A TASK ALLOCATION AND SCHEDULING FRAMEWORK TO FACILITATE EFFICIENT HUMAN-ROBOT COLLABORATION IN HIGH-MIX ASSEMBLY APPLICATIONS

Jeon Ho Kang[1], **Neel Dhanaraj**[1], **Omey M. Manyar**[1], **Siddhant Wadaskar**[1], **and Satyandra K. Gupta**[1*]
[1]Center for Advanced Manufacturing, University of Southern California, Los Angeles, CA, 90089

ABSTRACT

Automating assembly operations effectively increase efficiency while decreasing the need for humans to perform ergonomically challenging tasks. However, full automation of these tasks is still a work in progress. Leveraging the complementary strengths of humans and robots offers a solution. Humans can handle tasks requiring dexterity by working in a team, while robots undertake routine, supportive roles. However, contingency situations created by task execution failures occur more frequently in high-mix applications because of the high variability in the types of tasks. Work cells that enable collaboration between humans and robots are not likely to be economically viable unless contingency situations are detected and efficiently managed. In such situations, additional contingency tasks must be executed to recover from the contingency, and productively utilizing human agents can help the work cell quickly recover from contingencies. This paper presents a framework for automatically assigning and scheduling tasks to humans and robots to complete multiple assemblies while managing the computational complexity of generating optimal task plans. Additionally, we present methods to generate recovery task plans that effectively resolve those contingencies automatically. In our case study, we present an approach to graphically determine the number of tasks and products to consider with limited computing time.

1 INTRODUCTION

The automation of assembly operations has garnered considerable attention to enhance human productivity and alleviate the need for humans to engage in ergonomically demanding tasks. However, the inherent complexity of many assembly operations precludes their full automation. Recognizing that humans and robots possess complementary strengths, one viable approach to advancing automation in assembly tasks involves the collaboration of human and robot teams. In such an arrangement, humans would handle tasks that demand a high degree of dexterity, while robots would be assigned to execute more routine tasks. Traditional robotic manipulators did not allow humans and robots to collaborate because of safety considerations. Robots had to be enclosed in cages to separate them from humans. This constraint severely restricted the deployment of human-robot teams in manufacturing applications. In most cases, robots could only perform simple tasks such as insertion tasks or pick and place operations, which meant humans had to perform complex assembly operations without assistance from robots. Several advances have been made in robotic manipulators, making them safer for humans. These advances enable the development of collaborative work cells, where humans and robots can work nearby. However, to enable efficient human-robot collaboration, tasks must be appropriately allocated and scheduled in the human-robot team to maintain teaming efficiency.

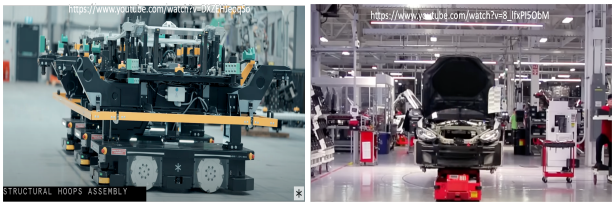

FIGURE 1: Real-world examples of factories that utilize mobile fixtures from the industry. On the left, it shows the arrival's implementation of mobile fixtures in assembly operations, and on the right, it shows Tesla's approach. Both are featured in wired.co.uk.

Robots are currently used for simple assembly tasks in mass-production applications. These tasks are carried out with the help

*Address all correspondence to this author. Email: guptask@usc.edu.

of extensive testing and custom fixtures, which eliminates uncertainty and prevents the possibility of task execution failures. However, this strategy cannot be utilized in low-volume applications due to high variability in task types, leading to contingency situations and task execution failures.

In contingency situations, recovery tasks must be generated effectively, and recovery tasks should be quickly and dynamically recomputed to the current schedule to minimize assembly cell shutdown time while managing failures and optimizing task completion. The efficient utilization of human agents can aid in quickly recovering the work cell from such situations. These plans may result in increased complexity and significant deviation from the original schedule, requiring a new schedule to be computed. However, limited computational resources and a fixed computation time budget may not allow the generation of an optimal solution. Therefore, we present a method to determine the optimal number of HTN to consider with limited computational resources.

The contribution of our paper includes:

1. A framework for automatically assigning and scheduling tasks for humans and robots to complete multiple product assemblies
2. Methods to generate recovery task plans that seek human help automatically.
3. A management strategy to help practitioners determine near-optimal solutions under computation constraints.

We use an all-terrain vehicle assembly case study to demonstrate the capabilities of our proposed task allocation framework.

2 RELATED WORK

Human-Robot teams in High-mix Assembly: Traditionally, robotic cells deployed for assembly and manufacturing operations were confined within enclosures designed to exclude humans from the robot's workspace [1, 2, 3]. These cells were tailored for producing a single product in high volumes, rendering them ill-suited for settings with high-mix and low-volume production [4]. However, the growing demand for customization has prompted a shift towards deploying robots alongside humans to address the challenges of the variability in parts associated with high-mix assemblies.

Human-robot collaboration has gained significant attention in manufacturing over the past decade, mainly due to the advancements in collaborative robot technology [5, 6]. Nevertheless, much of the previous research has focused on scenarios involving a single robot collaborating with a single human [7, 8]. Work done in [9, 10, 11] demonstrates the key technological elements required for a human-robot collaborative cell. In our prior work [12], we demonstrated the successful operation of a multi-robot cell with a single human operator, handling complex assembly tasks. Motivated by this work, we extend our previous framework to encompass multiple cells, each equipped with multiple robots collaborating seamlessly with several human operators. Notably, our focus shifts to cells manufacturing multiple types of products, departing from the conventional approach of assembling a single product.

Task Planning and Scheduling: Task assignment and scheduling for human-robot teams have been subjects of extensive research [13, 14, 15, 16, 17, 18]. A comprehensive overview of approaches to the multi-robot task planning problem is presented in [19]. Recent studies have increasingly embraced the hierarchical planning approach for task planning and scheduling, given its capability to capture sequential and parallel task relationships across all levels of abstraction [20]. Formulations based on Hierarchical Task Networks (HTNs) coupled with mixed-integer programming-based solvers [21] have gained traction, facilitated by recent advancements in computational resources [22, 23]. However, existing research has predominantly addressed more straightforward problems involving a single cell with multiple robots. In our work, we extend the HTN framework to accommodate multiple cells engaged in producing diverse parts. Additionally, we propose a methodology to manage the intricacies arising from the introduction of multiple products into the planning problem.

Contingency-aware Planning: During the online execution of plans in human-robot teams, failures can arise from uncertainties in various factors, such as sensors, modeling errors, and robot motion errors [24, 25, 26]. Effectively addressing contingencies becomes important for ensuring the successful execution of tasks. In the context of the proposed problem involving multi-robot and multi-human cells, the sources of failures can be complex, and managing them can pose computational challenges. Previous research has primarily focused on modeling contingencies and devising corrective plans to proactively address them [27, 28]. However, in complex scenarios, predicting and modeling contingencies in advance may prove infeasible. An alternative approach involves repairing plans in real-time as failures occur [29]. This approach has garnered significant interest due to its capacity to handle contingencies at low-level task execution and high-level planning. This is particularly relevant in multi-robot team scenarios, where the complexity of interactions and dependencies adds another layer of intricacy to the contingency management process. As such, the adaptability of this approach becomes crucial for ensuring the resilience and effectiveness of human-robot teams operating in a high-mix setting to recover from failures.

3 BACKGROUND AND TERMINOLOGY

Assembly cell refers to the factory area where agents perform the assembly. We assume that multiple stations in the assembly cell will be used to perform assembly operations. Assemblies-in-progress will move from station to station using a mobile plat-

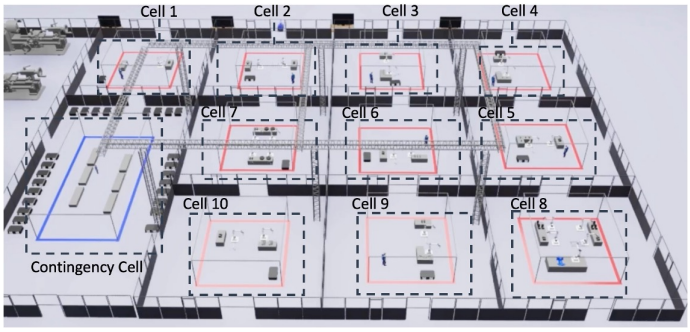

FIGURE 2: Top view of the ATV assembly cell.

form (called a mobile fixture). Fig. 1 shows an example of a mobile fixture used in industries. Several new factories are now utilizing this concept to build electric vehicles, satellites, and other applications. Humans and robots perform one or more assembly tasks in each station. There are three types of stations in the cell. The first type of station is the robot-only station. These stations will be used to perform routine assembly tasks. The second type of station will be a collaborative station. These stations will support collaboration among humans and robots. These stations will be used to perform challenging tasks, and humans will handle challenging aspects. For example, if a highly compliant part needs to be assembled, the human will perform the insertion task, and the robot will perform the fastening task. The third type of station will be a human-only station. This station will primarily perform rework and disassembly tasks that we need during unsuccessful assembly operations.

An HTN will describe the product assembly operations, specifying the tasks needed to realize the assembly and the precedence and mutex constraints among these tasks. Each task has a task type that determines what operation is required to complete it. Examples of task types include fastening, insertion, etc.

The cell consists of N stations, each capable of performing one or more task types. The objective is to finish assembling M products in the shortest possible time after the operation ends. Products being produced need not be identical. Each product is represented by its own HTN and, therefore, can include appropriate customization. An example of such a factory is shown in Fig. 2

The assembly cell has the following attributes:

1. Each station has one or more agents available to work on the task. For example, if two robots are needed to perform an assembly task, the station will have two robots. If a station serves a task type that requires human expertise, such as dexterous manipulation of soft materials or wires, then that station will have a human agent.
2. Each station can have the following states: (1) idle, (2) busy, and (3) unavailable. The idle state means that the station is available to perform a task. The busy state means that the station is currently executing a task. The unavailable state means that the station cannot be assigned a task. If a robot breaks down in a station, then the state of that station becomes unavailable.
3. The overall assembly cell has one more human agent. Humans play two roles. Either they work on their assigned station for regular operation or visit another station to resolve a contingency.
4. We assume that human agents available in the cell can handle every type of contingency that can occur in the cell.
5. Each station has input storage areas. Parts needed for tasks are delivered to the input storage area of the station using mobile robots. Once the assembly operation is completed in a station, the mobile fixture carrying the assembly-in-progress moves to the next station.
6. If a difficulty is encountered during the task execution and the station cannot complete the task, it invokes the contingency management system. Contingencies can either be resolved at the station or the assembly-in-progress moves to the specialized contingency handling station.
7. There is no fixed order for performing the assembly tasks. Tasks can be performed in any order consistent with the precedence constraints defined in HTN.

The assembly cell operates in the following manner:

1. A central cell planner assigns tasks to agents in the cell.
2. Parts are delivered to input storage areas of different stations in the cell by agents external to the cell.
3. Each agent with assigned tasks checks preconditions before starting its assigned tasks. These preconditions include the availability of parts in the input storage area and the completion of other tasks that serve as precursors to this task. If preconditions for task execution at a station are met, then the task execution commences immediately.
4. The cell uses sensors to monitor task execution. Task execution monitoring has three phases: (1) precondition checking, (2) monitoring of task-in-progress, and (3) post-completion verification. If a station successfully completes the task, the mobile fixture holding the assembly-in-progress moves to the next station.
5. If a contingency is detected, then an alert is issued, and the central planner updates the plan to handle the contingency.

Table 1 shows a few representative examples of a task assigned to a robotic station that fails and identifies a recovery action. When a station is unable to perform the task assigned to it, then a contingency event occurs. To deal with contingencies, the following modifications may need to be made to the task network. (1) Assign the contingency recovery tasks to possible agents that can execute the tasks. (2) Add an “undo” task to the network if a part that invoked contingency requires performing

TABLE 1: The table outlines various types of contingencies, details their respective impacts on HTN resource and task models, and proposes corresponding recovery actions.

Source of Failure	Impact on Resource and Task Network and Suggested Recovery Action
Cannot detect parts due to sensor failure	Change station state to unavailable and insert a task to repair the sensor.
Wrong or defective part is delivered to the input storage	Insert a task to move the defective part to the buffer area. The task can be attempted with a different part from the input again if input storage is not empty.
Parts on the input buffer are out of reach of the robot in the station	Insert a task to move the part on the input storage. This task type will be set to "difficult-move" and therefore it will need to be assigned to the human.
No part is available on input storage	Insert a task to fetch part. This task type will be performed by an external agent.
Grasping failure	If this is the first failure, then attempt the task one more time. If this is the second failure, then make the station unavailable and insert a repair task. The repair task will need to be performed by a human.
Execution failure with no damage to the assembly	Insert a "redo" task to attempt the task again. If the failure occurs a second time, then the task needs to be performed by a human.
Execution failure with damage to the assembly	Insert a repair task and make the station unavailable. The repair task will need to be performed by the human.
Unable to place the work-in-progress on output storage	Make the station unavailable and send an alert to the human.
Unexpected human presence in a station	Make the station unavailable and send an alert to the human.

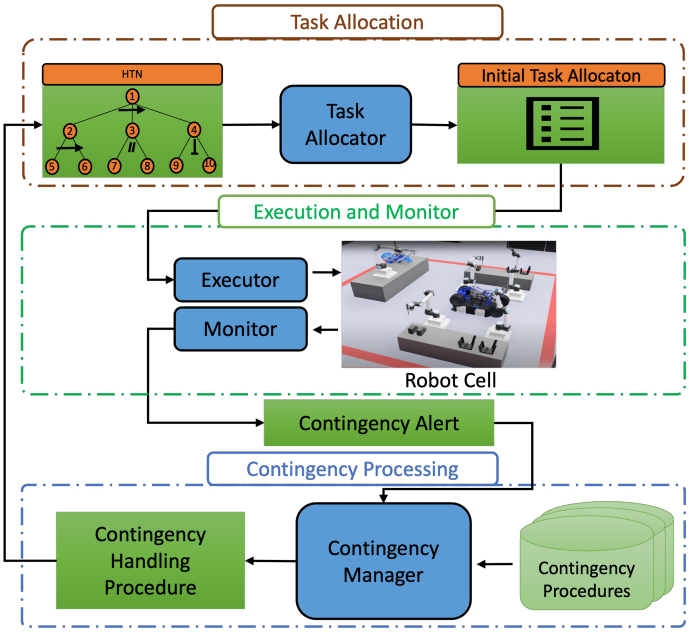

FIGURE 3: Software architecture of the robotic cell execution, task scheduling, and contingency management.

a task that reverses the effect of a previously completed task to gain access. (3) Add a "redo" task to the network that attempts the task again and counts the number of times the task has been attempted. (4) Add a "repair" task that requires repairing a resource in the cell. Consequently, this approach often necessitates substantial modifications to the HTN, which in turn can significantly escalate the complexity of the HTN.

4 PROBLEM FORMULATION

In our framework, multiple products will be assembled in the factory. Thus, managing the assembly process effectively requires a task scheduling system that accommodates multiple products, enabling the task schedule to accommodate as many tasks as possible. In this paper, we present an extension of the single-product HTN to a multi-product HTN, facilitating the simultaneous scheduling of tasks associated with multiple products to optimize task scheduling.

We employ an HTN framework that recognizes sequential, independent, and parallel task interdependencies, as elaborated in [30]. The task hierarchy originates with a root node and spans to sub-tasks that comprise the entire set of tasks. Atomic task nodes τ_i denote the fundamental tasks within the manufacturing workflow. Intermediate tasks, τ_{sub}, reside between atomic and root nodes, dictating the constraints for executing child tasks. The sequential node enforces an ordering constraint for child tasks in line with the sequential nature of assembly operations.

The independent node allows tasks to be executed in any sequence with no overlap, while the parallel node permits concurrent task execution by various agents. To transition from a single-product HTN to a multi-product HTN, we introduce a multi-product root node that amalgamates duplicates of the HTN for individual products. This integration employs a parallel constraint, ensuring agents can progress to subsequent products when unassigned. To prioritize tasks of the preceding product over subsequent assemblies, we implement task constraints among $\{HTN_1, HTN_2, HTN_3...\}$.

We have a resource model that monitors the status of components and agents within the system. Each task requires specific agents and components, the availability of which is crucial for task execution. Our model considers any changes in the availability of these resources or any operational disruptions that may occur during the process, treating such events as contingencies. For example, if there is a malfunction in the robot's control system or the robotic gripper, or if a part is not present at the station, the system flags these issues to the monitoring unit. This prompt detection allows immediate replanning to incorporate recovery tasks and adapt to the changed circumstances, ensuring minimal disruption to the production process.

Let H denote HTN that encapsulates the assembly constraints for tasks for assembling a single manufacturing product. Alongside, R represents its associated resource model, which captures the agents and components' real-time availability and status required to execute the tasks in H. Let $O(H,R)$ be a task planner that identifies a sequence S of task-to-agent allocations $A_j \leftrightarrow \tau_i$, associating each task τ_i within H with an agent A_i within R. The result is a schedule with makespan t. In situations where contingencies, F, demand system adjustments, such as modifications to H or the status of R, complications may ensue. These could stem from the availability of an agent, a faulty component, or an error in task execution. A contingency recovery method, α, is vital to reestablishing normal operations. The recovery task set τ_α, within α, is designed to recalibrate H in light of R and τ_i changes. When implemented, α modifies the original (H) to yield an updated (H'), a new resource model (R'), and a revised task set (τ, τ_α) as part of τ'. A new planner $O'(H,'R')$ then produces an updated task sequence with a new makespan t'.

Problem Statement: In environments where robots assemble multiple products, it is crucial to minimize the idle time of agents by implementing a scheduling system that optimizes for the shortest possible makespan within a manageable computational timeframe. However, this presents a challenge, especially when contingencies expand the number of nodes and decision variables in the modified H', thereby increasing computational complexity. Our approach aims to address the intricacies of computational management within the HTN framework and ascertain the ideal number of products to consider concurrently in the scheduling procedure. Furthermore, we propose strategies to manage instances where contingency recovery protocols significantly compound the complexity of the HTN.

Overview of Approach: To generate a task schedule from the set of HTNs at hand, we translate these HTNs into a constraint programming model, as detailed in section 5. Under normal operations, should any contingencies arise, we apply modifications to the HTN using the procedures outlined in section 6. When handling an HTN associated with multiple products, we may encounter a significant increase in complexity. To address this, we have developed a management strategy that allows us to derive near-optimal solutions with limited computational resources and time. This strategy is discussed in section 7. Section 8 presents a case study that shows how our approach can be applied to the All-Terrain Vehicle (ATV) assembly problem.

5 CONVERTING HTNs INTO A CONSTRAINT PROGRAMMING MODEL

Constraint programming is an exact method for solving scheduling problems. Recent advancements have shown that these methods can quickly find high-quality solutions for large-scale combinatorial optimization problems and further refine them. This section presents a constraint programming model derived from a flexible job shop problem formulation [31, 32, 33, 34, 35]. This model is used to find a solution for the decision variables that determine which agents should perform specific tasks and when these tasks should start while considering the HTN constraints.

First, we must encode the HTN into the CP Model. The HTN is a way of grouping tasks in a logical and natural manner that is easy for humans to understand. However, to encode the HTN into the constraint program, we must break down the HTN into lower-level precedence and independent mutex constraints for tasks. This is done by traversing the sub-task nodes in the HTN and generating a set of precedence and mutex task tuples for the atomic tasks encompassed by each sub-task.

We define the "children(.)" function, which returns the direct child nodes of a given sub-task node, and the "atomicChildren(.)" function, which returns the subset of leaf nodes (atomic tasks) that are a descendant of the node.

We illustrate the generation of lower-level precedence task pairs for sequential nodes in the following way: Let's assume we are generating lower-level constraints for node τ_{sub}, which is a sequential node. We first identify its direct children; for example $\{\tau_A, \tau_B, \tau_C\} \leftarrow \text{children}(\tau_{sub})$. These direct children nodes may also be sub-task nodes that group atomic tasks. We subsequently obtain the set of atomic tasks grouped under each child node, $A,B,C \leftarrow \text{atomicChildren}(\tau_A, \tau_B, \tau_C)$. In this example, the sequential task constraint would then specify that the set of atomic tasks $A = \{\tau_1, ...\tau_i\}$ must precede the set of atomic tasks $B = \{\tau_j ...\tau_l\}$. The set of task precedence pairs would be the following union of cartesian products $\{(A \times B) \cup (B \times C)\}$, which would then augment the existing set of task pairs P.

Likewise, if τ_{sub} represents an independent node, the mutex

task constraint would be that the set of atomic tasks for A, B, C cannot overlap. The set of task mutex tuples would be the cartesian product of atomic task sets $\{A \times B \times C\}$, augmenting the existing set of task mutex tuples. We repeat this process for every sequential and independent sub-task node in the HTN. Parallel nodes indicate no constraints for the child nodes.

We now present the CP model, the parameters, decision variables, and functions we use to create a schedule. It is worth noting that there is no standard way of presenting CP models. Therefore, we have adopted a notation similar to prior work with function names and decision variable types matching the naming conventions used in our OR-Tools CP-SAT solver implementation. It is important to mention that other solvers may use the same functions and decision variables but with different naming conventions. Thus, our formulation is a general constraint programming model.

Sets:

T: Set of tasks
N: Set of agents
T_k: Set of tasks that can be completed by agent k
N_i: Set of agents that can process task i
P: Set of task precedence pairs (i, j) specifying τ_i must precede τ_j
M: Set of task mutex tuples $(i, j, l, ...)$ specifying $\tau_i, \tau_j, \tau_l, ...$ cannot overlap with each other.

Parameters:

$p_{i,k}$: Nominal processing time of task i executed by agent k

Decision and Auxiliary Variables

$x_{i,k}$: Boolean variable, true if task i is processed by agent k, otherwise false
s_i: Integer start time of task i
p_i: Integer duration time of task i
e_i: Integer end time of task i
s_{ik}: Integer start time of task i being completed by agent k
e_{ik}: Integer end time of task i being completed by agent k
$o_i(s_i, p_i, e_i)$: Specifies interval variable for task i, which is a constraint that enforces $s_i + p_i = e_i$ and a sequencing variable used in other scheduling constraints such as $NoOverlap(.)$
$o_{ik}(s_{ik}, p_{ik}, e_{ik}, x_{ik})$: Specifies an optional interval variable for task i executed by agent k that enforces $s_{ik} + p_{ik} = e_{ik}$. If $x_{i,k}$ is true, then the interval constraint is enforced; if it is false, then the interval constraint is not enforced, and scheduling constraints will ignore the interval variable

Functions

NoOverlap$(\{o_i, o_j, ...\})$: The no overlap constraint ensures that the inputted interval variables do not overlap in time.

$$\text{Minimize} \quad t \tag{1}$$

subject to

$$s_j \geq e_i, \qquad \forall (i,j) \in P \tag{2}$$

$$\text{NoOverlap}(\{o_i, o_j, o_l\}), \qquad \forall (i,j,l) \in M \tag{3}$$

$$t = \text{Max}(\{e_i | i \in T\}) \tag{4}$$

$$\text{NoOverlap}(\{o_{ik} | i \in T_k\}), \qquad \forall k \in N \tag{5}$$

$$\sum_{k \in N_i} x_{i,k} = 1, \qquad \forall i \in T \tag{6}$$

$$s_i = s_{ik},\ p_i = p_{ik},\ e_i = e_{ik} \qquad \forall i \in T, \forall k \in N, \text{ if } x_{i,k} = 1 \tag{7}$$

The objective of this constraint program is to determine task assignments and task start times that minimize the time it takes to complete all tasks, which is the makespan represented by variable t. For all pairs of tasks with a sequential ordering constraint, Equation 2 states that task j must start after task i. For all pairs of tasks with an independent constraint, Equation 3 specifies that the intervals of task i and task j must not overlap. Equation 4 defines that the makespan variable t will be equal to the end time of the last task that is completed. Equation 5 requires that the intervals of tasks assigned to an agent must not overlap. This ensures each agent can execute at most one task for any given instance. Equation 6 specifies that a task can only be assigned to one agent. Lastly, Equation 7 states that if task i is assigned to agent k, then the start time, duration, and end time of the task will be equal to the corresponding start time, duration, and end time of the task-agent pair.

6 HANDLING CONTINGENCIES

Fig. 4 illustrates incorporating the recovery process within the revised HTN structure. Before the contingency plan α is incorporated, the contingency manager splits the parent node of the task affected by the contingency, referred to as τ_F, into two separate functions: (1) the contingency operation, which includes the node of the failed task and its associated recovery tasks, and (2) the regular operation, which contains the rest of the tasks in τ_{sub}. Subsequently, the failed task and its relevant α tasks are linked to the new τ_{sub}. This enables the scheduler to integrate the recovery and affected tasks into the scheduling process.

If a defect or incorrect assembly is detected in the engine after it has been mounted onto the chassis, it necessitates a disassembly for replacement or reorientation. This defect/misplacement could be detected at any point, such as during the placement of the lower cover on the chassis, as illustrated in Fig. 5. To facilitate the engine's re-assembly, the procedure of fastening the engine and any tasks that sequentially follow the engine assembly, like the lower cover assemblies, must be undone. The contingency manager examines the

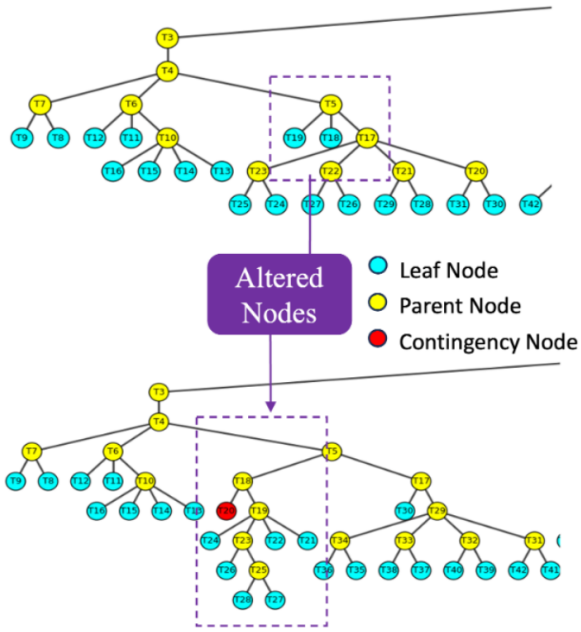

FIGURE 4: In contingency, recovery procedures get added to HTN to form new HTN with increased nodes and decision variables.

completed tasks to determine those sequentially linked within the HTN. This process entails inverting the sequence of completed operations to delineate the required disassembly steps. For instance, if the engine is affixed to the main chassis using screws, the task "fasten_engine_to_chassis" would be altered to "undo_fasten_engine_to_chassis." Such a transformation allows the production cell to backtrack on completed tasks, making the engine accessible for remedial actions. These identified disassembly steps are then integrated into the contingency plan α. The introduction of disassembly and the necessity to redo certain tasks results in an increased number of nodes within the HTN's structure, reflecting the additional complexity brought about by including reverse operations.

Addressing just one contingency event can introduce 4 to 50 additional tasks. Our system demonstrates robustness in scenarios involving multiple contingencies, denoted as F_i. Our constraint solver incorporates the tasks required for recovery from these multiple contingencies as decision variables, enabling it to generate optimal outcomes. For example, if one contingency would cause a more significant delay in the overall timeline than another, our solver prioritizes resolving one before the other. However, dealing with several concurrent contingencies presents a significant challenge due to the surge in computational complexity, which greatly increases the difficulty of maintaining an efficient and responsive assembly process.

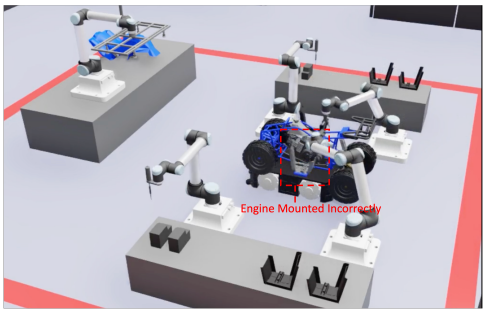

FIGURE 5: The situation shows the case of an engine mounted incorrectly, which was discovered while assembling the lower cover. The robot detects the contingency when it struggles to place a lower cover due to a tolerance issue caused by the incorrect placement of the engine.

FIGURE 6: Model of the ATV assembly used for our case study in Section 8.

7 MANAGING COMPUTATIONAL COMPLEXITY

The time needed to compute optimal task allocations and scheduling increases exponentially with the size of HTN. Instead, we want to find a high-quality solution given a fixed computation budget. Therefore, we cannot consider scheduling tasks over a very long time horizon. We need to determine how much computation budget to provide as a function of many products to consider in the time horizon when performing task scheduling.

We first study how using limited computing time affects the solution quality. We use three example HTNs of varying complexities, with differences in the number of nodes and decision variables. In Table 2, we present an analysis of the suboptimality factor, calculated as the ratio of a sub-optimal solution to a pseudo-optimal solution. We give the solver a large computation budget to obtain this pseudo-optimal solution, specifically over 7000 seconds. This extended runtime allows us to approximate

the optimal solution very closely, providing a benchmark against which the efficiency of quickly computed sub-optimal solutions can be measured. We present results for three different computation times. The table shows that with an increase in the number of products, there is a rise in the complexity of the HTN.

Our findings indicate that the number of decision variables is crucial in characterizing the HTN's complexity. This complexity is proportional to the increase of the makespan compared to the optimal solution when the available computation time is limited. This study suggests that decision variable management becomes a pivotal factor in mitigating the effects of and maintaining efficiency in multi-product assembly systems.

We investigate how considering different number of products in our look-ahead with limited computation time affects the sub-optimality factor. Figure 7 shows how different levels of product look-ahead affect the solution quality. When the number of products we look ahead is small, we can solve the problem optimally for the given level of product look-ahead. However, the solution does not consider all the possible tasks in complete production, and resources are not fully utilized. Therefore, sub-optimality arises from the resource utilization point of view. As shown in Fig. 7, an optimal level of look-ahead makes the best compromise between the complexity of the problem solved and inherent resource utilization inefficiency when using partial look-ahead.

To tackle the challenge of sub-optimality with constrained computation time, we put forward a method to visually analyze the influence of sub-optimality on the overall makespan for varying numbers of products. By referring to Fig. 7, we can find a strategy that employs an 'optimal look-ahead' in the number of products. This look-ahead strategy is designed to optimize the makespan within the bounds of the available solving time.

The concept of 'optimal look-ahead' involves scheduling over a specified number of future products to mitigate potential inefficiencies. This look-ahead approach allows us to strategically align tasks and resources for upcoming products. Doing so can reduce the reactive adjustments often required when scheduling without a look-ahead, which may lead to a greater makespan. We aim to select the most effective number of products to include in the scheduling horizon at any given time, aiming to maintain the makespan as close as possible to the optimal solution.

In Fig. 7, we present an analysis of three examples. This involves solving for eight products over an extended duration to reach a pseudo-optimal solution against solving for one to five products within a 120-second computation time constraint, thereby producing a sub-optimal solution. This visual representation enables us to analyze the varying degrees of effectiveness of different 'look-ahead' numbers for each HTN. We can also observe that for each specific HTN, a distinct number of products represent the best look-ahead count. This number is the point at which means the compromise between the sub-optimality caused by computation time constraint and sub-optimality caused by limited look-ahead.

The analysis described above is crucial for decision-makers to identify the optimal scheduling strategy that maximizes efficiency within the time constraints imposed on the scheduling process. We recommend users identify the computing time available for task assignment and scheduling. They should then conduct a parametric study to identify the product look ahead that they can use to minimize the sub-optimality factor. We used limited computational resources to solve the problem. Real-life assembly problems can be very complex; therefore, users should consider using sufficiently high-powered computers to ensure that the problem can be solved without incurring significant sub-optimality factor.

When contingencies occur, the look-ahead strategy in scheduling may need adjustment due to the resultant increase in decision variables. For example, HTNs shown in Table 2 with initial decision variable counts ranging from 200 to 1800, contingency handling procedures might introduce an additional 7 to 86 decision variables to handle and recover from the contingencies. Consequently, the recovery procedures can influence the number of decision variables. The available computational time during the contingency event may also be limited.

We conduct computational experiments to determine the optimal 'look-ahead' for HTNs in the presence of contingencies. This analysis is needed to decide whether to modify the number of products included in the scheduling process or if retaining the original 'look-ahead' count is feasible. This process is crucial in optimizing the scheduling strategy, especially in contingencies.

An example of this kind of analysis is shown in Fig. 8. This figure aids in visualizing how the optimal 'look-ahead' might shift in response to different contingency scenarios. In contingency scenarios, we allow a computation time of 120 seconds and a contingency handling procedure ranging from 20 to 60 nodes, 17 to 82 tasks, and 34 to 164 decision variables.

We recommend users analyze the impact of a contingency handling procedure on the decision variables. If the computation time is compressed during contingency handling, they should look at the optimal product look-ahead for the same computation time under a no-contingency situation. If a significant impact is expected on the decision variables because of the contingency, then the product look-ahead should be reduced by one.

8 CASE STUDY

This case study presents a micro-factory setup tailored for assembling ATVs shown in Fig 6. This factory is segmented into 11 stations, each with a designated function, including one exclusively assigned to manage contingencies. Agents are placed within these cells to attach components to the ATV chassis. The cells are equipped with robots for processes like insertion that do not require significant manual dexterity. However, for more intricate tasks, the cells are designed for human-robot collaboration.

TABLE 2: Comparative Impact of Product Count on Decision Variable Increase and Sub-optimality Under Time Constraints: 15s, 30s and 60s. "inf" refers to those that failed to produce the schedule within the allotted time.

	HTN 1				HTN 2				HTN 3			
		sub-optimality factor				sub-optimality factor				sub-optimality factor		
# of Products	# of Decision Variables	15s	30s	60s	# of Decision Variables	15s	30s	60s	# of Decision Variables	15s	30s	60s
1	343	1.17	1.06	1.05	297	1.31	1.10	1.02	234	1.08	1.08	1.08
2	686	1.40	1.24	1.17	594	1.70	1.14	1.15	468	1.20	1.20	1.20
3	1029	1.40	1.28	1.21	891	2.00	1.96	1.21	702	1.86	1.21	1.21
4	1372	1.40	1.38	1.38	1188	2.00	2.00	1.42	936	inf	1.41	1.22
5	1715	inf	1.42	1.42	1485	2.00	2.00	1.51	1170	inf	inf	1.51

Additionally, the factory utilizes moving platforms, which transport parts to the various assembly stations and function as assembly points. These platforms enhance the fluidity of the assembly sequence, as illustrated in Fig. 9. Upon arrival at the stations, robots are responsible for loading the parts onto the assembly stations.

The single product HTN for the assembly comprises 27 agents, 960 nodes, and 500 tasks, culminating in 768 decision variables. Using the method discussed in section 7, we determine that the optimal approach during regular operations is to plan with a look-ahead for four products. Under this nominal scheduling strategy, the makespan for completing ten products is calculated to be 1346 seconds. We use 60 seconds of computing time. The method we propose is illustrated through three examples of potential contingencies that could arise. For this analysis, we use 30 seconds of solving time to compute the task schedule:

Example 1: The robot detects that the engine was incorrectly assembled while assembling the lower cover, as shown in Fig. 5. Addressing this issue necessitates disassembling the vehicle down to the engine. This adjustment adds 44 new nodes to the task network, encompassing 24 re-do operations to correct the issue, six contingency operations to manage the process, and 13 un-do operations to dismantle the necessary components. Consequently, an additional 92 decision variables are introduced, and it becomes essential to reduce the look-ahead from four products to three to manage the increased complexity and maintain a feasible solution within the operational constraints.

Example 2: The required battery part is missing during the battery assembly phase, as depicted in Fig. 10, and the robot must wait for the part. The moving platform is then tasked with delivering the part to the station. Subsequently, the robot replenishes the inventory before continuing with the standard operations. This contingency introduces an additional 23 nodes, which include 14 operations related to the contingency—such as notifying the monitoring system, updating the cell status, and handling the parts upon arrival. Similar to the first example, this scenario also necessitates reducing the look-ahead from four to three products to accommodate the changes and sustain operational efficiency.

Example 3: During the screwing operation, a screw becomes jammed as the robot secures the engine, illustrated in Fig. 10. The procedure requires the cell to signal this contingency and pause operations pending the issuance of a revised schedule. Subsequently, the robot must reverse the screwing action to remove the stuck screw, detach and discard the compromised screw from its end-effector, retrieve and position a new screw, and attempt the screwing operation again. This contingency introduces 11 additional nodes, including six dedicated to handling the contingency. In contrast to the previous examples, this contingency does not necessitate reducing the look-ahead for product scheduling. The number of additional nodes and operations is relatively minor, and the system's existing capacity can absorb the impact without needing to adjust the look-ahead strategy.

9 CONCLUSIONS

This paper extends our prior work on handling contingencies in single-product single-station scenarios to multi-product multi-station scenarios. This requirement necessitated the development of a flexible factory layout capable of dynamically managing the assembly sequence to minimize idle time. We show that the management of computational complexity is needed to achieve near-optimal performance in this challenging problem. This method

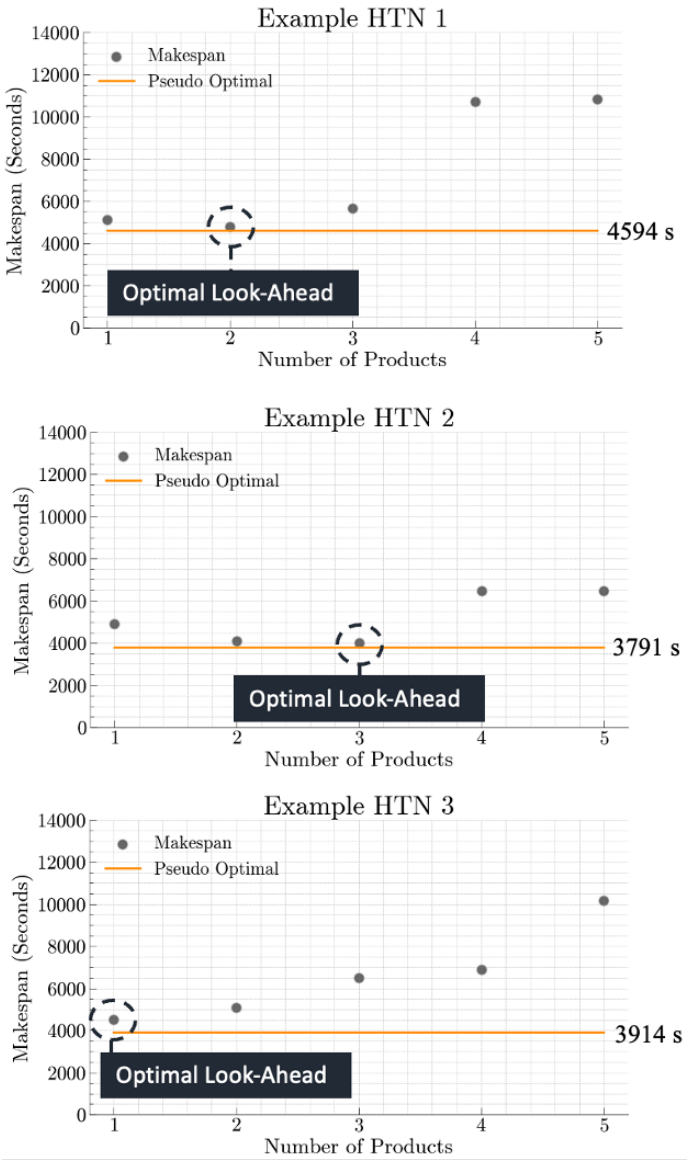

FIGURE 7: The graphs show that depending on the complexity of the HTN, the optimal look-ahead for product count in HTN differs. It is worth noting that the pseudo-optimal makespan could be lower if given a highly elongated compute time.

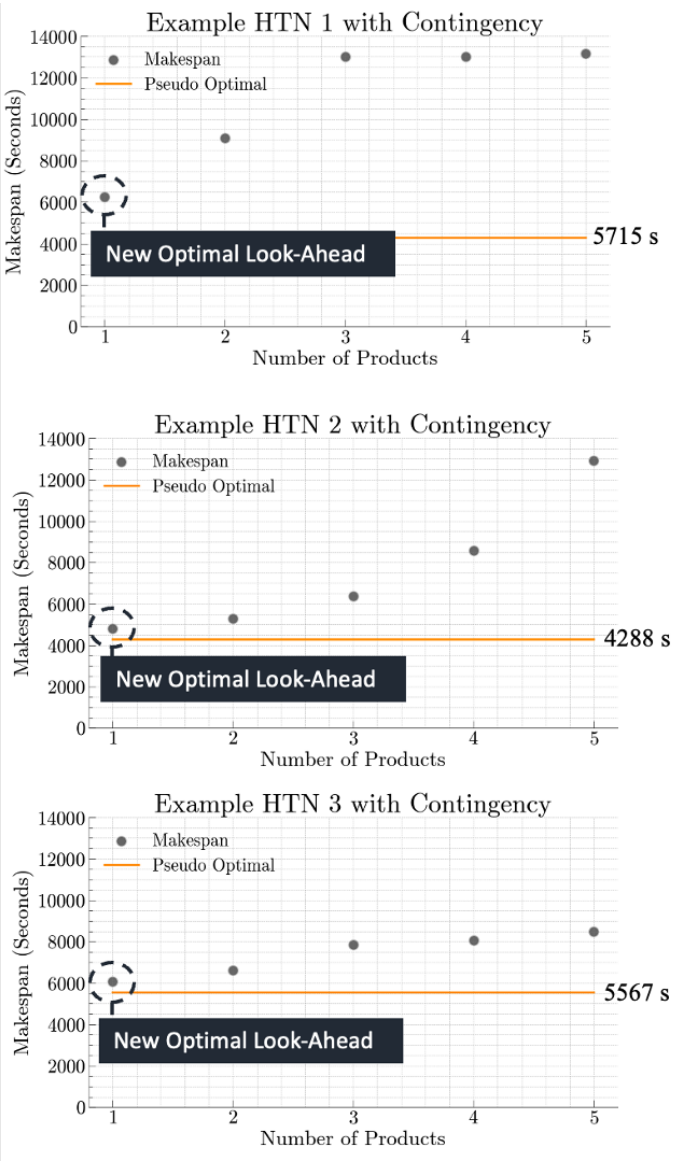

FIGURE 8: The graph shows the analysis of look-ahead including the contingency recovery procedures included in the computation.Example HTN 2 has the most reducrtion in the HTN count for computation in a contingency event, from 3 to 1, discounting two.

ensures that even with limited time and computational resources, the task schedules generated are as close to optimal even with limited time and computational resources. Furthermore, we integrate a contingency station within the cell that enhances operational resilience by enabling a flexible disassembly and rerouting process to revert to nominal operations during contingencies. This paper also introduces the ability to manage various contingencies. We have proposed and detailed a method incorporating contingency recovery tasks into the computational complexity management framework, ensuring that the task scheduling remains efficient even when encountering contingencies.

In our future work, we plan to incorporate travel costs associated with transitioning from one task to another so that the travel time of the agents can also be minimized when generating the schedule.

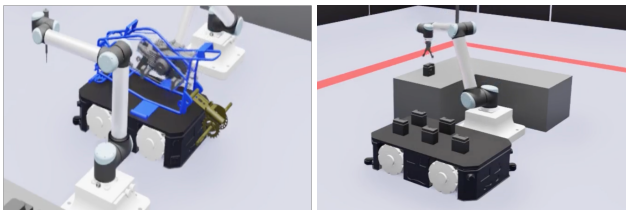

FIGURE 9: Autonomous driving vehicle is an assembly platform for agents to perform operations and bring parts to stations.

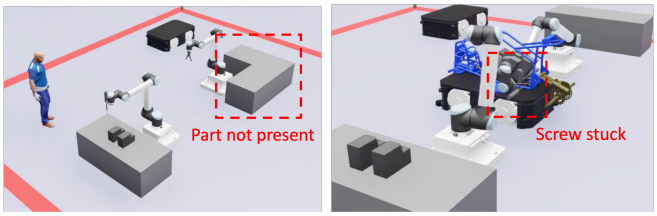

FIGURE 10: Example contingency 1: during the battery assembly operation, the part is not present, and the robot cannot proceed with the pick operation. Example contingency 2: the screw is stuck while the robot is fastening the engine.

Acknowledgement: This work is supported in part by National Science Foundation Grant #1925084. The opinions expressed are those of the authors and do not necessarily reflect the opinions of the sponsors.

REFERENCES

[1] Southern, W., and Lyons, C., 2002. "The study of a passive accommodation device in robotic insertion processes". *Journal of Materials Processing Technology,* ***124***(3), pp. 261–266.

[2] Hwang, M. J., Chung, S. Y., and Lee, D. Y., 2005. "Assembly approach for bimanual robots". In Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics., pp. 1482–1487.

[3] Grau, A., Indri, M., Bello, L. L., and Sauter, T., 2020. "Robots in industry: The past, present, and future of a growing collaboration with humans". *IEEE Industrial Electronics Magazine,* ***15***(1), pp. 50–61.

[4] Gan, Z. L., Musa, S. N., and Yap, H. J., 2023. "A review of the high-mix, low-volume manufacturing industry". *Applied Sciences,* ***13***(3).

[5] Tsarouchi, P., Makris, S., Michalos, G., Matthaiakis, A.-S., Chatzigeorgiou, X., Athanasatos, A., Stefos, M., Aivaliotis, P., and Chryssolouris, G., 2015. "Ros based coordination of human robot cooperative assembly tasks-an industrial case study". *Procedia CIRP,* ***37***, pp. 254–259. CIRPe 2015 - Understanding the life cycle implications of manufacturing.

[6] Matheson, E., Minto, R., Zampieri, E. G. G., Faccio, M., and Rosati, G., 2019. "Human–robot collaboration in manufacturing applications: A review". *Robotics,* ***8***(4).

[7] Wang, L., Gao, R., Váncza, J., Krüger, J., Wang, X., Makris, S., and Chryssolouris, G., 2019. "Symbiotic human-robot collaborative assembly". *CIRP Annals,* ***68***(2), pp. 701–726.

[8] Michalos, G., Makris, S., Spiliotopoulos, J., Misios, I., Tsarouchi, P., and Chryssolouris, G., 2014. "Robo-partner: Seamless human-robot cooperation for intelligent, flexible and safe operations in the assembly factories of the future". *Procedia CIRP,* ***23***, pp. 71–76. 5th CATS 2014 - CIRP Conference on Assembly Technologies and Systems.

[9] Yi, S., Liu, S., Xu, X., Wang, X. V., Yan, S., and Wang, L., 2022. "A vision-based human-robot collaborative system for digital twin". *Procedia CIRP,* ***107***, pp. 552–557. Leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022.

[10] Zhang, R., Lv, J., Li, J., Bao, J., Zheng, P., and Peng, T., 2022. "A graph-based reinforcement learning-enabled approach for adaptive human-robot collaborative assembly operations". *Journal of Manufacturing Systems,* ***63***, pp. 491–503.

[11] Hietanen, A., Pieters, R., Lanz, M., Latokartano, J., and Kämäräinen, J.-K., 2020. "Ar-based interaction for human-robot collaborative manufacturing". *Robotics and Computer-Integrated Manufacturing,* ***63***, p. 101891.

[12] Dhanaraj, N., Ganesh, N., Gurav, R., Jeon, M., Manyar, O. M., Narayan, S., Park, J., Yu, Z., and Gupta, S. K., 2023. "A human robot collaboration framework for assembly tasks in high mix manufacturing applications". In International Manufacturing Science and Engineering Conference, Vol. 87240, American Society of Mechanical Engineers, p. V002T07A011.

[13] Shriyam, S., and Gupta, S. K., 2018. "Task assignment and scheduling for mobile robot teams". In Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 5A: 42nd Mechanisms and Robotics Conference. Quebec City, Quebec, Canada. V05AT07A075.

[14] Dhanaraj, N., Narayan, S. V., Nikolaidis, S., and Gupta, S. K., 2023. "Contingency-aware task assignment and scheduling for human-robot teams". In 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 5765–5771.

[15] Manyar, O. M., McNulty, Z., Nikolaidis, S., and Gupta, S. K., 2023. "Inverse reinforcement learning framework for transferring task sequencing policies from humans to

robots in manufacturing applications". In 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 849–856.

[16] Gao, P., Siva, S., Micciche, A., and Zhang, H., 2023. "Collaborative scheduling with adaptation to failure for heterogeneous robot teams". In 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 1414–1420.

[17] Michalos, G., Spiliotopoulos, J., Makris, S., and Chryssolouris, G., 2018. "A method for planning human robot shared tasks". *CIRP Journal of Manufacturing Science and Technology,* ***22***, pp. 76–90.

[18] Dhanaraj, N., Malhan, R., Nemlekar, H., Nikolaidis, S., and Gupta, S. K., 2022. "Human-guided goal assignment to effectively manage workload for a smart robotic assistant". In 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pp. 1305–1312.

[19] Antonyshyn, L., Silveira, J., Givigi, S., and Marshall, J., 2023. "Multiple mobile robot task and motion planning: A survey". *ACM Computing Surveys,* ***55***(10), pp. 1–35.

[20] Cheng, Y., Sun, L., and Tomizuka, M., 2021. "Human-aware robot task planning based on a hierarchical task model". *IEEE Robotics and Automation Letters,* ***6***(2), pp. 1136–1143.

[21] Zhang, J., Liu, C., Li, X., Zhen, H.-L., Yuan, M., Li, Y., and Yan, J., 2023. "A survey for solving mixed integer programming via machine learning". *Neurocomputing,* ***519***, pp. 205–217.

[22] Dhanaraj, N., Narayan, S. V., Nikolaidis, S., and Gupta, S. K., 2023. "Contingency-aware task assignment and scheduling for human-robot teams". In 2023 IEEE International Conference on Robotics and Automation (ICRA).

[23] Yin, T., Zhang, Z., Zhang, Y., Wu, T., and Liang, W., 2022. "Mixed-integer programming model and hybrid driving algorithm for multi-product partial disassembly line balancing problem with multi-robot workstations". *Robotics and Computer-Integrated Manufacturing,* ***73***, p. 102251.

[24] Al-Hussaini, S., Gregory, J. M., and Gupta, S. K., 2023. "Generating task reallocation suggestions to handle contingencies in human-supervised multi-robot missions". *IEEE Transactions on Automation Science and Engineering*.

[25] Shriyam, S., and Gupta, S. K., 2018. "Incorporating potential contingency tasks in multi-robot mission planning". In 2018 IEEE International Conference on Robotics and Automation (ICRA).

[26] Choudhury, S., Gupta, J. K., Kochenderfer, M. J., Sadigh, D., and Bohg, J., 2022. "Dynamic multi-robot task allocation under uncertainty and temporal constraints". *Auton. Robots,* ***46***(1), jan, p. 231–247.

[27] Rhinehart, N., He, J., Packer, C., Wright, M. A., McAllister, R., Gonzalez, J. E., and Levine, S., 2021. "Contingencies from observations: Tractable contingency planning with learned behavior models". In 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 13663–13669.

[28] Schmerling, E., Leung, K., Vollprecht, W., and Pavone, M., 2018. "Multimodal probabilistic model-based planning for human-robot interaction". In 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE Press, p. 1–9.

[29] Coskun, A., and O'Kane, J. M., 2019. "Online plan repair in multi-robot coordination with disturbances". In 2019 International Conference on Robotics and Automation (ICRA), IEEE, pp. 3333–3339.

[30] Cheng, Y., Sun, L., and Tomizuka, M., 2021. "Human-aware robot task planning based on a hierarchical task model". *IEEE Robotics and Automation Letters,* ***6***(2), pp. 1136–1143.

[31] Müller, D., Müller, M. G., Kress, D., and Pesch, E., 2022. "An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning". *European Journal of Operational Research,* ***302***(3), pp. 874–891.

[32] Meng, L., Zhang, C., Ren, Y., Zhang, B., and Lv, C., 2020. "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem". *Computers & industrial engineering,* ***142***, p. 106347.

[33] Ham, A., Park, M.-J., and Kim, K. M., 2021. "Energy-aware flexible job shop scheduling using mixed integer programming and constraint programming". *Mathematical Problems in Engineering,* ***2021***, pp. 1–12.

[34] Teppan, E. C., 2022. "Types of flexible job shop scheduling: A constraint programming experiment.". In ICAART (3), pp. 516–523.

[35] Fatemi-Anaraki, S., Tavakkoli-Moghaddam, R., Foumani, M., and Vahedi-Nouri, B., 2023. "Scheduling of multi-robot job shop systems in dynamic environments: mixed-integer linear programming and constraint programming approaches". *Omega,* ***115***, p. 102770.