2022 31st IEEE International Conference on Robot and
Human Interactive Communication (RO-MAN)
August 29 - September 2, 2022. Naples, Italy.

2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN) | 978-1-7281-8859-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/RO-MAN53752.2022.9900551

Human-Guided Goal Assignment to Effectively Manage Workload for a Smart Robotic Assistant

Neel Dhanaraj[1], Rishi Malhan[1], Heramb Nemlekar[1], Stefanos Nikolaidis[1], and Satyandra K. Gupta[1]

Abstract— Managing robot workloads in human robot teams is critical for efficient team operation. If robots are overloaded with work, then they will miss deadlines and force humans to take on extra work. This paper presents a framework for a robot to assess its own workload based on an initial goal assignment. The robot does this by generating task and motion plans and computing the probability of missing deadlines due to the possibility of delays in task execution. A branch and bound based search is used to generate task and motion plans by minimizing task execution effort. The robot presents a diverse set of task and motion plans to the humans to offer multiple different options. Humans can either approve a plan or provide guidance to reduce the workload by either relaxing deadlines or removing goal(s) assigned to the robots.

I. Introduction

The advent of human-safe robots is creating new opportunities for humans and robots to collaborate in performing assembly, service, and maintenance operations, thereby increasing human productivity. Humans can focus on tasks that require a higher level of dexterity and human judgment to complete. For example, a human operator may need to assemble a fragile part to ensure that the part is not damaged during handling. A human may also need to perform an inspection to ensure that the two parts have formed a tight seal after an assembly step. On the other hand, robots can focus on supporting tasks such as fetching a tool/part or holding a part in place. This assistance can reduce the total time needed to complete the operation. Moreover, many supporting tasks may require extracting (placing) objects from (into) spaces that are hard-to-reach for human operators. Getting robots to do such ergonomically challenging tasks can also reduce the physical burden on human operators. Thus, humans must appropriately delegate goals to the robot for efficient human-robot teaming. The robot workload should be carefully managed. If the robot is overloaded with work, it will fail to meet deadlines and delay the overall work. If the robot is not given enough work, it will lead to poor resource utilization.

Our work considers tasks where robots serve as smart assistants to a human, and the human assigns a set of goals to the robots. For example, a human operator may ask a mobile manipulator to bring parts and tools at specific times during the task to facilitate the assembly or service operations that the user will perform. Figure 1 shows an example of the factory floor and challenges arising in task assignment and motion planning. Because of the variability in the human task execution times, the robot would be given a time window within which it must complete each goal, i.e., deliver each part. In this work, we particularly focus on problems where the robot may be unable to complete some of the goals in their given time windows due to delays caused by stochastic events such as congestion [1].

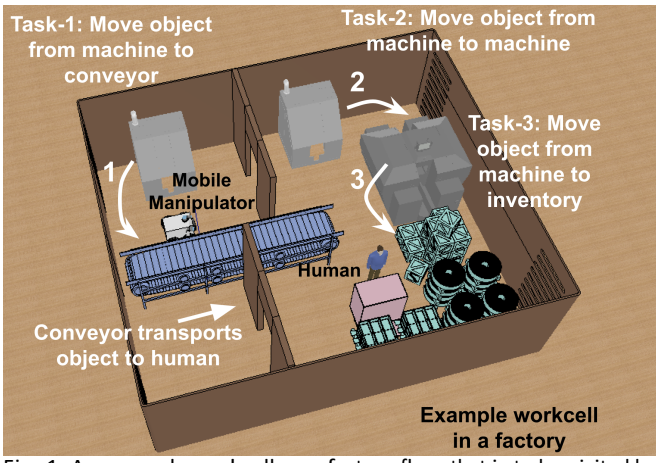

Fig. 1: An example workcell on a factory floor that is to be visited by the robot is illustrated. The mobile manipulator has three tasks assigned by the human. Human prefers the task-1 to be completed within a shorter timeline since the object is needed for assembly. Task-3 is difficult as moving from machine to inventory space has a congested workspace making motion planning challenging and task success uncertain.

We want to enable robots to evaluate the feasibility of completing the initial set of goals given by a human and generate plans to complete as many goals as possible. More importantly, we want to allow robots to reason over the goals and with human-guidance determine which can be skipped or changed to complete the remaining goals successfully. Our key insight is that humans have a preference for which goals the robot can or cannot skip or change, e.g., humans may prefer that the robot skips one or more simple goals to ensure the completion of a critical goal. Thus, we propose to account for human preferences by (i) relaxing the deadline for specific goals and (ii) removing certain goals from the robot's assignment.

Given an initial set of goals, we first use an integrated task and motion planning framework to covert goals into actions and actions into motion plans. We then evaluate the motion plans to determine if the robot can complete the given set of goals by the given deadlines. During this evaluation, we use a model of the environment to obtain probabilistic estimates of the travel times and determine the probability of completing each goal within the given deadline. This environment model will capture uncertainties like workspace congestion which

[1]Viterbi School of Engineering, University of Southern California, CA USA {dhanaraj, rmalhan, nemlekar, nikolaid, guptask}@usc.edu

can cause delays. The robot generates multiple plans that minimize human (and robot) effort while considering the importance of goals and present them to the human for refinement.

A key challenge is to shortlist a small set of good but diverse task plans to present to the human user to avoid overloading the user with too many or too similar options. We achieve this by recording best-seen plans, scoring based on the difference in the task sequence, and presenting the good plans with high diversity. The user can select their preferred task plan and refine the goal assignment if none of the plans are acceptable. For the latter, we allow the users to remove a goal from the robot's assignment or relax the deadline for an assigned goal. Each time the goal assignment is changed, the robot will re-plan and generate a new set of task plans to present to the user.

II. Related Works

Extensive work exists for task scheduling of multi-agent teams under temporal and spatial constraints [2], [3]. Approaches using hybrid mixed-integer linear programming, constraint programming methods, and auction-based methods have demonstrated task scheduling with upper and lower bound time constraints. Advancements in heuristic schedulers like apprenticeship learning [4], [5] have further increased scalability and efficiency.

Prior work exists in the multi-agent scheduling space that also considers the human agent's preferences and decision authority for the task scheduling process [6]. Santana et al. has presented a risk-sensitive scheduling algorithm that incorporates temporal uncertainty models for different activities when modeling uncertainties in the scheduling problem [7]. These works do not account for generating optimal task and motion plans when finding the best task schedules. For realizing capable robotic assistants, task and motion planning (TAMP) becomes essential and challenging to solve [8].

A detailed survey of TAMP problems and methods of solving them is provided in [9]. New works have leveraged symbolic planning, heuristics and learning to advance algorithmic approaches like in [10]. Works described in [11]–[13] discuss algorithmic methods for solving task and motion planning problems under uncertainty. Mansouri et al. addresses multi-robot planning under uncertain travel times using generalized stochastic petri nets [14], which is later interpreted as an MDP to generate optimal policies.

Our work focuses on managing the workload of a smart robot assistant, where the human takes on tasks that the robot may not be able to complete. This problem has similar elements with multi-agent scheduling, and multi-agent TAMP works. However, previous work does not treat hybrid scheduling and TAMP style problems where the goal is to minimize human effort from intervention, and how human input can help better manage robot workload.

The robot must present diverse options of plans so that humans are informed and can choose and refine the best option. Different works have investigated the generation of diverse solutions [15]–[19]. These methods use distance/diversity metrics to give insight into the plan set's diversity . Sohrabi et al. efficiently finds best-k solutions and then uses clustering to get diverse plans as cluster representatives [20]. Work of Nguyen et. al further considers user preferences to find a set of plans [21]. Our work builds on past work by finding best-k schedules and defining a diversity score (distance metric) to find the best diverse plans.

III. Problem Formulation

A. Background

A mobile manipulator is given a set of tasks to complete on a factory warehouse. The robot traverses a factory floor which consists of a large 2-D environment with n locations that the robot can occupy. Items are located in different work cells of the factory floor. An inventory of m objects describes the available items and the corresponding locations and items that can be picked up and transported by a robot with a payload p. The robot is a holonomic mobile manipulator that can move at a max velocity of v m/s and a max payload (p_{max}) of k items. We assume that different factors in the environment can delay the robot as it traverses the factory floor. We model these as probabilistic delays at different locations in a stochastic environment. We define the function travelTime(locA,locB) that simulates the robot executing a path plan once withing the stochastic environment and returns the time taken to travel. Figure 2 shows an example of a 2-D factory floor, its visitable locations, the areas for delays and the item inventory.

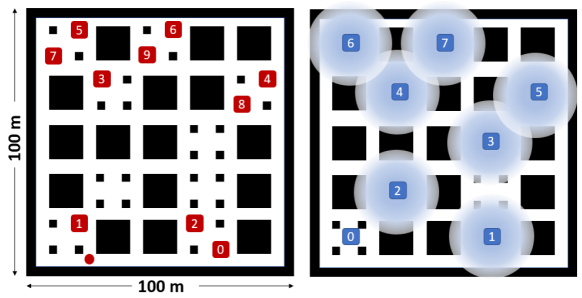

Fig. 2: 2D factory floor with height and width 100 x 100 m. (Left) Item inventory (item index shown at current location). (Right) Visitable locations (location index shown) and distribution of where congestion can occur.

We will now formalize the planning domain and provide definitions that are used in this work.

Definition 1. Workspace State Space S: The state of the environment is formally defined by an array of symbolic representations for all the items, robot, robot payload and current time step of the world. We define these symbols below.

- atLoc(item, location, $t_{arrival}$): Symbolic representation for item index, it's location in the factory floor, and time of arrival.
- robotLoc(robot, location): Symbolic representation for current robot location. The first element is the specific robot, and the second element is the location in the factory floor.
- p = [item1,item2,..itemn]: Representation for the set of items present in the robot payload.

- t_{world}: Defines the current time step of the world.

An example starting world state with two items and a robot with no items in its payload will be the following:

s_0 = [atLoc(1, 2, 0.0), atLoc(2, 5 , 2.0), robotLoc(1, 3), p = [], t_{world} = 5.0].

Definition 2. Tasks T: The legal transitions between s and s' is represented by a set of robot tasks with associated preconditions (pre) and effects (eff). In the environment, the robot can move to a location, pick an item and place it. Lastly the robot can wait for some amount of time.

1) moveRobot(robot1, loc1, loc2, travelTime(loc1, loc2))
   - pre: robotLoc(robot1, loc1), t_{world} = t
   - eff: robotLoc(robot1, loc2), t_{world} = t + travelTime(loc1,loc2)
2) pick(robot1, item1, loc1)
   - pre: atLoc(item1, loc1, $t_{arrival}$), robotLoc(robot1, loc1), |p| < p_{max}
   - eff: atLoc(item1, robot1, t_{world}), item1 ∈ p
3) place(robot1, item1, loc2)
   - pre: atLoc(item1, robot1, $t_{arrival}$), robotLoc(robot1, loc2), item1 ∈ p
   - eff: atLoc(item1, loc2, t_{world}), item1 ∉ p
4) wait(robot1, t_{wait})
   - pre: t_{world} = t
   - eff: t_{world} = t + t_{wait}

Definition 3. Goal g: A goal g_i to be completed by a robot is defined by the state of an item i i.e. atLoc(item, location, $t_{arrival}$) that satisfy the constraint $t_{min} \leq t^i_{arrival} \leq t_{max}$. This goal describes the item, the location it should be delivered, and the time windows that it should be delivered by.

Definition 4. Goal Importance $I(g_i)$: Different goals will have a different value of importance to the human. We define a function that takes in a goal input and returns a scalar value of importance. This value is defined by the human.

Definition 5. Effort to Complete Goal $e(g_i)$: Each goal will require some value of effort to complete it. The effort function takes in a goal input and returns a scalar value of the effort required to complete that specific goal by the robot (e^r) or a human (e^h).

Definition 6. Plan (Γ^a, Γ^s): A solution candidate for this problem will be a plan that takes the form of a sequence of goals attempted by the robot, $\Gamma^a = \{g_i, ..g_n\}$ and a set of goals skipped by the robot, $\Gamma^s = [g_j, ..g_m]$.

A fetching goal has an associated time window that the item must arrive within. A successful completion of a goal is affected by the uncertainty present in environment. Because we have modeled an environment with stochastic occurring delays, the robot runs multiple simulations of the corresponding motion plan. It gets a distribution of goal completion times and this determines the probability of goal completion within the deadline, as follows.

Definition 7. Probability of Goal Completion $P(g_i)$: The probability that goal g_i is completed by the robot within the arrival time deadline constraint.

By conducting multiple simulations of the robot complet-ing a proposed goal sequence Γ^a, the robot then gets a vector $[P(g_1), P(g_2), ..P(g_n)]$ that details the probability a goal is completed within a deadline. For any goals skipped by the robot in Γ^s, we set P(G) = 0.

Definition 8. Set of Constraints C: A plan can be constrained in the following ways. 1) As previously mentioned, the human can impose a time window deadline that the goal must be completed by. The solution may further be constrained by demanding 2) that the robot must attempt a particular goal or 3) the robot must skip a goal. 4) Lastly, the solution can be constrained by defining that the probability of success of an attempted goal must be above a certain threshold δ. These four constraints are defined below.

1) $t^i_{min} \leq t^i_{arrival} \leq t^i_{max}$
2) $g_i \in \Gamma^a$
3) $g_j \in \Gamma^s$
4) $P(g_i) \geq \delta_i$

B. Problem Statement

Given a set of goals $G = [g_1, g_2, ..g_m]$, a set of constraints C and an initial workspace state s_0, we want to determine a plan for the robot to execute that minimizes the effort expended by the human, and uses the robot efficiently. If $P(g_i)$ is the probability that the robot will accomplish the goal, then we define that the probability the human will have to accomplish the goal is $(1 - P(g_i))$. The effort that the human will have to exert for a proposed goal sequence is then defined as $\sum_g e^h(g)(1 - P(g))$. We, therefore, want to generate the best plan that minimizes the effort expended by the human and the effort expended by the robot. The cost function is the sum of the weighted human effort and the weighted robot effort for a certain plan. The goal importance is accounted for by multiplying the scalar returned from $I(g_i)$ by both the human and robot effort terms in the cost function so as to account for the importance of the goal. Lastly, we tune the A and B parameters so that the human effort is prioritized. The cost function and objective is formulated below.

$$\text{Cost} = A \sum_g I(g) e^h(g)(1 - P(g)) + B \sum_g I(g) e^r(g) \quad (1)$$

$$\text{minimize(Cost), subject to } c \in C \quad (2)$$

Overview of Approach. This paper presents a framework where the robot first generates the best plans that minimizes human effort, then select diverse plans to present and lastly enable the human to refine a plan by updating planning constraints. Figure 3 shows the system framework.

To generate the plans, the robot uses an integrated task and motion planning framework to find the best goal sequences and associated task and motion plans. The time delays while traveling to locations are incorporated during the search process. A branch and bound search is used in our work due to the anytime behavior of the algorithm. The task and motion plans generated in the search are simulated using a congestion model to find delays. The robot then evaluates the goal completion times against the deadline constraint to score each goal with the probability of it being completed. The method produces a good solution fast and then improves upon the solution by exploring other branches in the tree.

This is presented in Section IV.

The robot then records the best evaluated goal sequences, calculates a diversity factor for every good plan and populates a list of different diverse plans to be returned later to the human. This method is presented in Section V

Lastly a constraint sensitivity analysis table and a diverse set of plans are presented to the human from which they can select a suitable plan. If the human does not find any good plans, they can refine plans by updating planning constraints. We discuss this in Section VI.

IV. Plan Generation

Applying classical scheduling techniques is not feasible to generate optimal plans since the state space is large and there is uncertainty in motion plan execution. Furthermore, because the robot can carry more than one item, we must find a task plan that efficiently utilizes the payload space for a determined goal sequence. Our approach evaluates the probability of successfully executing good candidate plans by accounting for robot motion and workspace congestion. The formal description of the algorithm is given in Algorithm 1. A best-first branch and bound (BFBnB) search is used in our approach to find the goal sequences that meet our objective in Equation 1. The goal states we defined earlier are treated as nodes in the search algorithm. The robot will complete tasks (pick, place, or wait) to get to the next node. Two nodes in the tree are connected using an edge which represents the robot motion plan found by using A* algorithm.

For every new node that is explored, the path to get to that node is a partial goal sequence to be attempted by the robot. In our approach, we first use a standard search to find the best task plan, i.e [pick,pick,move,wait,place], to complete that partial goal sequence (Line 19). For a task plan, we use an A* path planner to find the path that takes the least amount of time from the robot's location to the goal location. The path planner generates a deterministic path without accounting for delays due to congestion. A velocity parameterization of the path using an environment model and a Gaussian delay model takes the congestion into account.

We model the environment as a coarse discrete grid in 2D where the robot can either travel in a delay-free zone or a congested zone. We simulate the travel through a delay free zone at max speed of the robot. When the robot traverses a grid cell in a congested zone, then the robot will have some average delayed velocity V_{avg}. We model this delay by setting the robot's delayed speed in the cell to be $V_{delayed}$ = delayFunction(). The delay function samples a Gaussian probability distribution, where $\mu = v_{max} - v_{avg}$ and $2\sigma = v_{maxdelay}$. From this formulation, we simulate a motion plan n times and get a distribution of robot arrival times for completing each action. We then score a candidate task plan by evaluating the item's probability of being delivered within the deadline constraint (Line 20).

The algorithm computes the cost of the sequence task and motion plans in order to score different goal sequences generated from the BFBnb. We use Equation 1 to calculate the cost of a scored sequence (Line 21). For the cost function, we make the effort expended to complete a goal as a function of the total distance d traveled to retrieve a part. The human effort function is defined to be $e^h(d) = w_1 d^2$. We choose a scaled quadratic function of distance traveled because we make the assumption that there is a diminishing decrease in perceived effort for humans traveling short distances compared to longer distances to complete a task. Lastly, the robot effort function is defined to be a scaled linear function of distance $e^r(d) = w_2 d$ so that plans with efficient robot actions are chosen. We choose a linear function because the distance traveled by the robot is proportional to its expended energy. The BFBnB algorithm searches through a combination of attempted and skipped nodes in order to find the plan (Γ^a, Γ^s).

For this problem, we choose a BFBnB search that quickly finds low cost solutions using heuristic knowledge first and then keeps on improving solutions. Given an initial popped partial goal sequence to branch on, we order the remaining goals to be completed by arrival deadline times and incrementally add the next earliest goal that can be completed (Line 17). This produces a new candidate partial solution which is scored from the associated task and motion plan (Line 20). If the success probability of that last attempted goal in the sequence is above the threshold constraint, then we calculate the worst-case cost of the partial sequence. The algorithm then stores the partial solution back in the queue and orders the queue by the worst-case cost (Line 33). The process then repeats, and the lowest cost partial solution is taken off the queue. We branch on the sequence to generate new partial solutions that contain a different possible next goal to complete.

If the success probability of the last goal in the partial sequence is below the threshold constraint defined by the human, then it is considered to have failed (Line 22). The algorithm then returns the goal sequence without the last goal as solution. For that returned sequence, if the robot still had unattempted goals that could have been completed within the deadline, then inherently a better solution exists where the robot's time is used more efficiently. Therefore the returned sequence is inefficient and pruned. If the last goal in the sequence failed because the robot ran out of time, then the cost of the complete solution is compared to the best cost found at the time.

If there are no more goals to complete, it is considered a complete solution and is compared against the best solution found at the time. We terminate the algorithm when there are no more partial solutions in the queue to branch upon. The best stored solution is the goal sequence Γ^a that will be attempted by the robot. Any assigned goals that are not in the best goal sequence are goals that are skipped by the robot Γ^s. The terminated algorithm returns a plan (Γ^a, Γ^s).

V. Diverse Plan Selection

Although the algorithm we present in Section IV will generate the optimal solution, there exist multiple sub-optimal plans that a human might consider useful for plan refinement. While the search algorithm explores different candidate solutions, the robot maintains a list of explored best solutions that

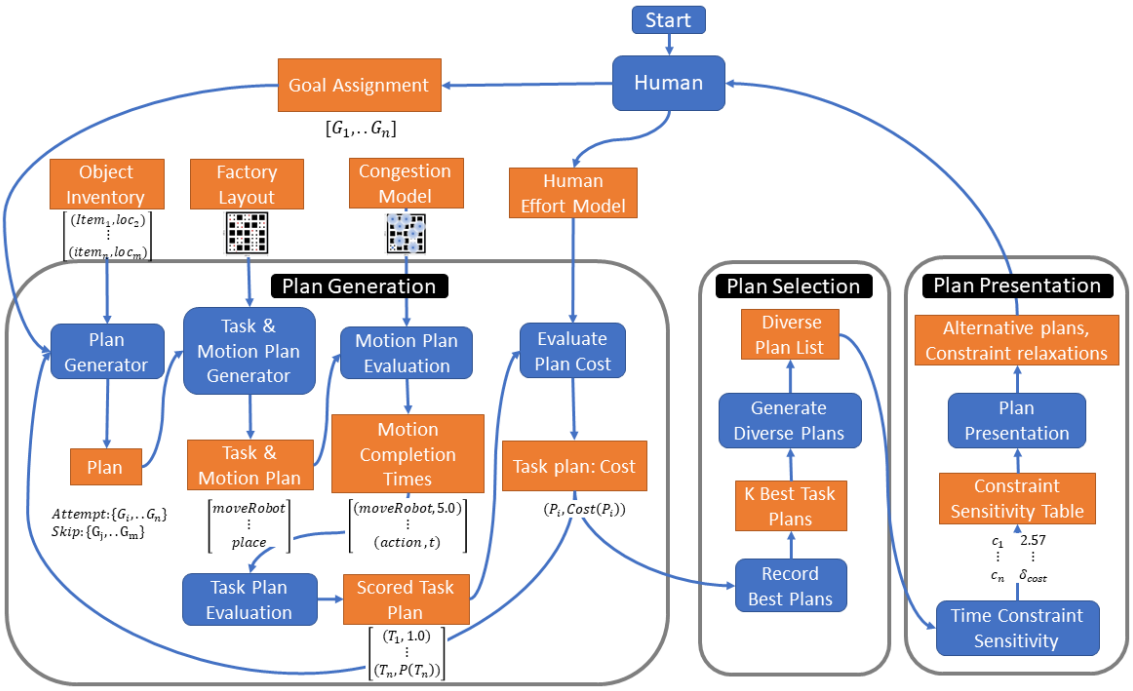

Fig. 3: Block diagram describing process of generating plans, selecting the best diverse plans and presenting them to the human.

have a cost within some percentage of the optimal solution cost. The user defines this percentage.

It would be inefficient to use a k best solutions approach and show every possible best solution to the human because it would result in cognitive overload. Instead, we want to generate a list of goal sequences that are different from each other. We define a diversity factor D that quantifies the difference or diversity of two plans (Γ^a_1, Γ^s_1) and (Γ^a_2, Γ^s_2). In this paper, we formulate this factor to be dependent on how different the order of two attempted goal sequences, Γ^a_1 and Γ^a_2 are with each other, as well as whether one plan contains an attempted goal that the other plan skips.

We quantify D_1, the diversity in the order that goals in two sequences are attempted by taking the difference of the location of each goal in the sequences. This method is formulated as the following, where the function $pos(goal, \Gamma^a)$ returns the index of a goal in a specific sequence.

$$D_1 = \sum_{g_k \in \Gamma^a_1 \cap \Gamma^a_2} |pos(g_k, \Gamma^a_1) - pos(g_k, \Gamma^a_2)| \quad (3)$$

This paper quantifies the difference in what goals are skipped for each plan by defining D_2 to be the number of goals skipped in one plan but not the other. First, the function $mem(goal, \Gamma^s)$ is defined to return one if the goal is a member of the set of skipped goals and otherwise returns zero if not. D_2 is defined as:

$$D_2 = \sum_{g_i \in \Gamma^s_1} (1 - mem(g_i, \Gamma^s_2)) + \sum_{g_j \in \Gamma^s_2} (1 - mem(g_j, \Gamma^s_1)) \quad (4)$$

We formulate the diversity between two plans as the weighted sum of D_1 and D_2. The weights are defined by the user based on their preference of whether goal order or skipped goals makes a plan more diverse.

$$D(P_1, P_2) = w_1 D_1 + w_2 D_2 \quad (5)$$

The robot initializes a list of diverse plans with just the optimal solution as its member. It then takes alternative solutions and calculate the diversity factor between each member of the diverse plan list. A plan's diversity scores gets assigned the lowest calculated diversity factor. If the diversity score is not within a defined threshold of the highest diversity score, it is not added to the list of diverse plans.

VI. Getting Human Guidance

Even though the robot generates diverse plans for a user to choose from, the human may still not find them acceptable. A user would therefore update planning constraints in order to search for a better solution. For example, goals may implicitly have more relaxed deadlines then initially specified. The human would decide to relax those deadline constraints to generate and select a lower cost plan. A user could also add more constraints by specifying goals they need the robot to attempt or skip.

In order for the robot to get guidance, the human must first be informed of what constraints are affecting workload management. In this work, we focus on conducting a constraint sensitivity analysis for the deadline constraints. For every explored solution, there already a distribution of goal completion times sampled from simulating a motion plan. We individually examine each assigned goal's deadline constraint $t^i_{min} \le t^i_{arrival} \ge t^i_{max}$, by relaxing t_{max} by an increment of time δt and recalculate the scored goal sequences and cost of the best-found plans. We find the percentage decrease of the lowest-cost plan for relaxed constraints compared with

Algorithm 1 Task Plan Generation

```
g_seq: Current Goal Sequence
C_seq: Current Goal Sequence Cost
g_bestseq: Current Goal Sequence
C_bestseq: Current Goal Sequence Cost
BFPartialSol: Best First Ordered List of Partial Solutions

function BRANCH-AND-BOUND-SOLVE()
    INITIALIZE
    g_bestseq ← {}
    C_bestseq ← ∞
    g_seq ← {}
    g_seq ← 0
    P ← EMPTY PLAN
    Insert P in BFPartialSols
    while BFPartialSol is not empty do
        P_partial ← MIN-COST-PLAN(BFPartialSol)
        GoalSequences ← GET-NEXT-PLANS(P_partial)
        for sequence in Goal Sequences do
            motionplan ← GET-BEST-ACTION-PLAN
            scoredgoals ← SIMULATE-MOTION-PLAN
            PlanCost ← GET-COST
            if last goal failed then
                if C_seq < C_bestseq then
                    g_bestseq = g_seq
                    C_bestseq = C_seq
                end if
            else if no goals left to attempt then
                if C_seq < C_bestseq then
                    g_bestseq = g_seq
                    C_bestseq = C_seq
                end if
            else
                Insert sequence in BFPartialSols
            end if
        end for
    end while
     return g_bestseq
end function
```

the original optimal plan cost: $\frac{|Cost_{relaxed} - Cost_{orginal}|}{Cost_{orginal}} \times 100\%$. A constraint sensitivity analysis table is then presented to inform the user of how changing a deadline constraint can decrease the solution cost.

Along with the table, the robot shows n diverse plans that were previously generated as described in Section V. If the human does not find any plans acceptable, they would give the robot inputs to help the robot find a suitable plan. The input is given by updating planning constraints and then finding new solutions that satisfy the constraints. We define four input types that the human may give and map these inputs to a constraint modification action as shown in Table I. The user can iteratively give inputs and have the robot find new plans until an acceptable plan is selected, which then terminates the search as shown in Figure 4.

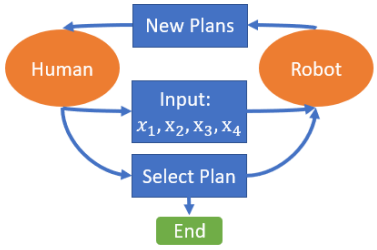

Fig. 4: Flowchart showing human input for plan refinement with robot.

Human Input and Associated Constraint Modification
"Increase goal's upper bound deadline time"
$x_1(g_i, t_2) \rightarrow (t_1 \leq g_i(t_{arrival}) \leq t_2) \in C$
"Robot must complete goal"
$x_2(g_i) \rightarrow (g_i \in \Gamma^a) \in C$
"Robot can skip goal"
$x_3(g_i) \rightarrow (g_i \in \Gamma^s) \in C$
"Increase/decrease goal minimum probability of success "
$x_4(, g_i, \delta_i) \rightarrow (P(g_i) \geq \delta_i) \in C$

TABLE I: This shows human inputs mapped to constraint updates for plan refinement.

VII. Results

A. Experimental Case Study Details

We define a 100m by 100m factory floor with 8 locations for this work. Figure 2 shows the map of this workspace. The robot is a 1m by 1m holonomic mobile manipulator that can move at a max velocity of 1.3 m/s and a max payload (p_{max}) of two items. We give the robot six goals to complete within 30 minutes. We first design three different goal assignment types. The planner finds the best goal sequence, task and motion plan, and alternative solutions for each goal assignment to present to the human. Lastly, we show examples of how a single user can give different human inputs to guide plan refinement. For two scenarios, a single user is first presented both the best and alternative goal sequences, and then they help the robot refine the plan by giving a constraint modification input through the command line.

B. Generating Goal Sequences For Different Problem Inputs

We first create three sets of goal assignments, (G_1, G_2, G_3). Each set has six goals with randomly generated items and delivery locations that must be completed with different deadline constraints. We choose large delivery time windows (loose deadlines), small delivery time windows (tight deadlines), and a mix of loose and tight deadlines as constraints for the respective goal assignments. These goal inputs and the delivery time windows over a 30-minute span is visualized in Figure 5 .

We execute our plan generation algorithm for each input to generate the optimal plan solution. These optimal plans are shown in Figure 6. We also show an example of a task plan generated for the third goal assignment in Figure 7. For a 30-minute span, the robot can accomplish about five tasks at best. The generated goal sequence has a higher proba-bility of success for the loose deadline goal assignment. In contrast, goals with tight deadlines have a lower probability

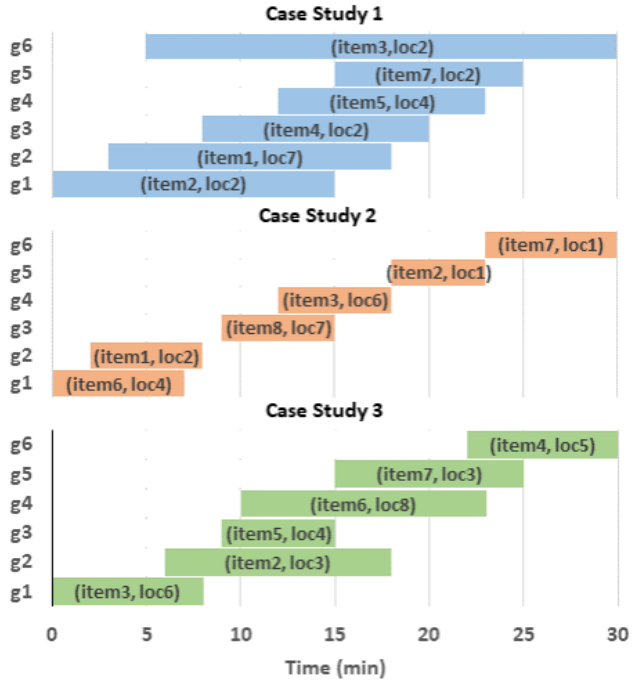

Fig. 5: Three different goal assignment types: loose, tight, and mixed deadlines.

of success. The plan generated for the mixed deadline goal assignment has a slightly less probability of success for each goal than the first goal assignment. A set of goals with tight deadlines results in poor plans being generated. However, both loose and mixed deadlines result in lower-cost plans being generated. We also generated best alternative plans using the diversity scoring metric.

$$
\begin{array}{lll}
\boldsymbol{Plan_1} & \boldsymbol{Plan_2} & \boldsymbol{Plan_3} \\
goal \quad P(g) & goal \quad P(g) & goal \quad P(g) \\
\Gamma_a = & \Gamma_a = & \Gamma_a = \\
\begin{bmatrix} g_2 & 1.0 \\ g_3 & .93 \\ g_4 & .88 \\ g_5 & .82 \\ g_6 & .7 \end{bmatrix} & \begin{bmatrix} g_1 & .71 \\ g_3 & .88 \\ g_4 & .81 \\ g_5 & .49 \\ g_6 & .17 \end{bmatrix} & \begin{bmatrix} g_1 & .94 \\ g_4 & .57 \\ g_2 & .89 \\ g_5 & .92 \\ g_6 & .7 \end{bmatrix} \\
\Gamma_s = & \Gamma_s = & \Gamma_s = \\
[g_1 \quad 0.0] & [g_2 \quad 0.0] & [g_3 \quad 0.0]
\end{array}
$$

Fig. 6: Optimal plans generated with associated cost

C. Getting Human Guidance for Refining a Plan

In order to facilitate human guidance for refining the plan for the third goal assignment, the robot generates a sensitivity analysis table for adjusting the deadline constraints. The analysis individually increases each constraint to see how it affects the best plan cost. We see that the plans for g_3 can be improved by relaxing time constraints for g_1, g_5, g_6.

We present two examples of inputs the human can give to manage the robot's workload. These examples are shown in Figure 8. The three plans calculated for g_3 are first presented to the human for both examples.

Example 1. For the first example, we show plan refinement for a human who is 1) concerned with reducing the probability of the robot missing a goal and 2) has a higher perceived effort for certain goals that were not captured by the human effort function. The human wants to refine the plan without compromising on the assigned deadlines. In example 1, let us say that the human's perceived effort for goal 3 is actually greater than what is modeled by the human effort function. The human decides to consider the lowest presented $plan^3_{D1}$. For that plan, they would have to complete goal 3 (a perceived high effort goal), and there is a higher probability that the robot will miss goals four and six. In this example, the human takes responsibility for goals 4 and 6 and directs the robot to attempt high-effort goal 3. This task reassignment is mapped as inputs $x_2(g_3)$ and $x_3(g_4, g_6)$. Furthermore, the human adds a threshold constraint so that the newly generated plan must have the robot complete goal 3 with $P(g_i) > .9$. This scenario shows that the robot can generate a new plan that satisfies the updated planning constraints provided by the human.

Task	g_1	g_2	g_3	g_4	g_5	g_6
Cost Reduction	15%	0%	8%	18%	10%	20%

TABLE II: Each task's upper deadline bound was extended by 5.0 min to see the effect on the cost of the best solution

Example 2. In the second example, we show plan refinement iteration with another human who has situational knowledge of goals with deadlines that can be further relaxed. With information from the constraint sensitivity analysis shown in Table II, the human gives an input to increase the upper deadline bound for goals four and six. This input decreases the cost of the first presented plan: $P_{D\,1}$, which the human selects. In this example, we show that the human can give deadline relaxations to improve the robot's workload management.

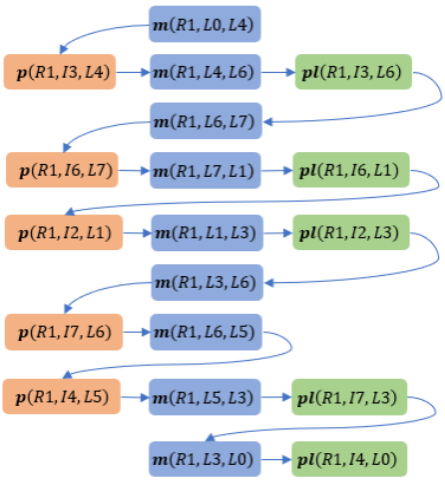

Fig. 7: Task and motion plan for goal sequence 3, where the robot can pick (p), move (m) or place (pl)

VIII. Conclusions

We have presented a framework for a human to manage a robot's workload based on their preference. We have demonstrated that a branch and bound-based search can be used to generate and evaluate alternative task sequences to minimize task execution effort. Our work has also illustrated how congestion modeling can determine the probabilities of

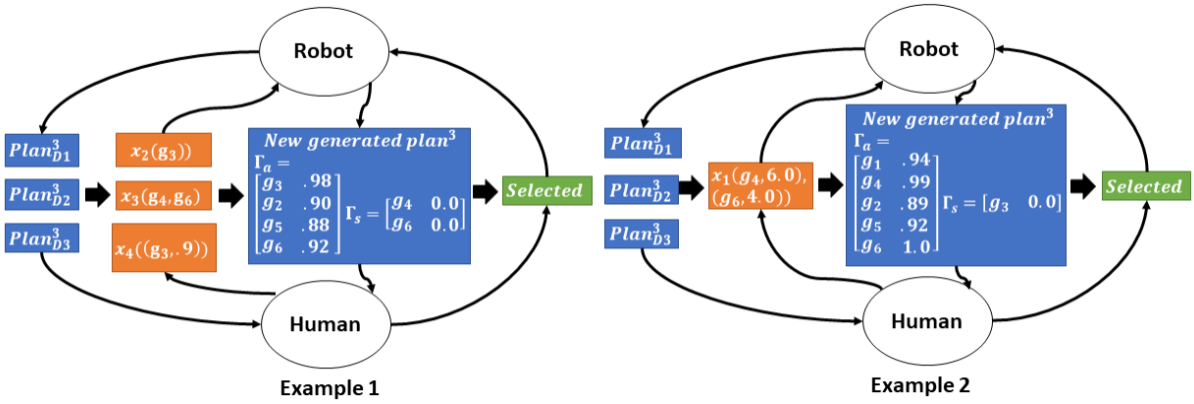

Fig. 8: Two examples of human guidance with workload management

completing a goal within a given time window. Furthermore, we have demonstrated how task execution effort sensitivity can be computed by relaxing deadlines and offering humans helpful information to make decisions. This framework enables humans to take actions based on their preferences and needs. They can either remove goals from the robot's list or increase their workload. Alternatively, they can relax deadlines for the robot to complete specific goals and accept a delay in task completion.

A key limitation to this work is the lack of a human user study with a real robot. This is needed to evaluate the computational tools presented in this paper and give insight into how human risk tolerance, preferences, and availability of information affect the robot workload management process. Furthermore, the robot has full observability of the environment and does not consider sensor coverage [22], [23], or human viewpoint [24].

Acknowledgements

This work was supported by the National Science Foundation NRI (# 2024936).

References

[1] S. Lim, H. Balakrishnan, D. Gifford, S. Madden, and D. Rus, "Stochas-tic motion planning and applications to traffic," The International Journal of Robotics Research, vol. 30, no. 6, pp. 699–712, 2011.

[2] C. Zhang and J. A. Shah, "Co-optimizating multi-agent placement with task assignment and scheduling." in IJCAI, 2016, pp. 3308–3314.

[3] M. Pearce, B. Mutlu, J. Shah, and R. Radwin, "Optimizing makespan and ergonomics in integrating collaborative robots into manufacturing processes," IEEE Transactions on Automation Science and Engineering, vol. 15, no. 4, pp. 1772–1784, Oct 2018.

[4] M. Gombolay, R. Jensen, J. Stigile, S.-H. Son, and J. Shah, "Apprenticeship scheduling: Learning to schedule from human experts." IJCAI, 2016.

[5] M. Gombolay, R. Jensen, J. Stigile, T. Golen, N. Shah, S.-H. Son, and J. Shah, "Human-machine collaborative optimization via apprenticeship scheduling," Journal of Artificial Intelligence Research, vol. 63, pp. 1–49, 2018.

[6] M. C. Gombolay, C. Huang, and J. Shah, "Coordination of human-robot teaming with human task preferences," in 2015 AAAI Fall Symposium Series, 2015.

[7] P. Santana, T. Vaquero, C. Toledo, A. Wang, C. Fang, and B. Williams, "Paris: A polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty," in Proceedings of the International Conference on Automated Planning and Scheduling, vol. 26, 2016, pp. 267–275.

[8] A. M. Kabir, S. Thakar, P. M. Bhatt, R. K. Malhan, P. Rajendran, B. C. Shah, and S. K. Gupta, "Incorporating motion planning feasibility considerations during task-agent assignment to perform complex tasks using mobile manipulators," in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 5663–5670.

[9] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," Annual review of control, robotics, and autonomous systems, vol. 4, pp. 265–293, 2021.

[10] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," The International Journal of Robotics Research, 2018.

[11] M. Mansouri, F. Pecora, and P. Schüller, "Combining task and motion planning: Challenges and guidelines," Frontiers in Robotics and AI, vol. 8, p. 133, 2021.

[12] N. Shah, D. Kala Vasudevan, K. Kumar, P. Kamojjhala, and S. Srivastava, "Anytime integrated task and motion policies for stochastic environments," in 2020 IEEE International Conference on Robotics and Automation (ICRA), May 2020, pp. 9285–9291.

[13] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," IJRR, 2013.

[14] M. Mansouri, B. Lacerda, N. Hawes, and F. Pecora, "Multi-robot planning under uncertain travel times and safety constraints," in The 28th International Joint Conference on Artificial Intelligence (IJCAI19), August 10-16, Macao, China, 2019, pp. 478–484.

[15] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in GECCO, 2011.

[16] A. Coman and H. Munoz-Avila, "Generating diverse plans using quantitative and qualitative plan distance metrics," in Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.

[17] M. Fontaine and S. Nikolaidis, "A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy," Robotics: Science and Systems, 2020.

[18] ——, "Differentiable quality diversity," Advances in Neural Information Processing Systems, vol. 34, 2021.

[19] M. C. Fontaine, Y.-C. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis, "On the importance of environments in human-robot coordination," Robotics: Science and Systems, 2021.

[20] S. Sohrabi, A. V. Riabov, O. Udrea, and O. Hassanzadeh, "Finding diverse high-quality plans for hypothesis generation," in ECAI 2016. IOS Press, 2016, pp. 1581–1582.

[21] T. A. Nguyen, M. Do, A. E. Gerevini, I. Serina, B. Srivastava, and S. Kambhampati, "Diversity article -generating diverse plans to handle unknown and partially known user preferences," Artificial Intelligence, vol. 190, pp. 1–31, 2012.

[22] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 5204–5210.

[23] S. Nikolaidis, R. Ueda, A. Hayashi, and T. Arai, "Optimal camera placement considering mobile robot trajectory," in 2008 IEEE International Conference on Robotics and Biomimetics. IEEE, 2009, pp. 1393–1396.

[24] S. Nikolaidis, A. Dragan, and S. Srinivasa, "Viewpoint-based legibility optimization," in 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI). IEEE, 2016, pp. 271–278.