Dynamic Scheduling for Heterogeneous Resources with Time Windows and Precedence Relation

Shudong Liu†, Ernest Kurniawan†, Peng Hui Tan†, Peng Zhang‡, Sumei Sun†, and Sarah Ye‡

†Institute for Infocomm Research, 1 Fusionopolis Way #21-01 Connexis (South Tower), Singapore 138632
‡Procter & Gamble (Guangzhou) Ltd., 2-4 Aether Square 986 Jie Fang Bei Road, Guangzhou, China 510075

Email: (liush, ekurniawan, phtan, zhangp, sunsm)@i2r.a-star.edu.sg, ye.sa.1@pg.com

Abstract—We consider dynamic scheduling for heterogeneous resources in flexible manufacturing systems. The resources are mobile robots with different capabilities and/or people with different skills. The tasks in the system arise over time and each has a time window for completion. We have developed a method for dynamic scheduling based on mixed integer programming and rolling horizon. The results show it can solve practical scheduling problems fast, and can significantly reduce the cost when compared to current practice.

I. INTRODUCTION

Robots have been used in industries for a few decades, but they are mainly robotic arms with a static foundation. With the technological development of robotics, more and more companies ([11], [2]) start to adopt mobile robots to increase flexibility in manufacturing and/or service systems, improve throughput, increase product quality and reduce costs. These mobile robots can execute complex tasks such as transporting materials from one place to another autonomously navigated. One key issue to effectively utilize mobile robots is an appropriate scheduling method/tool which can deal with situations such as complex constraints/requirements in realistic environment.

In this paper we consider dynamic scheduling for heterogeneous resources in manufacturing environment. The resources are robots and different robots may have different capabilities due to requirement of manufacturing environment. For example, some robots mainly transport material from one place to another, while some other robots may do quality check for finished products. During the production process, a lot of tasks such as transporting materials arise over time. Each task has its own time window, failing of compliance will result in production interruption. In addition, some tasks have inter-dependence such that a task has to be finished before another task can start. We need to schedule these resources to execute the tasks in an economic way while satisfying a variety of constraints. We call this scheduling problem as Dynamic Scheduling for Heterogeneous Resources with Time Window and Precedence Relation (DSHR-TWPR). To our best knowledge, there is no previous literature to address this type of problem yet, though there is some related work in Vehicle Routing Problem (VRP) and Multiple Robot Task Allocation (MRTA).

We have developed an innovative method to dynamically schedule the heterogeneous resources. We first develop a flexible Mixed Integer Programming (MIP) model to address complex requirements/constraints, then combine it with rolling horizon and real-time information about factory status to dynamically schedule for long-term production. This method has advantages of both efficiently addressing very complex situations/constraints and fast obtaining near-optimal solutions. Comparing with current practice in a factory, it can save traveling time by more than 40% for robots whose main tasks are transporting materials around machines.

The remainder of the paper is organized as follows. In Section II, the literature review is presented. Problem statement is described in Section III. In Section IV, scheduling over a fixed horizon is formulated as a mixed integer programming problem. It is then extended to long horizon in Section V. A case study is reported in Section VI. Finally, conclusions are given in Section VII.

II. LITERATURE REVIEW

There is a very rich literature about scheduling since early papers such as Johnson’s in 1954 [3] which is to minimize makespan for a two-machine flow shop. A rich variety of problem types have been addressed and many mathematical programming models are developed. Due to computational complexity (most are NP-hard/NP-complete) of scheduling, many heuristics and meta-heuristics are developed. Potts and Strusevich [4] provide a comprehensive literature about scheduling until 2009.

Scheduling with robots appear in the early papers such as [5] and [6]. Bedini et al. develop a heuristic to determine sequencing to optimize working cycle of a robotic cell. Kondoelon uses simulation to analyze the effects of configurations on the cycle time. Since then, a lot of papers consider the sequencing and scheduling in robotic cells ([7], [8], [9]). For a comprehensive review, refer to [10]. Note that the robots in above literature are not mobile. They often sit in a robotic cell to serve multiple machines.

The problem type of Multiple Robot Task Allocation (MRTA) is most close to our problem (DSHR-TWPR). In MRTA, the objective is to find the task-to-robot assignment to achieve the overall system goals. In general, there are
two approaches to assign robots to tasks: market-based and optimization-based. In the market-based approach, robots bid for tasks based on their capabilities and the negotiation process is based on market theory of auction. Optimization-based approach uses optimization techniques such as mixed integer programming, simulated annealing, genetic algorithms, taboo search and so on. The optimization-based approach can get better solutions than the market-based approach. In MRTA, Darrah et al. ([11]) consider task allocation problem in the context of UAV cooperation, using the mixed integer linear programming approach. Kim et al. ([12]) consider the scheduling of a fleet of UAVs which need to refuel at service stations. They develop an mixed integer programming model and a genetic algorithm to solve it. Note that in MRTA, in general, the precedence relation among tasks and time window are not considered.

Another stream of related literature is the vehicle routing problem with time window. In this type of problem, the objective is to minimize traveling distance of multiple mobile agents, considering time window. However, the precedence relation among tasks are not considered either. For a comprehensive review, refer to [13].

III. PROBLEM DESCRIPTION

Consider a factory which has multiple machines. The machines are arranged as a flow shop. Raw materials are transported from a warehouse to a common buffer area of machines, and then send to feeders of machines. The machines transform materials into final products. Final products are then sent to final product warehouse. An example of layout of the factory is shown in Figure 1.

In the factory, there are many types of tasks such as transporting materials from one place to another, putting material to feeders of machines, checking product quality, packaging and so on. Some tasks have precedence relation. For example, we need to first transport materials to a place near a feeder, then it is possible to put the material into the feeder. Each task has its own time window which should be strictly followed, otherwise the production line may have to stop due to missing material or bad quality of products.

There is a fleet of robots with different capabilities. Some robots are mainly designed for transporting materials, and some can be more flexible to handle a few types of tasks. In normal production, the machines run at constant speeds, and the production tasks are regular and predictable. Sometimes unusual events may happen, for example, not passing quality check and machine failure. The status of factory shop floor is continuously monitored by an information system, and the scheduling and control is managed by a central command server.

We need to develop an effective scheduler to schedule robots to do the tasks over time while satisfying all constraints. The production is a continuous process and the planning horizon can be very long (one shift of eight hours or days). In the following sections, we first consider scheduling robots over a fixed short planning horizon (e.g. less than one hour due to limits of computational time or available information for future tasks), then extend it to long production period (e.g. a shift or days) using rolling horizon.

IV. SCHEDULING OVER A FIXED HORIZON

In this section we formulate the scheduling as a mixed integer programming problem.

A. Assumptions and notations

Assume each robot can do only one task at a time, and each task needs only one robot. For each task, there is a start location and an end location. The start location of a task is not necessary to be the same as its end location. The factory map is available to the scheduler, hence the distance and shortest path from one location to another is known. The robots’ speeds are given and assumed to be constant. So, for each robot, the traveling time from one location to another can be obtained (assuming the shortest path is used). Following are the notations for sets, parameters and variables in the problem.

\( R_{type} \), set of resource types.
\( N_{R}^{r}, r \in R_{type} \), number of available resources in each resource type \( r \).
\( I_{R}^{r} = \{1, \ldots , N_{R}^{r}\}, r \in R_{type} \), set of index of resources of type \( r \).
\( \{r, i\}, r \in R_{type}, i \in I_{R}^{r} \), index for resource.
\( R^{r} = \{(r, i)| i \in I_{R}^{r} \}, r \in R_{type} \), set of resources of type \( r \).
\( \bar{R} = \cup_{r} R^{r} \), set of all resources of all types.

\( A_{type} \), set of all task types including transporting materials, quality check and so on. Note that each task type has its own start location and end location. So different locations of tasks mean different task types.
\( N^{a}_{A}, a \in A_{type} \), number of times of repeating for task type \( a \). Each task type may repeat many times which depends on the length of planning horizon and other factors such as material consumption speeds.

\( I_{A}^{a}_{begin}, a \in A_{type} \), index of the first task of type \( a \). It is 1 by default. It is used for generalization of model for considering some special cases such as sliding windows for tasks.

\( I_{A}^{a} = \{I_{A}^{a}_{begin}, \ldots , I_{A}^{a}_{begin} + N^{a}_{A} - 1\}, a \in A_{type} \), set of index of tasks of type \( a \).

\( (a, i), a \in A_{type}, i \in I_{A}^{a} \), the task which is of task type \( a \) and is the \( i_{th} \) index in the group of type \( r \).

\( A = \cup_{a} A^{a} \), set of all tasks of all types.

\( P_{re} = \{(a, i, 1, a2, i2)| (a, i), (a2, i2) \in A \}, \) set of precedence relation among tasks, which means task \( (a2, i2) \) cannot start until task \( (a1, i1) \) is finished. Here \( i1 \) may be different from \( i2 \).

\( T_{R_{a},i} \), release time of task \( (a, i) \). The task cannot be started before this time.

\( T_{D_{a,i}} \), deadline time of task \( (a, i) \). The task must be finished before this time.

\( T_{P_{a}}^{r} \), processing time of task type \( a \) by resource type \( r \).
\( QU_r^a \in \{0, 1\} \), indicator for qualification of resource type \( r \) for task type \( a \); 1 if resource type \( r \) is qualified for task type \( a \), 0 otherwise.

\( TC_{a1,a2}^r, r \in R_{type}, a1,a2 \in A_{type} \), changeover time for resource type \( r \) from task type \( a1 \) to task type \( a2 \). The changeover time is the traveling time for the resource to travel from the end location of task type \( a1 \) to the start location of task type \( a2 \).

\( M \), a very big number used for modeling, which is larger than any time (expressed in real number) in the system.

\( ST_{r,a}^\beta \in \{0,1\}, (r,a) \in R,A \in A_{type} \), initial state in task type of resource \((r,a)\), i.e., the task type of the task that the resource has last finished or is doing. \( ST_{r,a}^\beta = 1 \) if resource \((r,a)\)'s last finished task or the working task (before running the current resource allocation) is of type \( r \), and \( ST_{r,a}^\beta = 0 \) otherwise.

\( ST_{r,j} \in R, (r,j) \in R \), initial state in earliest available time of resource \((r,j)\), \( ST_{r,j}^\beta = 0 \) if the resource has finished its last task before the beginning of planning horizon, otherwise \( ST_{r,j}^\beta \) is the remaining time for finishing its current task so that it can take new tasks. Initial state of robots at the scheduling epoch consists of \( ST_{r,a}^\beta \) and \( ST_{r,j}^\beta \). From this initial state, we can know the initial location of each resource. The resource allocation needs to satisfy some time constraints that the resource has enough time to move from its initial location to the place where its first task is.

Decision variables are as follows:

\( x_{r,a}^{i,j} \in \{0,1\}, (a,i) \in A,(r,j) \in R \). It is 1 if resource \((r,j)\) is assigned to do the task \((a,i)\), 0 otherwise.

\( s_{a,i} \), starting time of task \((a,i)\). From \( x_{r,a}^{i,j} \) and \( s_{a,i} \), the scheduling of robots is completely determined, but the model still needs other variables. For example, we need variables to extract traveling time/distance of robots.

\( N_{R}^{used} = \sum_{r \in R_{type}} N_{r}^{a} \), total number of resources used.

\( N_{A} = \sum_{a} N_{A}^{a}, a \in A \), the total number of tasks.

\( \beta_{a1,i1}^{a2,i2} \in \{0,1\}, (a1,i1),(a2,i2) \in A, 1 \) if the same resource first does task \((a1,i1)\), and then does task \((a2,i2)\) (i.e. there is no other tasks between these two tasks), 0 otherwise.

\( \beta_{a1,i1,i2}^{a2,i2} \in \{0,1\}, (a1,i1),(a2,i2) \in A, (r,j) \in R, 1 \) if the resource \((r,j)\) first does task \((a1,i1)\), and then does task \((a2,i2)\) (there is no other tasks between these two tasks), 0 otherwise.

\( cd_{a1,i1,i2}^{a2,i2} \in \{0,1\}, (a1,i1),(a2,i2) \in A \), the traveling time (changeover time) from task \((a1,i1)\) to task \((a2,i2)\). It is the true changeover time from task \((a1,i1)\) to task \((a2,i2)\) of a resource if both tasks are done by the same resource and \( \beta_{a1,i1}^{a2,i2} = 1 \), 0 otherwise.

\( di_{a1,i1,i2}^{a2,i2} \in \{0,1\}, (a1,i1),(a2,i2) \in A, (r,j) \in R \), the traveling time (changeover time) of resource \((r,j)\) from task \((a1,i1)\) to task \((a2,i2)\). It is the true changeover time from task \((a1,i1)\) to task \((a2,i2)\) if \( \beta_{a1,i1}^{a2,i2} = 1 \), 0 otherwise.

\( w_{a,i}^{r,j} \in \{0,1\}, a \in A_{type}, (r,j) \in R \), 1 if the first task of type \( a \) is assigned to resource \((r,j)\), 0 otherwise.

\( id_{r,j} \), \((r,j) \in R \), the traveling time (changeover time) of resource \((r,j)\) from the initial state to its first task.

Based on the above sets, parameters and variables, we can build the mathematical programming model.

**B. The MIP Model**

The number of resources of each type is already given, which is determined by production requirement. Now we schedule the given resources to finish the tasks. The objective of optimization is to minimize total traveling time of resources, assuming the speeds of all robots are the same. It is equal to minimize the total traveling distance. So, the mixed integer programming for the scheduling problem is as follows:
min \[ \sum_{(a,i),(a,j) \in A} cd^{a_{r},i}_{a_{r},j} + \sum_{(r,j) \in R} id_{r,j} \] \hspace{1cm} (1)

s.t. \[ \sum_{(r,j) \in R} x^{r,j}_{a_{r},i} = 1, \forall (a,i) \in A \] \hspace{1cm} (2)

\[ x^{r,j}_{a_{r},i} = 0, \forall (a,i) \in A, (r,j) \in R \text{ s.t. } QU_{a,0} = 0 \] \hspace{1cm} (3)

\[ TR_{a,i} \leq s_{a,i}, \forall (a,i) \in A \] \hspace{1cm} (4)

\[ s_{a,i} + \sum_{(r,j) \in R} (TP^{r}_{a_{r},i} - x^{r,j}_{a_{r},i}) \leq TD_{a,i}, \forall (a,i) \in A \] \hspace{1cm} (5)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (6)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (7)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (8)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (9)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (10)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (11)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (12)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (13)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (14)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (15)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (16)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (17)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (18)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (19)

\[ \forall (a,i), (a,j) \in A \text{ s.t. } QU_{a,0} = 0 \text{ or } QU_{a,2} = 0 \] \hspace{1cm} (20)

The objective (1) is to minimize total traveling time (changeover time) including time from the initial location to the first task of each resource. Constraint (2) is to make sure that for each task there is one resource assigned to it. Constraint (3) is to ensure not to assign resources to the tasks if these resources are not qualified for these tasks. Constraints (4-5) are about time windows of tasks. They enforce that the start time of a task must not be earlier than the release time of the task, and the task should be finished before its deadline. Constraint (6) is to make sure the precedence relation between some tasks.

Constraint (7) is about the initial state, which is to make sure there is enough time for changeover from initial state to the first task for each resource. Constraint (8) indicates that if a resource is not qualified to do a task, then it is not possible for the resource to change from this task to another task or from other tasks to this task. Constraint (9) ensures a resource cannot transfer a task to itself. Constraint (10) defines the total transferring from one task to its immediate next task. Constraint (11) ensures that for each task, there is at most one immediate following task. Constraint (12) ensures that for each task, there is at most one immediate precedent task. Constraint (13) implies that there are \( N^{used}_{R} \) tasks which have no immediate following task, as each one of \( N^{used}_{R} \) resources will have one last task which has no immediate following task.

Constraints (14-15) are to calculate the traveling time of a resource. Constraint (16) ensures the starting time of task \((a_{2},i_{2})\) is greater than the ending time of task \((a_{1},i_{1})\) plus the traveling time if \( \beta^{a_{2},a_{2}}_{a_{1},i_{1}} = 1 \).

Constraints (17-20) are to extract traveling time from the initial location to the first task of each resource. Constraint (17) ensures that if the resource is not qualified to do a type of task, then we cannot assign this type of task as the first task of this resource. Constraint (18) ensures for each resource, there is one and only one first task assigned to it. Constraint (19) ensures the allowed traveling time from initial location to its first task is large enough for the changeover needed. Constraint (20) ensures if a task is the first task of a resource \((r,j)\), then there is no other task in \( A \) precedes it on this resource.

V. DYNAMIC SCHEDULING USING ROLLING HORIZON

In the above MIP model, the task list is given and planning horizon is fixed. The planning horizon cannot be very long. One reason is that the longer the horizon, the more the computational time to get a solution for the MIP model. Another important reason is that it is not easy to have accurate prediction for future tasks, as there is some uncertainty in production system such that some unusual tasks may arise, or the time windows may not be as long as expected. So, even if we get a schedule for a long horizon, the system needs to reschedule again to respond to dynamic situations. Hence, longer horizon is not necessarily better. On the other hand, if the planning horizon is too short, then the optimization model works like an instant dispatching policy and there is not much degree of freedom for optimization. As such, we need to choose an appropriate planning horizon for the above MIP model.
However, the production is a continuous process and we need to schedule tasks over the whole production period. We use the rolling horizon method to dynamically schedule robots for the tasks: at each scheduling epoch, the shop floor monitoring system provides the scheduler the future tasks over a certain period (a fixed length horizon) ahead and the current status of robots, then the scheduler runs the above MIP model to get the schedule. With the time passing and at the next scheduling epoch, the above procedure repeats. The scheduling epochs can be periodic or event-driven or hybrid.

It is well known that MIP problems are difficult to solve and its long computational time is a serious issue to hinder its practical usage. Hence many researchers develop heuristics to get a feasible solution within a reasonable time. However, heuristics such as genetic algorithms are less capable to address complex constraints, even in finding feasible solutions. In our study, we use general purpose commercial solvers such as CPLEX to solve the MIP model to obtain good schedules.

VI. NUMERICAL RESULTS

In this section we report a case study based on realistic production situation to check the feasibility and benefits of our approach. The plant layout is shown in Figure 1. From the figure, there is a raw material warehouse, a finished product warehouse, multiple workstations and buffer areas. Different types of robots move around. Raw materials first are transported to the buffer areas and then are moved to feeders of work stations. Semi-finished products are moved from one station to its next station. Finished products are transported to finished product warehouse.

In the factory practice, they use heuristic dispatching rule to assign robots to tasks. The dispatching rule is mainly based on First-Come-First-Served (FCFS) policy and human experience. Note that the pure FCFS policy is not feasible, as there are some precedence constraints for tasks.

In order to investigate the performance of our method, we developed a heuristic as follows, which is better than the dispatching rule in practice. The heuristic scheduling algorithm is called Least-Laxity-First (LLF) [14]. The laxity is defined as the deadline minus the processing time of a task. In other words, it is the latest starting time of the task without missing the deadline. LLF scheduling is superior to FCFS policy and Earliest-Deadline-First (EDF) policy since it takes both the deadline and processing time into account. In the LLF heuristic, the precedence condition is checked before assigning any resource to any task. Resource allocation for a task with the least laxity would be postponed if any of its precedent tasks is not completed yet. This is to ensure that the resultant schedule does not violate any precedence constraint. In addition, when multiple resources are available for a task, we assign the one with the least changeover time to process the task.

In the system, there are 7 robots of 4 types. One robot of type 1 (denoted as R1-1) is for transporting materials from warehouse to the buffer area before workstations. Another robot of type 2 (denoted as R2-1) transports finished products to finished product warehouse from end of production line. Two robots of type 3 (denoted as R3-1 and R3-2) are for packaging near end of production line. Three robots of type 4 (denoted as R4-1 to R4-3) are very flexible which are used to transport materials among workstations and buffer areas and can do quality check and other related tasks.

The MIP models and dynamic scheduling are run on a notebook with Intel Core i7 CPU @ 2.50 GHz and 16G RAM. The planning horizon of each MIP model is 60 minutes and we use a periodic way to rerun the MIP model for a shift (8 hours) of production. The last task assigned to a robot in a rolling horizon window (60 minutes) is the initial state of next rolling horizon window. The MIP model is solved using the solver CPLEX 12.6 and the time limit is set as 5 minutes for each rolling horizon window. The LLF heuristic can get the results in 1 second. Numerical results of both the MIP method and LLF heuristic are shown in Figure 2 and Table I.

Figure 2 is the Gantt chart from MIP model for tasks assigned to each robot. From the figure, we can see that the robot of type 1 (R1-1) does not do many tasks (transporting raw materials). It is due to the fact that this robot has large capacity. Robots of types 2 and 3 do the tasks quite regularly. They work around the end of production line where the finished products are generated at a constant speed. The tasks for type 4 robots are quite complicated, and each robot is quite busy (not much gap between one task and the following task).

Table I shows the traveling time of type 4 robots from MIP method and the LLF heuristic. For other types of robots, the traveling time of both methods is almost the same, because there is almost only one robot to do a type of task (e.g. only one robot can do the task of transporting raw material to the buffer area), hence there is not much space to optimize the scheduling for these types of robots.

From Table I, the traveling time from MIP model is significantly less than that of the heuristic, reduced by more than 40%. So, there is a huge space to improve the performance when there are multiple robots doing the same group of tasks, as compared to the heuristic and current practice.

In the above case study, we do not consider the robot battery charging. In fact, the model can easily address this issue. With the real time information about the robots’ remaining power, we can add a task of charging the robot with a time window, and then run the scheduling model to get the appropriate schedule.

VII. CONCLUSIONS

We have considered dynamic scheduling problem for heterogeneous resources in manufacturing environment where there are a lot of complex constraints such as strict time window and task precedence relations. We have proposed a method for it: first develop a mixed integer programming model and then use it as a building block in a rolling horizon context. Complex constraints are included in the MIP model and hence it is quite flexible to address complex practical situations in realistic production environment. We have also developed an LLF heuristic for dynamic scheduling which
takes both the deadline and the processing time of tasks into account. Numerical results show that the MIP method can obtain good solutions quickly and significantly reduce traveling time of robots, compared to the heuristic method.

There are a few interesting directions for future research, considering the trend that mobile robots will be more and more popular in industries. First, it is useful to schedule resources for multiple production lines simultaneously. Through sharing robots among multiple lines, we may gain more benefit. Obviously, the operations of robots will be more complicated. Second, we may develop methods to further reduce the computational time of MIP models. We may find out the optimal length of rolling horizons. Finally, optimizing the composition of the fleet of robots is also useful.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Crystal Xu Zhen for her effort in data collection and information gathering for the simulation setup.

REFERENCES


