In the first experiment, we compare system performance with and without implementing simulation budget allocation methods under the default algorithm. In the second experiment, we compare the performance of the system designed under our proposed method and the default algorithm. The AOAP is implemented with non-informative priors. Both experiments are conducted in a parallel computing environment with $m = 24$ processors. The total simulation budget is $T = 4800$, and each metric is estimated by 1000 independent macro experiments. The code of the proposed scheduling method for the numerical experiments can be found at https://github.com/xiaotianliu01/2nd-Solution-for-WSC-2022-Case-Competition.

Experiment 1 (Effectiveness of Simulation Budget Allocation Method) We begin by conducting a simple numerical experiment to demonstrate the effectiveness of simulation budget allocation method by comparing the default algorithm with its adaptation using simulation budget allocation method. In the adapted default algorithm, we consider r and c in the dispatching module, as well as the reservation length ℓ_{res} in the partial request module of the default method, as tuning parameters. We aim to find the parameter combination (r, c, ℓ_{res}) that leads to the minimal ACT for each scenario. Specifically, we set $r = 1, 4, 7, 10$; $c = 1, 10, 19, \cdots, 100$, and $\ell_{res} = 1, 2, 3, \cdots, 16$, resulting in a total of 768 designs, which are assigned evenly to 24 processors.

By using AOAP to intelligently allocate 200 simulation replications in each processor, we obtain $(r^*, c^*, \ell_{res}^*) = (64, 1, 6)$ for Scenario 1 and $(r^*, c^*, \ell_{res}^*) = (73, 4, 7)$ for Scenario 2. The optimal parameter combination for each scenario is different from that used in the default algorithm. We observe that the values of the optimal parameter combination in Scenario 2 are larger than those in Scenario 1, which can be attributed to the reason that Scenario 2 is more challenging than Scenario 1. Table 1 shows various performance metrics of the system under the default algorithm and its adaptation for the two scenarios.

Table 1: System performance with and without implementing simulation budget allocation methods under the default algorithm.

		R_f	W_{order}	W_{loaded}	W_{empty}	ACT
Scenario 1	Default	0.838	1388.9s	25.2s	5.8s	240.9s
	Adaptation	0.847	812.8s	9.8s	2.9s	222.7s
Percentage change		**1.1%**	-41.5%	-61.1%	-50.0%	**-7.6%**
Scenario 2	Default	0.586	2055.4s	16.3s	9.2s	687.4s
	Adaptation	0.667	2791.8s	23.6s	16.7s	565.8s
Percentage change		**13.8%**	35.8%	44.8%	81.5%	**-17.7%**

In Table 1, we can observe that the adapted default algorithm results in an increased average ratio of finished orders R_f and a decreased ACT compared to the default algorithm. In Scenario 2, although the adapted default algorithm increases the waiting times W_{order}, W_{loaded} and W_{empty}, which can be attributed to its larger request for square units to avoid collisions, it lowers the ACT. Comparing Scenario 1 and Scenario 2 shows that the advantage of implementing simulation budget allocation method appears to be more pronounced as the scenario becomes more challenging.

Experiment 2 (Effectiveness of the Proposed Scheduling Method) We conduct numerical experiment to demonstrate the effectiveness of the proposed scheduling method. We set $r = 1, 4, 7, 10$; $c = 1, 10, 19, \cdots, 100$; $s_1 = 1, 2, 3, 4$, and $s_2 = 0, 1, 2, 3$, resulting in a total of 768 designs, which are assigned evenly to 24 processors. We then use AOAP to find the optimal parameter combination (r, c, s_1, s_2) in the dispatching and route planning algorithms that leads to the minimal ACT.

We obtain $(r^*, c^*, s_1^*, s_2^*) = (91, 7, 2, 3)$ for Scenario 1 and $(r^*, c^*, s_1^*, s_2^*) = (100, 7, 2, 3)$ for Scenario 2. The proposed scheduling method makes the optimal parameter combination for each scenario different from that used in the default algorithm and that used in the adapted default algorithm. Scenario 2 sets a larger range that orders can be assigned than Scenario 1. However, values of c^*, s_1^*, and s_2^* are the same in the two scenarios.

Figure 4 shows heat maps of ACTs under different parameter combinations (r, c). The heat maps use color to depict the ACT values, with yellow indicating the maximum and dark blue indicating the minimum. We can see that for both scenarios, there is no obvious pattern for different parameter combinations (r, c) that make the system perform better. Figure 5 shows the ACT values for different designs in ascending order. We can see that different parameter combinations have significant impacts on the system performance, which underscores the importance of using simulation budget allocation method to identify the best design for a stochastic system. Figure 6 provides visual demonstrations of AGV movements in two simulation scenarios under our proposed scheduling method.

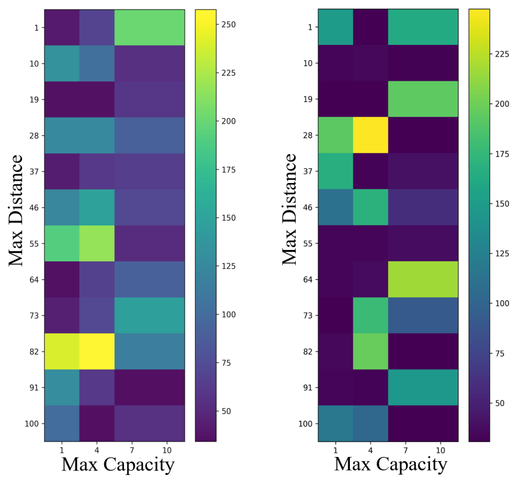


Figure 4: ACTs under different parameter combinations (r,c): (a) Scenario 1; (b) Scenario 2. The x-axis represents c, and the y-axis represents r.

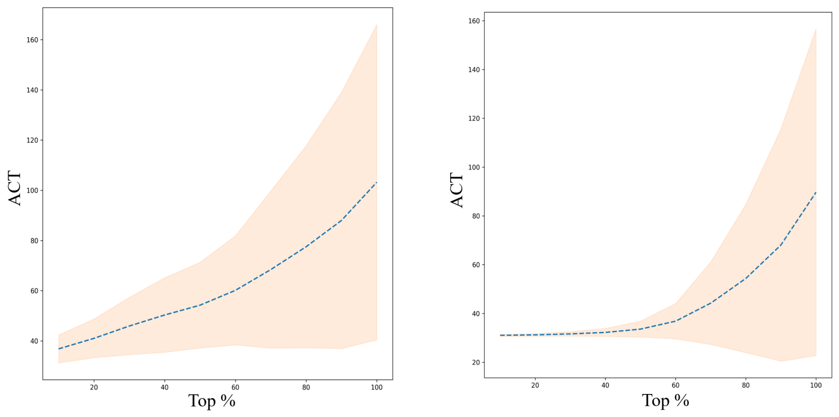
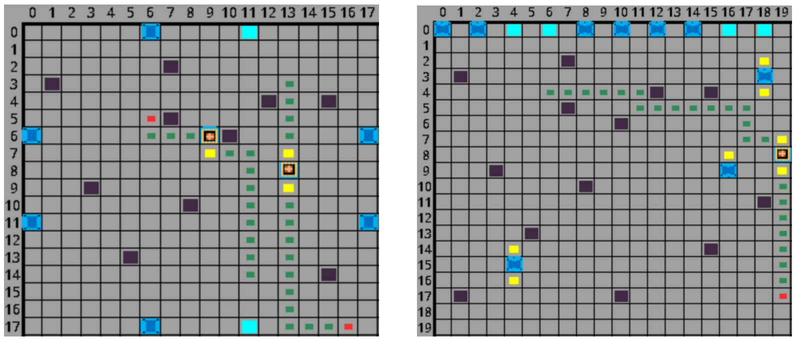


Figure 5: ACT values for different designs in ascending order: (a) Scenario 1; (b) Scenario 2.

Table 2 displays various performance metrics of the system under the default algorithm and our proposed scheduling method. We can observe that our method significantly improves the average ratio of finished orders R_f and lowers the ACT compared to the default algorithm. Furthermore, compared to the numerical results from *Experiment 1*, our scheduling method outperforms the adapted default method, which solely uses simulation budget allocation method to improve the system performance. Although our proposed scheduling method increases certain waiting times during AGV movements, which can be attributed to the constraint set that avoids vehicle collisions in the route planning algorithm, it reduces the ACT. Comparing Scenario 1 and Scenario 2 shows that the advantage of our proposed scheduling method appears to be more pronounced when the scenario becomes more challenging.

5 CONCLUSION

This paper studies a scheduling problem for AGVs in a stochastic warehouse management system. We develop efficient rule-based dispatching and collision-free route planning algorithms. The optimal parameter



(a) Scenario 1 (b) Scenario 2

Figure 6: Visual demonstrations of AGV movements: (a) Scenario 1; (b) Scenario 2. The blue square units represent berths, the purple square units represent obstacles, the red square units represent delivery locations, the green square units represent paths, and the yellow square units represent requested ones.

Table 2: The performance of the system designed under the default algorithm and our proposed scheduling method.

		R_f	W_{order}	W_{loaded}	W_{empty}	ACT
Scenario 1	Default	0.838	1388.9s	25.2s	5.8s	240.9s
	Proposed Method	0.961	801.7s	69.5s	6.7s	86.3s
Percentage change		**14.7%**	-42.3%	175.8%	15.5%	**-64.2%**
Scenario 2	Default	0.586	2055.4s	16.3s	9.2s	687.4s
	Proposed Method	0.989	1468.6s	33.0s	5.4s	48.9s
Percentage change		**68.8%**	28.5%	102.5%	41.3%	**-92.9%**

combination for these algorithms is found using simulation budget allocation method implemented in a parallel computing environment. Numerical experiments demonstrate that simulation budget allocation method significantly enhances the system performance, and the proposed scheduling method outperforms the default algorithm.

The deadlock avoidance in the scheduling method deserves future work. Parallel fixed-budget simulation budget allocation policies are worthy of further study. Future research also includes developing multi-agent reinforcement learning algorithms for the simultaneous dispatching and route planning of AGVs. How to develop a warehouse system simulation model that can incorporate multi-agent reinforcement learning algorithms could also be future work.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 72250065, 72022001, and 71901003, and by the China Scholarship Council (CSC) under Grant CSC202206010152.

REFERENCES

Adamo, T., T. Bektaş, G. Ghiani, E. Guerriero, and E. Manni. 2018. "Path and Speed Optimization for Conflict-Free Pickup and Delivery Under Time Windows". *Transportation Science* 52(4):739–755.

An, Y., M. Li, X. Lin, F. He, and H. Yang. 2020. "Space-Time Routing in Dedicated Automated Vehicle Zones". *Transportation Research Part C: Emerging Technologies* 120:102777.

Chen, C.-H., J. Lin, E. Yücesan, and S. E. Chick. 2000. "Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization". *Discrete Event Dynamic Systems* 10:251–270.

Corréa, A. I., A. Langevin, and L.-M. Rousseau. 2007. “Scheduling and Routing of Automated Guided Vehicles: A Hybrid Approach”. *Computers & Operations Research* 34(6):1688–1707.

De Ryck, M., M. Versteyhe, and F. Debrouwere. 2020. “Automated Guided Vehicle Systems, State-Of-The-Art Control Algorithms and Techniques”. *Journal of Manufacturing Systems* 54:152–173.

Fazlollahtabar, H., and M. Saidi-Mehrabad. 2015. “Methodologies to Optimize Automated Guided Vehicle Scheduling and Routing Problems: A Review Study”. *Journal of Intelligent & Robotic Systems* 77:525–545.

Fransen, K., J. Van Eekelen, A. Pogromsky, M. A. Boon, and I. J. Adan. 2020. “A Dynamic Path Planning Approach for Dense, Large, Grid-Based Automated Guided Vehicle Systems”. *Computers & Operations Research* 123:105046.

Frazier, P. I., W. B. Powell, and S. Dayanik. 2008. “A Knowledge-Gradient Policy for Sequential Information Collection”. *SIAM Journal on Control and Optimization* 47(5):2410–2439.

Hart, P. E., N. J. Nilsson, and B. Raphael. 1968. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Hu, H., X. Jia, Q. He, S. Fu, and K. Liu. 2020. “Deep Reinforcement Learning Based AGVs Real-Time Scheduling with Mixed Rule for Flexible Shop Floor in Industry 4.0”. *Computers & Industrial Engineering* 149:106749.

Hu, H., X. Yang, S. Xiao, and F. Wang. 2023. “Anti-Conflict AGV Path Planning in Automated Container Terminals Based on Multi-Agent Reinforcement Learning”. *International Journal of Production Research* 61(1):65–80.

Kabir, Q. S., and Y. Suzuki. 2019. “Comparative Analysis of Different Routing Heuristics for the Battery Management of Automated Guided Vehicles”. *International Journal of Production Research* 57(2):624–641.

Keslin, G., B. L. Nelson, M. Plumlee, B. K. Pagnoncelli, and H. Rahimian. 2022. “A Classification Method for Ranking and Selection with Covariates”. In *Proceedings of the 2022 Winter Simulation Conference*, edited by B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C. Corlu, L. Lee, E. Chew, T. Roeder, and P. Lendermann, 1–12. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lee, C. Y. 1961. “An Algorithm for Path Connections and Its Applications”. *IRE Transactions on Electronic Computers* (3):346–365.

Lee, J. H., B. H. Lee, and M. H. Choi. 1998. “A Real-Time Traffic Control Scheme of Multiple AGV Systems for Collision Free Minimum Time Motion: A Routing Table Approach”. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 28(3):347–358.

Lee, L. H., E. P. Chew, K. C. Tan, and Y. Wang. 2010. “Vehicle Dispatching Algorithms for Container Transshipment Hubs”. *OR Spectrum* 32:663–685.

Nishi, T., Y. Hiranaka, and I. E. Grossmann. 2011. “A Bilevel Decomposition Algorithm for Simultaneous Production Scheduling and Conflict-Free Routing for Automated Guided Vehicles”. *Computers & Operations Research* 38(5):876–888.

Peng, Y., E. K. Chong, C.-H. Chen, and M. C. Fu. 2018. “Ranking and Selection as Stochastic Control”. *IEEE Transactions on Automatic Control* 63(8):2359–2373.

Peng, Y., and G. Zhang. 2022. “Thompson Sampling Meets Ranking and Selection”. In *Proceedings of the 2022 Winter Simulation Conference*, edited by B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C. Corlu, L. Lee, E. Chew, T. Roeder, and P. Lendermann, 3075–3086. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ryzhov, I. O. 2016. “On the Convergence Rates of Expected Improvement Methods”. *Operations Research* 64(6):1515–1528.

Vis, I. F. 2006. “Survey of Research in the Design and Control of Automated Guided Vehicle Systems”. *European Journal of Operational Research* 170(3):677–709.

Vis, I. F., R. De Koster, K. J. Roodbergen, and L. W. Peeters. 2001. “Determination of the Number of Automated Guided Vehicles Required at a Semi-automated Container Terminal”. *Journal of the Operational Research Society* 52(4):409–417.

Zhen, L., Y.-W. Wu, S. Zhang, Q.-J. Sun, and Q. Yue. 2020. “A Decision Framework for Automatic Guided Vehicle Routing Problem with Traffic Congestions”. *Journal of the Operations Research Society of China* 8:357–373.

AUTHOR BIOGRAPHIES

GONGBO ZHANG is a Ph.D. candidate in Guanghua School of Management at Peking University, Beijing, China. His research interests include stochastic modeling and analysis, simulation optimization and reinforcement learning. His email address is gongbozhang@pku.edu.cn.

XIAOTIAN LIU receives his bachelor degree from School of Electronics Engineering and Computer Science at Peking University. His reserach interests include deep reinforcement learning and its applications. His email is xiaotianliu01@gmail.com.

YIJIE PENG is an Associate Professor in Guanghua School of Management at Peking University, Beijing, China. His research interests include stochastic modeling and analysis, simulation optimization, machine learning, data analytics, and healthcare. His email address is pengyijie@pku.edu.cn.