Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

PLANNING A MATERIAL REPLENISHMENT THROUGH AUTONOMOUS MOBILE ROBOT IN AN ASSEMBLY PLANT USING A HYBRID SIMULATION APPROACH

Rupesh Bade Shrestha[1], Ellie Hungerford[1], Konstantinos Mykoniatis[1]

[1]Dept. of Industrial and Systems Eng., Auburn University, Auburn, AL, USA

ABSTRACT

This paper focuses on planning the implementation of an autonomous mobile robot in an existing material replenishment system of an assembly plant using a hybrid simulation approach. In this research, we aim to identify proper strategies for the number of containers or payloads the robot should carry per shift under different scenarios using Agent-Based Modeling and Discrete Event Simulation. Our primary objective is to minimize the number of shifts a mobile robot should travel for replenishment while keeping idle time low across all stations and ensuring timely material replenishment. The results show that choosing which strategy to use for the implementation of the autonomous navigating robot depends on the maximum number of containers it can carry and the utilization of payload space. While this paper focuses on Tiger Motors Assembly line at Auburn University, its applications could be extended to similar assembly plants equipped with a similar material replenishment system.

1 INTRODUCTION

In an assembly plant, material handling is the movement and control of materials (raw, finished, or packaged) throughout the assembly process. The movement of raw materials plays a very important role in the performance of an assembly plant that affects assembly time, cost, productivity, utilization of space, work environment, flexibility, safety, inventory level, etc. For the same reason, equipment to be used in material handling (material replenishment) and its efficient use is also very important (Arora and Shinde 2007).

Autonomous Mobile Robots (AMRs) are ground vehicles that can plan travel while preventing collisions with environmental obstacles (Hentout et al. 2010). These robots can operate autonomously without any human intervention and excel in handling material replenishment along with surveillance, inspection, and maintenance (Datta and Ray 2007). However, efficient and effective use of an AMR is very important to use it, and requires proper planning and scheduling. (Liu et al. 2023)

Simulation can help in planning the operations in an assembly plant (Villarreal and Alanís 2011). Hybrid simulation is the combination of at least two kinds of simulation methods (Mykoniatis and Angelopoulou 2020). The hybrid simulation approach is usually preferred to get the benefit of many different simulation methods in a single model or be able to answer modeling questions that can be better addressed by different simulation approaches. This study combines Discrete Event Simulation (DES) and Agent-Based Modeling (ABM), using AnyLogic simulation software version 8.8.6. DES is a process-centric approach (Mykoniatis and Angelopoulou 2020) and is used in modeling systems where entities compete to use resources, for which they form a queue (Caro and Möller 2016). However, the highly detailed behavior of entities and dynamic decision-making of individual entities are very difficult to model in DES, whereas they can be better captured using ABM (Dubiel and Tsimhoni 2005). On the other hand, ABM models are computationally very expensive and very difficult to validate, which is not the case in DES (Khodabandelu and Park 2022). This combination allows us to model the rule-based behavior of our agents in ABM, and the process-based environment in DES. This helps us to counterbalance the strengths and weaknesses of DES and ABM (Araya 2022).

The Tiger Motors lab at Auburn University provides a downsized assembly process for hands-on training and production runs of miniature Lego cars. There are three sub-assembly cells in the Tiger Motors lab, with five stations in each. Each of these stations holds containers with various Lego parts used to assemble the vehicles. As assembly occurs, the Lego parts are pulled from the containers and the AMR will replenish the containers with Lego parts as needed. An operator in the first workstation of cell 1 (as shown in Figure 1) starts building a car using Lego bricks from the containers, passes it to the next station and the operator in that station builds on top of that, and this goes on until a complete car is built in station 15 of cell 3. Other operators enhance the assembly process through flex roles, specifically tasked with replenishing the supply of Lego bricks during production. The implementation of an AMR would allow for the role of flex operators to become automated. Successful implementation of the AMR would remove unnecessary operator work, robot movement, and excess inventory.

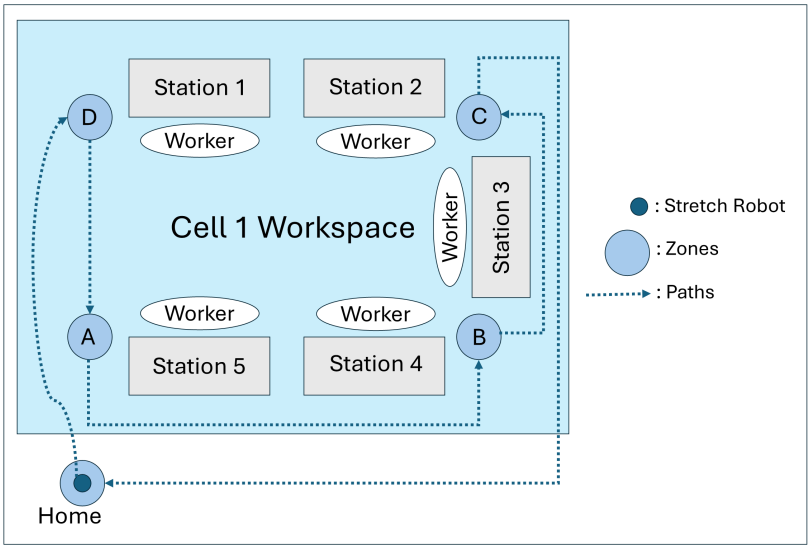


Figure 1: Cell 1 layout in Tiger Motors lab. Cell 1 consists of five stations, for which four zones are assigned where the Stretch robot can deliver payloads. Initially, the Stretch robot stays in the Home position.

An AMR delivers containers based on different conditions (scenarios), and each station can only accommodate two containers of each type. Hence, it's essential to maintain a consistent quantity of containers, ensuring there are always at least one and at most two containers for each type. When the AMR carries too many containers for replenishment, the robot will travel fewer shifts, but will have high idle time in stations or there will not be enough space to add the containers in the stations. This happens because when a robot delivers too many containers at the same time, the robot either must wait for a long time to make a delivery, or it must make an early delivery at the stations. When the robot carries too few containers, a lot of time will be wasted during travel, and because of that travel time, stations can still be idle.

In our case study, we utilized the Stretch RE1 mobile manipulator from Hello Robot; however, the gripper attached to the Stretch robot remained unused, and only one type of Lego car was produced through the stations. The stretch robot can plan its path to the programmed location or coordinates, avoid and plan paths around the obstacles autonomously, and have a decentralized control system. In this work, we consider only cell 1 which consists of five stations and a Stretch robot. When Stretch goes to the home position, it is filled with the corresponding containers using its robotic arm which is out of scope for our study. Whenever any station becomes idle, the parameters that make the station idle are rejected. In Tiger Motors lab, a toy car also requires other kinds of Lego bricks, but they are not considered in the study since they don't need any replenishment. Also, the Stretch robot only follows the path from Home to Zone

D, Zone D to A, Zone A to B, Zone B to C, and Zone C to Home in each shift as shown in Figure 1. The number of bricks in each container helps in calculating the time for how long that container will last, but the number of bricks in the container will not be traced or used in the model. Although containers can have different kinds of LEGO bricks, the size of the container is fixed ($8 * 10 * 19 cm^3$).

2 LITERATURE REVIEW

In modern manufacturing, the utilization of an AMR could implement new levels of efficiency and adaptability in manufacturing environments. An AMR for managing material replenishment could improve operations and reduce worker movements.

Many studies revolve around enhancing and optimizing manufacturing processes when utilizing AMRs. Nielsen et al. (2017) proposed a methodology to implement AMRs in manufacturing facilities by considering it as an optimization problem. This methodology consists of a mathematical model and a genetic algorithm-based heuristic. Zhou and Xu (2017)'s research similarly followed the works of Nielsen et al. (2017). However here, a support vector machine addressed the challenge of multiple load carriers for material delivery. Kousi et al. (2016) proposed a decision-making method for optimal plan generation of part supply operations using Mobile Assistant Units (MAU) based on time and inventory level. The implementation of this study ran a MATLAB environment to simulate a monitoring system. On the other hand, Waseem and Chang (2023) used Q-learning for real-time scheduling of AMR in a multi-product flexible manufacturing system. In this method, the AMR responded and adapted to potential rewards in the environment to optimize scheduling decisions within manufacturing.

However, using only a mathematical model to represent a complicated system can be a highly complex task, and it can be challenging to test what-if scenarios, change configurations, and visualize the results (Carson 2005). The use of a simulation model can help overcome these obstacles. Hence, Moretti et al. (2021) proposed an ABM simulation model composed of factory warehouses, assembly stations, and AMRs for dynamic assignment of tasks, which would be impractical to model using a mathematical model only. Similarly, Vieira et al. (2018) explored the planning and scheduling of industrial mobile robots using a DES for flexibility and automation. The model created a decision-support system for production planning and scheduling. Pappert et al. (2010) also worked on a general framework for simulation-based scheduling of assembly lines. They offered a meta-model as a tool to develop optimization models for various production facilities and layouts.

A hybrid simulation, which is the combination of at least two simulation methods, can also be used to make the modeling process more effective since it can take benefit of different simulation methods, and also account for the cons of an individual simulation model. According to Zheng et al. (2017), the use of hybrid simulation made their installation planning easy to use and easily understood by many people, with a high level of performance. Also, Conn et al. (2010) used a hybrid model composed of the Generalized Semi-Markov Model and Monte Carlo simulation to evaluate maintenance plans for offshore oil installations, which allowed flexibility and possible modifications.

Our study focuses on planning the installation of AMR by comparing different strategies. Planning the installation or implementation of AMR into an existing replenishment using a hybrid simulation tool is a unique contribution of this paper. Hybrid modeling composed of DES and ABM is preferable for our installation scheduling as our model uses ABM to replicate the rule-based behavior of material replenishment and depletion, while DES captures the container delivery process by the Stretch robot and its statistics.

3 RESEARCH METHODOLOGY

The use of hybrid simulation allows for several aspects of the Tiger Motors Lab to be utilized into a singular model. Since it was easier to model the container-carrying behavior of a Stretch robot with a process-centric approach, we used DES to model that (explained in sub-section 3.4). For the depletion and replenishment behavior of containers, each container had to follow specific rules or behaviors per state,

hence we used ABM (explained in sub-section 3.2). With the use of hybrid simulation, we can model all of these aspects simultaneously.

Table 1: Input and output parameters of our model.

Variables	Name in Model	Type	Description
x_{1i} $i \in [1,10]$	TimeToEmpty	Input variable	Time taken in second by each type of container to empty.
x_{2j} $j \in [1,5]$	Trip Time	Input variable	Trip time for a Stretch RE1 in second to travel from Home to Zone D, D to A, A to B, B to C and C to Home respectively.
x_3	CarryLimit	Input variable	The limit of containers a Stretch RE1 can carry in a single shift.
x_4	RobotWaitTime	Input variable for Scenario 1	The time in second a Stretch RE1 has to wait in a Home position before going for a next shift.
x_5	RushTime	Input variable for Scenario 3	The time difference between current time and NearestDeadline before which a Stretch RE1 rushes or moves to deliver container.
C_{i-1}	Container 0, 1, ... 9	Agent	Container agents representing ten different types of container.
y_{1i}	Deadlines	Intermediate Variable	Variable assigned for each type of container that tells when the Container of each kind becomes empty.
y_{2i}	ContainerCondition	Intermediate Variable	Available number of containers for each kind.
y_3	RobotContainer	Intermediate Variable	A collection of strings, that collects the name of containers that will be replenished in the next shift based on Deadlines.
y_4	NoOfTotalAvailableContainers	Intermediate Variable	Total number of containers available in the system.
y_5	NearestDeadline	Intermediate Variable	The earliest deadline for a container which represents the time at which any container will be empty and station becomes idle.
y_6	NoOfContainersReplenished	Intermediate Variable	Total number of containers replenished.
y_7	Shifts	Output variable	Total number of shifts done by Stretch RE1 in 40 minutes for replenishment.
y_8	IdleNo	Output variable	Total number of times a station became idle because of empty container in the station in 40 minutes.
y_9	PayloadUtilization	Output variable	Utilization of CarryLimit in percentage.

Table 1 illustrates the input and output parameters for the model. We collected the Trip Time data using a Stretch RE1 mobile manipulator developed by Hello Robot. We used thirty-three observations for *Trip Time* of *moveToHome* block (as shown in Figure 4). Similarly, we used thirty-nine observations for the rest of the *moveTo* blocks from Figure 4. For each observation, Stretch RE1 was programmed to travel and follow the path shown in Figure 1 autonomously. For the input analysis and goodness of fit of *Trip Time*, we used @RISK. Table 2 shows the time distributions from input analysis.

Table 2: Trip Time Distribution in seconds.

Block	Trip Time Distribution	Description
moveToD	triangular(22,36,22)	Travel time in second from Home to Zone D
moveToA	normal(1.5324,8.3846)	Travel time in second from Zone D to A
moveToB	uniform(51.316,78.684)	Travel time in second from Zone A to B
moveToC	pareto(7.4755,31)	Travel time in second from Zone B to C
moveToHome	uniform(44.219,70.781)	Travel time in second from Zone C to Home

During the development of our hybrid model, we made five assumptions. These help the model to have the required level of simplicity and clarity. These assumptions are listed below:

- The cycle time to produce a single car is 80 seconds and the stations build 30 cars for 40 minutes.

- The layout in Figure 1 assumes that Zone D will be accessible by Station 1 operator, Zone C by Station 2 operator, Zone B by Station 3 as well as Station 4 operators, and Zone A by Station 5 operator. So, whenever the robot enters that zone, the corresponding worker will be able to pick up the filled containers and replace them with the empty ones.
- In Zone D, *Container 0* and *Container 1* needs to be replenished. Similarly, Zone A requires replenishment of *Container 2* and *Container 3*; Zone B requires *Container 4*, *Container 5*, *Container 6* and *Container 7*; and Zone C requires *Container 8* and *Container 9*.
- The workers will be able to replace the filled containers with the empty ones while the Stretch robot is trying to align itself in that zone because, during the alignment process, the Stretch robot will be already in the respective zone and trying to align itself more accurately.
- We assume that there is enough supply of all types of Lego bricks in the store or home.

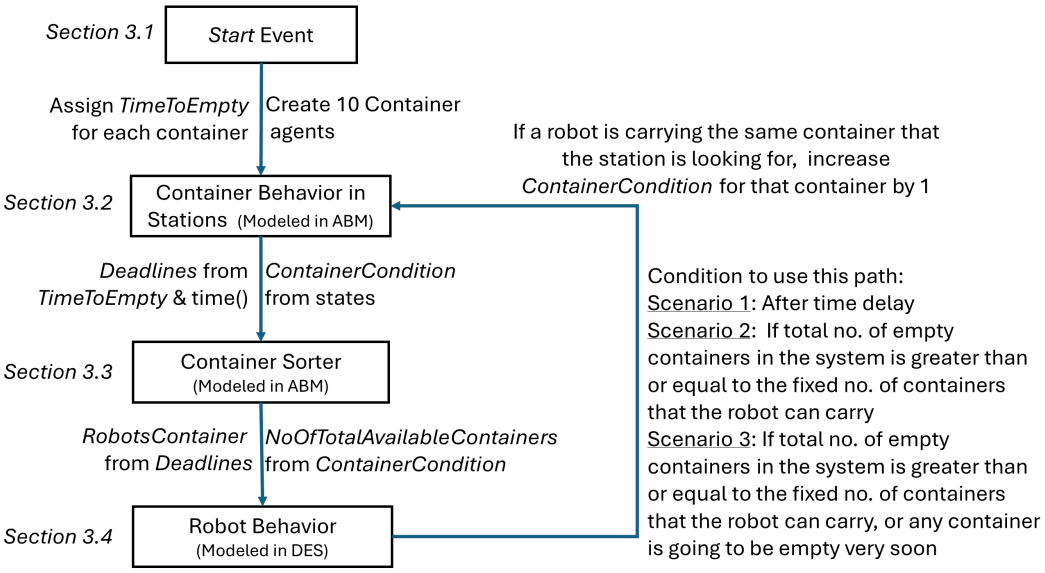


Figure 2: Model Summary.

The model has four important parts i.e., *Start* Event, Container Behavior in Stations, Container Sorter, and Robot Behavior. The way how these pillars interact using variables is shown in Figure 2. The following sub-section provides a detailed explanation of how these pillars perform:

3.1 *Start* Event

This event is triggered at the very start of the simulation. Table 3 provides all the necessary information related to the containers that need replenishment. It shows which station or zone each container belongs to, the number of Lego bricks in each container, the number of Lego bricks from that container required to build a single car, how many cars can be built from each container, and time in second until the depletion of that container. Since it takes 80 seconds to build each car, we calculate the time until depletion for each container by multiplying 80 by the number of cars that can be built from each container. These values are assigned to *TimeToEmpty* in the *Start* event.

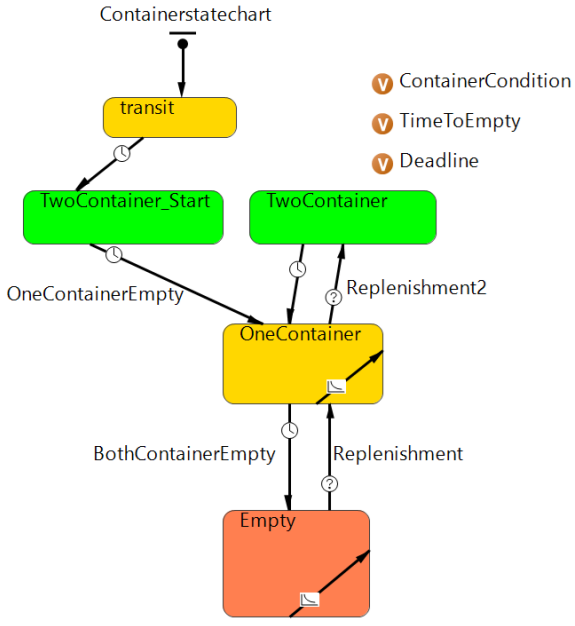
3.2 Container Behaviour in Stations

The replenishment and depletion behavior of containers, which shows the rule-based behavior of our agents (containers), is captured using ABM. Agent population *containers* represents the Container Behavior in

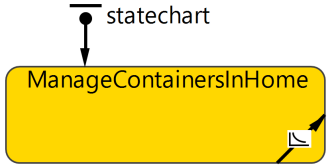
Table 3: Container Information. Different containers deplete at different rates.

Container no.	Station no.	Zone	Lego bricks quantity	No. of pieces in each car	Cars yield	Time until depletion in second (Cars yield * 80sec)
0	1	D	4	1	4	320
1	1	D	8	2	4	320
2	5	A	8	1	8	640
3	5	A	8	1	8	640
4	4	B	28	4	7	560
5	4	B	12	1	12	960
6	3	B	6	1	6	480
7	3	B	7	2	3.5	280
8	2	C	16	4	4	320
9	2	C	14	2	7	560

Stations and has three main states such as *TwoContainer*, *OneContainer* and *Empty* as shown in Figure 3a. *TwoContainer* state signifies there are two containers of that type in the station, *OneContainer* state signifies there is only one container of that type left in the station, and *Empty* state signifies that there is no container of that type in the station. Two variables affect these states: *TimeToEmpty* and *ContainerCondition*. We update the value of *ContainerCondition* in two places i.e. when containers carried by robot reach stations that need the same container, and in agent population *containers* (which signifies the consumption of Lego bricks in the container). The Robot Behavior part replenishes the containers and updates their state. So, the value and state of *ContainerCondition* are connected. The states in the *containers* update the *Deadline* for each kind of container. This part also updates the *NoOfContainersReplenished* during replenishment for the calculation of *PayloadUtilization*.



(a) State chart of Container Behaviour in Stations.



(b) State chart of Container Sorter.

Figure 3: ABM parts for our model.

In Figure 3a, *TwoContainer_Start* and *TwoContainer* have the same function. The only difference is the assignment of *Deadline* variable. At the start of the simulation, *Deadline* can be assigned as 2 * *TimeToEmpty*, as there are two containers available in the stations. But later when the containers go to *TwoContainer* from *OneContainer* state, it also has to consider how long the container was in the state of *OneContainer* before it went to *TwoContainer*. For example, if we have only one container of that type left in the station, that filled container will only last for 320 seconds. Robot makes the delivery of its second container when the container in use is half used. So it would have lasted the next 160 seconds when the delivery was made. After the container is delivered, the new deadline should be $160 + 320 + currenttime$. We model this in *TwoContainer* state. If *TwoContainer_Start* was used, the new deadline would be $320 * 2 + currenttime$, which doesn't represent accurate behavior. This behavior is only required in the initial state. Also, we used state *transit* in a model to delay the process until all the agents get the value of *TimeToEmpty* from the *Start* event before we assigned the default common value.

3.3 Container Sorter

In this part (Figure 3b), we sort names of containers based on *Deadlines* and insert the required container names into *RobotContainer* (collection). Additionally, we update *NoOfTotalAvailableContainers* based on values of *ContainerCondition*. The Container Sorter includes a single state that updates every half second. This also assigns updated values of deadlines and available numbers of each number to the variables, helping in the validation of the model. Since these tasks use dynamic decision-making, Container Sorter uses ABM.

3.4 Robot Behavior

The Stretch robot carries containers from the home position and delivers them to different stations in an assembly cell within a specific time or requirements. We model this behavior of our Stretch robot with a DES approach as shown in Figure 4, since the DES approach works better on modeling process-centric behavior and makes the validation process more achievable. Here, we represent the Stretch robot as an entity, and the Stretch robot's behavior acts as an environment for *containers* agent population, making the robot behavior simple to model. There is only one Stretch robot in the system. So, the source creates only one Stretch robot, and since there is no sink, the same robot moves through the loop. If the condition for replenishment is met, the Stretch robot follows the path D-A-B-C path and returns to the home position. If the Stretch robot is carrying the same container that the station is looking for and the station has less than two containers of that type, we increase the value of *ContainerCondition* for that container by 1.

Initially, all the stations are full of containers. After the simulation or production run has started, it takes more time to get the demand of containers for the first shift. To model this behavior accurately and level the *RobotWaitTime*, we used *initialDelay* block as shown in Figure 4. From Table 3, the nearest time for the first container to be empty is 280 seconds. So, the Stretch robot has to start its first shift before 280 seconds. After some simulation runs with different numbers, we choose the first waiting to be 240 seconds which levels the *RobotWaitTime*. Therefore, the delay time for *initialDelay* block is $240 - RobotWaitTime$, because the robot waits again in *Waiting* block for the delay value of *RobotWaitTime*.

We obtain results by running the optimization experiment and varying the value of *CarryLimit* for each experiment. Subsequently, we generate different graphs to represent the outcomes of these experiments. The optimization parameters for the experiments are discussed in section 5.

4 VERIFICATION AND VALIDATION

This section provides information about how we ensured that our model is correct (verification) and that the model represents our real system (validation). For verification of the model, we used intermediate variables shown in Table 1. *RobotContainer* represents the containers carried by a Stretch robot during

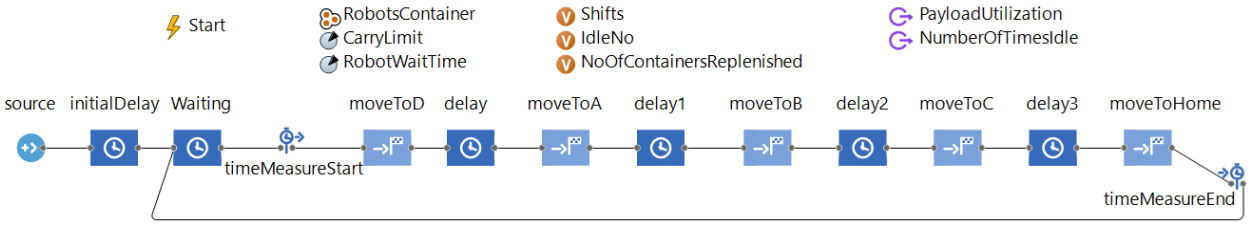


Figure 4: DES logic of Robot Behavior.

the replenishment process. During the simulation, whenever the Stretch robot reaches Zone A, B, C, or D, we update *Deadlines* and *ContainerCondition* of the containers that were included in a collection *RobotContainer*, to verify our model. During each container delivery, we estimate the expected values of intermediate variables. If the model does not meet our expectations, we analyze the discrepancies to identify the causes and make necessary corrections. We also monitor the replenishment conditions in the console to ensure that the Stretch robot initiates its replenishment shift as expected. We validated our replenishment system using a base model. We performed a two-sample t-test to compare the average of forty-six simulation data points with the average of thirty-three real observations of the trip time of the Stretch robot in a shift, under the null hypothesis that the two means are equal. The results indicate that the difference between the two means is not statistically significant. Therefore, we consider the model to be valid. After we validated the base model, we tested alternative scenarios, explained in section 5.

5 RESULTS AND DISCUSSION

This section explains the optimization parameters and experiment results for the base model and the alternative scenarios. Here, we want to find out the different parameters to help us plan the implementation of AMR. Scenario 1 considers the different *CarryLimit* of containers or payload for a Stretch robot, Scenario 2 tests the full utilization of payload for each shift, and Scenario 3 explores maximizing shift utilization and meeting immediate needs. Each scenario is built on a verified and validated base model and includes a sensitivity analysis with varying *CarryLimit* values. In all scenarios, we assume sensors track the state of each container, allowing us to reduce the number of containers the robot needs to carry. This is beneficial because the Stretch robot may sometimes be unable to carry all containers due to weight or volume limitations. The multiple-part feeder or robotic arm in the home decides which containers to put on the Stretch robot based on sensor data. For Scenario 2 and 3, we need to adjust the DES part of the base model as shown in Figure 5. Here, we added a loop in a *Waiting* block so that the replenishment is only triggered when the condition is satisfied. For Scenario 2, the condition is $NoOfTotalAvailableContainers <= (20 - CarryLimit)$, whereas for Scenario 3, the condition is $NoOfTotalAvailableContainers <= (20 - CarryLimit) || (NearestDeadline - time()) < RushTime$, where $time()$ returns current simulation time. Other parts of the model are the same as the base model. More details about the changes are explained below.

5.1 Base Model: Optimization of *RobotWaitTime* for Each Shift

First, we explore a simple model where a robot carries all ten containers and delivers them in a specific time gap. In the base model, it is not required to decide what container to deliver or to know what container is required to be replenished. The robot just replenishes containers after the allocated time difference. The experimental parameters for the base model are shown in Table 4.

5.2 Scenario 1: Consideration for Carry Limit of Mobile Robot

In this scenario, the objective is to minimize the number of shifts for healthier battery life while optimizing the container capacity the robot should carry. A trade-off between these two factors is necessary. The

Table 4: Experimental Parameters.

Number of Replication	100
Optimization engine	Genetic
Objective	Minimize $root.Shifts$
Parameter	$CarryLimit$: Fixed (9,8,7,6,5,4,3 for each run)
Requirement	$root.NumberOfTimesIdle <= 0$ (*NumberOfTimesIdle* is the final value of *IdleNo*)

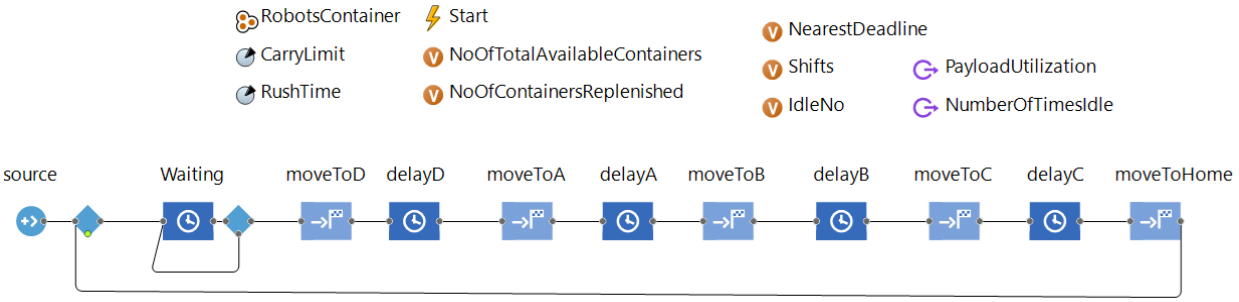


Figure 5: Modified model for Scenario 2 and 3.

additional optimization parameter for Scenario 1 is, *RobotWaitTime* (described in Table 1), which is defined as:
RobotWaitTime : Discrete (minimum:0, maximum:150, step:2).

5.3 Scenario 2: Full Utilization of Each Shift

Scenario 2 enables the Stretch robot to decide when to replenish the containers. If the container carrying capacity of a Stretch robot is fixed to six, the Stretch robot only starts replenishing the containers when there is a demand for at least six containers. This system aims to ensure that the shift of the Stretch robot is fully utilized. The experimental parameters for this model are the same as the base model. However, this scenario did not provide any feasible solution. So in our case, it is not possible to make the Stretch robot wait long enough to get the demand of containers to be greater than or equal to *carryLimit*. There is a high chance of a station being idle.

5.4 Scenario 3: Maximizing Shift Efficiency and Meeting Immediate Needs

In this scenario, the Stretch robot can start replenishment either when the demand for containers is more than or equal to *CarryLimit* of the Stretch robot (similar to Scenario 2), or when any deadline of the container is very close. This ensures that there is very little chance of a station being idle, and provides full utilization of the Stretch robot's carrying capacity. The additional optimization parameter for Scenario 3 is *RushTime* (described in Table 1), which is defined as:
RushTime : Discrete (minimum:20, maximum:300, step:2).

The results from all feasible scenarios are shown in Figure 6. As illustrated in the plot 'a' of Figure 6, Scenario 1 is generally preferable as it requires fewer shifts for replenishment, except when the *CarryLimit* is five. In that case, Scenario 3 becomes the better option. For Scenario 1, if the *CarryLimit* is between 6 and 10, the number of shifts required for replenishment remains at 8. However, if the *CarryLimit* is 5, the number of shifts increases to 9, and for a *CarryLimit* of 4, it increases to 10. In contrast, Scenario 3 consistently requires 9 shifts for *CarryLimit* values from 5 to 10, and 11 shifts for a *CarryLimit* of 4. Additionally, the first plot indicates that a *CarryLimit* of less than 4 is not feasible, as it would cause the

stations to become idle. Therefore, optimizing the *CarryLimit* is crucial for minimizing shifts and ensuring operational efficiency.

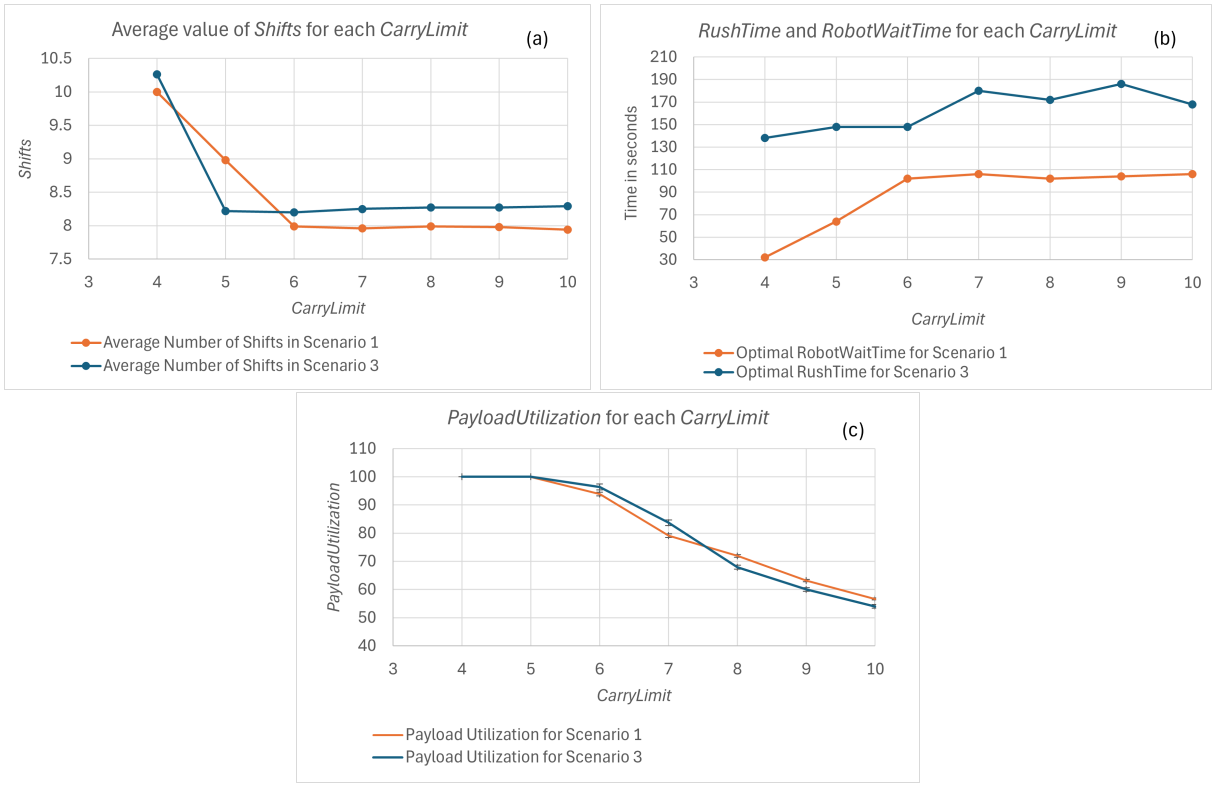


Figure 6: Experimental Result.

The second plot, 'b', in Figure 6 identifies the optimal parameters to use for different *CarryLimit* values based on the chosen scenario. For instance, with a *CarryLimit* of 5, Scenario 3 is optimal since it requires fewer shifts, with a *RushTime* of 148 seconds. Similarly, plot 'c' illustrates the payload utilization across various scenarios and *CarryLimit* values. This plot is based on the results of one hundred replications in the parameter variation experiment, using the optimized parameters from plots 'a' and 'b'. For a *CarryLimit* of 8, 9, and 10, Scenario 1 demonstrates better payload utilization, while for a *CarryLimit* of 6 and 7, Scenario 3 is more efficient. For other *CarryLimit* values, the payload utilization is comparable across scenarios.

The plots in Figure 6 provide crucial data to guide our strategy for implementing a mobile robot in our sub-assembly plant. For instance, if our Stretch RE1 is limited to carrying 6 containers, plot 'a' suggests selecting Scenario 1 due to its lower number of required shifts for replenishment. On the other hand, according to plot 'c', Scenario 3 achieves better payload utilization. Therefore, the decision hinges on our priorities. Suppose minimizing the number of shifts takes precedence over payload utilization. In that case, Scenario 1 is the preferred choice. Implementing Scenario 1 requires setting an appropriate wait time at the home position before each shift begins. In our case, this wait time is approximately 102 seconds. Thus, we must program Stretch RE1 accordingly, ensuring it waits 102 seconds at the home position before initiating each repetitive shift.

6 CONCLUSIONS AND FUTURE WORKS

In this study, we explore planning the implementation of an AMR into an existing material replenishment system within assembly plants using hybrid simulation. The primary objective is to assess different scenarios

for optimizing the containers a robot should carry per shift, with a focus on minimizing travel shifts while ensuring timely material replenishment to eliminate idle time across all stations. From the experiments, we find that the decision to choose between strategies highly depends on the payload limit of AMR. By the use of a hybrid model, we are also able to get optimized parameters for different strategies and compare the strategies using a key performance indicator or parameter.

Initially, for Scenario 1, we test a simple model where an AMR carries a fixed number of containers, replenishing them at specific time intervals. Building upon this, Scenario 2 aims at optimizing the robot's shift utilization and minimizing wasteful trips. However, Scenario 2 does not provide a feasible solution. Finally, Scenario 3 introduced the consideration of approaching deadlines for containers, prompting the robot to act proactively and start delivering containers when a container's deadline approached, while also trying to maximize the utilization of payload in the AMR. This dynamic approach aims at reducing the chances of stations being idle and providing flexibility in determining the optimal container capacity for the robot. For the payload limit of five, Scenario 3 gives a better result whereas, for other limits, Scenario 1 is a better choice if we only consider minimizing the number of shifts. If we also consider payload utilization, things change. These results help in comparing and selecting a plan or strategy to implement an AMR in a sub-assembly cell. From the experiment, we also get the optimized parameters for each scenario, which can be used during the implementation of the selected plan.

The findings of the study provide valuable insights for Tiger Motors lab to implement AMR for material replenishment. The results indicate that planning the implementation of AMR using hybrid simulation can help us compare different strategies, and select the best one based on our requirements for different performance measures. This methodology can also help assembly-plant managers and engineers to adapt hybrid simulation for planning the implementation of AMR in their assembly plant.

Although there are many benefits of this study, the study is limited in using a single AMR and a single station of Tiger Motors lab. Also, the study considers that all the containers are of the same size. These boundaries set a limitation to help managers and engineers of other assembly plants for their adoption of this methodology that has different requirements than ours.

Future work in this domain could explore more on using multiple AMRs and consider whole sub-assembly cells of the assembly plant. The research can also expand to consider the variability of the product, where the assembly plant produces different models of Lego cars.

REFERENCES

Araya, F. 2022. "Integration of discrete event simulation with other modeling techniques to simulate construction engineering and management: an overview". *Revista de la construcción* 21(2):338–353.

Arora, K. C. and V. V. Shinde. 2007. *Aspects of materials handling*. Firewall Media.

Caro, J. J. and J. Möller. 2016. "Advantages and disadvantages of discrete-event simulation for health economic analyses". *Expert review of pharmacoeconomics & outcomes research* 16(3):327–329.

Carson, J. S. 2005. "Introduction to modeling and simulation". In *Proceedings of the Winter Simulation Conference, 2005.*, 8–pp. IEEE.

Conn, A. R., L. A. Deleris, J. R. Hosking, and T. A. Thorstensen. 2010. "A simulation model for improving the maintenance of high cost systems, with application to an offshore oil installation". *Quality and Reliability Engineering International* 26(7):733–748.

Datta, S. and R. Ray. 2007. *AMR vision system for perception, job detection and identification in manufacturing*. INTECH Open Access Publisher.

Dubiel, B. and O. Tsimhoni. 2005. "Integrating agent based modeling into a discrete event simulation". In *Proceedings of the Winter Simulation Conference, 2005.*, 9–pp. IEEE.

Hentout, A., B. Bouzouia, I. Akli, and R. Toumi. 2010. "Mobile manipulation: a case study". *Robot manipulators, new achievements*:145–167.

Khodabandelu, A. and J. W. Park. 2022. "Applications of agent-based modeling (ABM) in enhancing facility operation and management".

Kousi, N., G. Michalos, S. Makris, and G. Chryssolouris. 2016. "Short–term planning for part supply in assembly lines using mobile robots". *Procedia CIRP* 44:371–376.

Liu, J., B. Sun, G. Li, and Y. Chen. 2023. "An integrated scheduling approach considering dispatching strategy and conflict-free route of AMRs in flexible job shop". *The International Journal of Advanced Manufacturing Technology* 127(3):1979–2002.

Moretti, E., E. Tappia, and M. Melacini. 2021. "Scheduling mobile robots in part feeding systems". In *Hamburg International Conference of Logistics (HICL) 2021*, 129–149. epubli.

Mykoniatis, K. and A. Angelopoulou. 2020. "A modeling framework for the application of multi-paradigm simulation methods". *Simulation* 96(1):55–73.

Nielsen, I., Q.-V. Dang, G. Bocewicz, and Z. Banaszak. 2017. "A methodology for implementation of mobile robot in adaptive manufacturing environments". *Journal of Intelligent Manufacturing* 28:1171–1188.

Pappert, F. S., E. Angelidis, and O. Rose. 2010. "Framework for simulation based scheduling of assembly lines". In *Proceedings of the 2010 Winter Simulation Conference*, 1690–1698. IEEE.

Vieira, M., A. P. Barbosa-Póvoa, S. Moniz, and T. Pinto-Varela. 2018. "Simulation-optimization approach for the decision-support on the planning and scheduling of automated assembly lines". In *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*, 265–269. IEEE.

Villarreal, B. and M. d. R. Alanís. 2011. "A simulation approach to improve assembly line performance.". *International Journal of Industrial Engineering* 18(6).

Waseem, M. and Q. Chang. 2023. "Adaptive Mobile Robot Scheduling in Multiproduct Flexible Manufacturing Systems Using Reinforcement Learning". *Journal of Manufacturing Science and Engineering* 145(12).

Zheng, Z., Z. Chang, and Y. Fei. 2017. "A simulation-as-a-service framework facilitating webgis based installation planning". *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42:193–198.

Zhou, B. and J. Xu. 2017. "An adaptive SVM-based real-time scheduling mechanism and simulation for multiple-load carriers in automobile assembly lines". *International Journal of Modeling, Simulation, and Scientific Computing* 8(04):1750048.

AUTHOR BIOGRAPHIES

RUPESH BADE SHRESTHA is a PhD student in the Department of Industrial and Systems Engineering, at Auburn University. He holds a Bachelor of Engineering in Industrial Engineering from the Institute of Engineering, Thapathali Campus, Tribhuvan University. His research interests are Modeling and Simulation, Industry 4.0, Lean Systems, and Reinforcement Learning. His email address is rzb0117@auburn.edu.

ELLIE HUNGERFORD is a senior undergraduate student in the Department of Industrial and Systems Engineering, at Auburn University. Her email address is ekhungerford@gmail.com.

KONSTANTINOS MYKONIATIS is an Assistant Professor in the Industrial and Systems Engineering department at Auburn University. He holds an engineering B.Eng. in Production Engineering and Management from the Technical University of Crete and an M.S. and Ph.D. degrees in Modeling and Simulation from the University of Central Florida (UCF). Prior to joining Auburn University, he worked as an Assistant Professor at Wichita State University and as a simulation instructor at the Industrial Engineering and Management Systems at UCF. In addition to his academic positions, he has held professional engineering positions as a modeling and simulation professional in the Business Analytics and Industrial Engineering Division of Universal Orlando Parks and Resorts, the Institute for Simulation and Training, the Institute for Advanced Systems Engineering, and the Public Power Corporation S.A Hellas. His primary research interests are in the areas of hybrid modeling and simulation, advanced manufacturing systems, robotics, and automation. His email address is kmykoniatis@auburn.edu.