Multi-Robot Scheduling and Path-Planning for Non-Overlapping Operator Attention

Sebastián A Zanlongo, Franklin Abodo, Philip Long, Taskin Padir, Leonardo Bobadilla

Multi-Robot Scheduling and Path-Planning for Non-Overlapping Operator Attention

Sebastián A. Zanlongo*, Franklin Abodo*, Philip Long†, Taskin Padir† and Leonardo Bobadilla*
*School of Computing and Information Sciences
Florida International University
Miami, FL 33199, USA
Email: {szanl001, fabod001}@fiu.edu, bobadilla@cs.fiu.edu
†Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115, USA
Email: {p.long, t.padir}@northeastern.edu

***Abstract*—There is a growing need for robots to perform complex tasks autonomously. However, there remain certain tasks that cannot - or should not - be completely automated. While these tasks may require one or several operators, we can oftentimes schedule when an operator should assist. We build on our previous work to present a methodology for allocating operator attention across multiple robots while attempting to minimize the execution time of the robots involved. In this paper, we: 1) Analyze of the complexity of this problem, 2) Provide a scalable methodology for designing robot policies so that few operators can oversee many robots, 3) Describe a methodology for designing both policies and robot trajectories to permit operators to assist many robots, and 4) Present simulation and hardware experiments demonstrating our methodologies.**

I. Introduction

The ability of robots to autonomously perform tasks has vastly increased in recent years. Despite this, there are many tasks that currently cannot - or will not - be automated, and therefore require a human operator to step in and assist the robot. Examples include industrial [30] and defense tasks such as traversing a dangerous environment, executing sensitive maneuvers that include complicated kinematics and actions that require human oversight - such as removing detonators. These portions of a robot's actions cannot be automated for practical, ethical, or other concerns. Moreover, many robot designs assume that there will be dedicated operator for each robot, and some robots may require separate operators for different actions (e.g. discrete operator for navigation, manipulation, monitoring, etc.). However, while these tasks might not be automated, they can be planned for. Indeed, we see great benefits in combining human abilities with robots [26], in particular in industrial applications [11].

One such example is robot assisted disaster response teams. The ratio of operator to robots in robot assisted rescue task is generally about 2:1 [20]. However, this number can increase for more complex robotic systems, for example humanoid robots. Humanoid robots are equipped with high degrees of freedom and thus can be viewed a general solution to a wide range of tasks. As seen during the DARPA Robotics Challenge (DRC), in complex dynamic environments, humanoids require large teams of operators and shared control is indispensable [5]. In [32], the human robot interfaces used at DRC is analyzed and several design guidelines for future systems proposed. Among these points, there are two particularly relevant to this work. First, the authors state that a decrease in the amount of operator input needed to control the robot actually increases system performance. In fact, during the DRC operator errors led to major failures [1]. Second, less operators means less confusion and a more effective operation. Moreover, there are other reasons to reduce the number of operators, for instance during the emergency response to Fukushima, there was an explicit desired minimize the number of operators to reduce unnecessary radiation exposure [21].

In most situations, an operator is only required in specific parts of an operation. Knowing this, we can schedule these operator interactions so that a single operator can perform multiple tasks. The contributions of this work are: extending our preliminary ideas [34] in the following directions. *First*, we analyze the complexity of this problem. *Second* we present a sampling-based algorithm that allows us to design policies for a large number of tele-operators instead of a complete algorithm that only works for a small set of operators. *Third*, we allow re-planning of the robots alongside the operator. *Finally*, we present results of both simulated and physical experiments.

The remaining work is organized as follows: Section II discusses related literature and the original work being extended. Section III describes the problem, and Section IV defines the solution. In section V we present both software and hardware simulations, and conclude in Section VI along with future directions.

II. Related Work

The inspiration for this work comes from a lack of planning tools for mission autonomy that can help operators plan requirements and robot policies. There are some existing approaches, such as in [4], [19], [3], however, in Gerkey et al. [8] we find that many existing solutions are based on experimental evidence, and don't have formal approaches. We also find evidence of both a need for human collaboration with multi-robot systems [26], and an emphasis on experimental results in [25]. This is of particular interest as Hughes [12] argues that leveraging human operators alongside robots will result in robust systems that are more reliable, which has led to work on tele-operation using 1 : 1 operator:robot paradigms [33], [9].

Of particular interest and relevance to this research is the work by Trautman [27], [28], which incorporates *anticipated* human behavior into robot behavior so that the robots can execute complex plans. In [30], we also find evidence for the need to optimize human-robot interaction in industrial applications which involve repeated human-robot interaction in a controlled environment. We aim to improve upon these foundations by providing a formal methodology for scheduling operator attention across multiple robots.

This paper builds upon [34], in order to perform multi-robot planning [23], [24], and also has some similarities with that of Hauser [10], where the goal is to minimally displace obstacles in order for a robot to travel from its initial to goal state. Likewise, in this paper we aim to remove operator "collision obstacles" in order to minimize the ending time of the mission. Work by Wang et al. [29] attempts to deal with the large complexity of coordinating many robots and operators by dividing planning and scheduling into separate components. This is also an approach used by LaValle and Hutchinson [16], [17], and is incorporated into our solution as well.

III. Preliminaries

We are provided a set m of robots $\mathscr{A} = \{\mathscr{A}^1, \mathscr{A}^2, \cdots, \mathscr{A}^m\}$. Each robot $\mathscr{A}^i$ has a configuration space $\mathscr{C}^i$ that represents the set of all possible transformations, where the set of valid configurations in $\mathscr{C}^i$ is called the free space $\mathscr{C}^i_{free}$. Robots also have initial $q^i_I \in \mathscr{C}^i_{free}$ and goal $q^i_G \in \mathscr{C}^i_{free}$ configurations, and a trajectory τ^i. The trajectory takes the robot from the initial configuration $\tau^i(0) = q^i_I$ through $\mathscr{C}^i_{free}$ to the final configuration $\tau^i(l^i) = q^i_G$, where l^i is the length of the trajectory. The configuration space for all robots is defined as: $\mathscr{C} = \mathscr{C}^1 \times \mathscr{C}^2 \times \cdots \times \mathscr{C}^m$. When executing the trajectory, the robot $\mathscr{A}^i$ may encounter k^i critical segments $S^i = \{[s^i_1, f^i_1], \ldots, [s^i_k, f^i_k]\}$, during which it will require one of the p operator's supervision. Each interval $[s^i_j, f^i_j]$ is a closed subset of the trajectory and defines the start s^i_j and end f^i_j points of the segment $\tau^i[s^i_j, f^i_j]$ requiring attention. The set of all critical segments is defined as $S = S^1 \bigcup S^2 \bigcup \ldots \bigcup S^m$. Given a range of time $T = [0, t_f]$, where the mission is executing, we will attempt to minimize $t_f = max(t^1_f, \ldots, t^m_f)$, when all robots have finished.

A conflict is defined as $\bigcap_{i=0}^{p+1} [s^i_a, f^i_a] \neq \emptyset | a \in k^i$, meaning that $p+1$ robots require supervision at the same time. Let the set of invalid configurations X_{obs} indicate configurations where the number of robots requesting supervision exceeds p, and $X_{free} = X \backslash X_{obs}$ to be configurations where the number of requests does not exceed p.

Define a Coordination Space $X = [0, l^1] \times \cdots \times [0, l^m]$ (following a procedure similar to [15], chapter 7) representing all possible configurations of the robots along their trajectories. At $(0, \ldots, 0)$ all robots are in their initial configuration, and at $(l^1, \ldots, l^m) \in X$ all robots are in their final configuration.

Problem 1: Scheduling for Multiple Operators: *Given a set of robots $\mathscr{A}$, each with a trajectory τ^i, and a set of critical segments S^i, determine a policy $\pi^i : T \rightarrow [q^i_I, q^i_G]$ for each robot mapping time to a position along its trajectory such that 1) robots enter their critical sections only when an operator can supervise them, 2) the number of operators requested at any time is less than or equal to p, and 3) an effort is made to minimize the ending time of the mission t_f.*

Building on the previous problem, we might also ask: Can we get better results if we generate an alternative trajectory for robots so that they do not require supervision at the same time as other robots, thus avoiding operator attention "collisions" altogether?

Problem 2: Scheduling with Re-Planning: *Given a set of robots $\mathscr{A}$, a number of operators p, and a Workspace W where the robots reside, find trajectories τ through $\mathscr{C}_{free}$ that avoid operator conflicts and reduce the completion time of the mission.*

Problem 2 is a more concrete extension of Problem 1. As before, robots will have initial q^i_I and goal q^i_G configurations, and will be moving through a Workspace W. This workspace may include environmental obstacles $o \in \mathscr{O}$, such that the robots should only move in the obstacle-free region $W \backslash \mathscr{O}$. Thus, the free configuration space for the robots consists of removing all robot-obstacle collisions $\mathscr{C}_{obs} = \{q \in \mathscr{C} | \mathscr{A}(q) \bigcap O \neq \emptyset\}$ to get $\mathscr{C}_{free} = \mathscr{C} \setminus \mathscr{C}_{obs}$. Moreover, the workspace may contain dangerous regions $d \in \mathscr{C}_{free}$, which the robot can only traverse if there is an operator to supervise them.

IV. Methods

Here, we provide solutions to the problem defined in Section III.

A. Scheduling for Multiple Operators

In our earlier work [34], we provided a solution to the scheduling problem for both a single operator and multiple operators using a novel geometric approach. However, the complexity of creating the coordination space makes the solution intractable for problems with a large number of robots. We sketch a complexity proof of the problem as follows:

1) Computational Complexity Sketch: This problem can be shown to be NP-hard via proof by reduction [7]. In the Multiprocessor Scheduling problem, we have a set of J jobs, where each job j^i has a length l^i. Given a set of processors p, determine the minimum time needed to schedule all jobs such that they do not overlap. Starting with our problem, assume that each robot will require an operator during a single segment $[s^i_1, f^i_1]$, whose length is equal to τ^i, and that this segment occurs immediately at $t = 0$. The length of the segment now corresponds to the length of time needed from an operator (a job), and we must schedule these jobs across p operators (the processors). This is the Multiprocessor Scheduling problem where j jobs must be scheduled across p processors. We see then that our problem is at least as hard as the known NP-hard problem of Multiprocessor Scheduling.

2) A sampling based solution: Knowing that the problem is NP hard we then ask: how can we improve the runtime of the solution by using a heuristic approach?

We can improve the scalability of the solution in [34] by generating a path through the coordination space and using a lazy evaluation approach to check the locations visited, rather than generating the entire coordination space; using RRT^* described in [13], [14], and replicated in Algorithm 1 for reference. We modify the calls to *SampleFree* to use an obstacle detection function as in Algorithm 2, and to *CollisionFree* as in Algorithm 3.

Algorithm 1 Attention $RRT^*(x_{init}, S)$

```
1: $V \leftarrow \{X_{init}\}$
2: $E \leftarrow \emptyset$
3: for $j \in [1,n]$ do
4:     $x_{rand} \leftarrow SampleFree(j)$
5:     $x_{nearest} \leftarrow Nearest(G=(V,E), x_{rand})$
6:     $x_{new} \leftarrow Steer(X_{nearest}, x_{new})$
7:     if $ObstacleFree(X_{nearest}, x_{new})$ then
8:         $X_{near} \leftarrow Near(G=(V,E), x_{new}, |V|)$
9:         $V \leftarrow V \bigcap \{x_{new}\}$
10:        $x_{min} \leftarrow x_{nearest}$
11:        $c_{min} \leftarrow Cost(x_{nearest}) + c(Line(x_{nearest}, x_{new}))$
12:        for $x_{near} \in X_{near}$ do
13:            if $CollisionFree(x_{new}, x_{near}) \wedge Cost(x_{near}) + c(Line(x_{new}, x_{near})) \leq Cost(x_{near})$ then
14:                $x_{min} \leftarrow x_{near}$
15:                $c_{min} \leftarrow Cost(x_{near}) + c(Line(x_{near}, x_{new}))$
16:        for $x_{near} \in X_{near}$ do
17:            if $CollisionFree(x_{new}, x_{near}) \wedge Cost(x_{new}) + c(Line(x_{new}, x_{near})) \leq Cost(x_{near})$ then
18:                $x_{parent} \leftarrow Parent(x_{near})$
19:                $E \leftarrow (E \backslash \{(x_{parent}, x_{near})\}) \bigcup \{(x_{new}, x_{near})\}$
20: return $G=(V,E)$
```

Algorithm 2 checks if a point in X also resides in X_{free}. We accept the location to check x, a set of operators p, and the set of supervision segments S. In line 1, initialize the collision count to 0. Iterate over the attention segments in lines 2-3, checking to see if for each dimension of the location, it intersects a corresponding attention obstacle in line 4. If there is a collision, we increment the number in line 5, and if the number of collisions is greater than the number of operators, we break and indicate that x represents an invalid configuration, else the location represents a valid configuration (lines 6, 7).

Algorithm 3 expands upon Algorithm 2, checking if a line lies completely within X_{free} or intersects an obstacle. In lines 4, 5, we order the start and end point of dimension i of the line so that they are in ascending order, and in 6, 7, we take the max and min of the endpoints of the line, and the endpoints of the attention segment. If they have an overlap (line 8), then that

Algorithm 2 ObstacleFree(x, S, p)

```
1: $num\_colls \leftarrow 0$
2: for $S^i \in S$ do
3:     for $s^j \in S^i$ do
4:         if $s_s^j \leq x[i] \leq s_f^j$ then
5:             $num\_colls \leftarrow num\_colls + 1$
6:             if $num\_colls \geq p$ then
7:                 return True
8:             break
9: return False
```

Algorithm 3 CollisionFree$(x, start, end, p)$

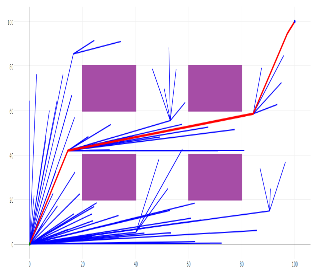
```
1: $num\_colls \Leftarrow 0$
2: for $S^i \in S$ do
3:     for $s^j \in S^i$ do
4:         $line_{lower} \leftarrow \min(start[i], end[i])$
5:         $line_{upper} \leftarrow \max(start[i], end[i])$
6:         $intersection_{lower} \leftarrow max(line_{lower}, s_s^j)$
7:         $intersection_{upper} \leftarrow min(line_{upper}, s_f^j)$
8:         if $intersection_{lower} \leq intersection_{upper}$ then
9:             $num\_colls \leftarrow num\_colls + 1$
10:            if $num\_colls \geq p$ then
11:                return False
12: return True
```

location within X will require an operator for that instance of s^j. If the number of intersecting attention segments is greater than the number of operators (lines 9-11), then the location is invalid.

Executing the modified RRT^* algorithm, we receive a path $h : [0,1] \to X$ through X_{free}, where the path goes from $h(0) = (0, ..., 0)$ to $h(1) = (l^1, ..., l^m)$, as seen in Figure 1. We then proceed as in our original work [34] by mapping h to the sequence of configurations $\tilde{x}^i$ that correspond to robot $\mathscr{A}^i$. To accomplish this, define $\sigma^i : T \to [0, l^i]$, which yields the location of $\mathscr{A}^i$ along τ^i at time t. Next, compose $\phi^i = \tau^i \circ \sigma^i$ to derive a mapping $T \to \mathscr{C}^i_{free}$. With this mapping, we can now find the configuration of robots at any time via $\phi^i(t) = \tau^i(\sigma^i(t))$. The method described here can handle robots with different-length trajectories by extending the corresponding dimensions accordingly. RRT^* is probabilistically complete, and thus will not yield $opt(t_f)$, however given a large enough set of samples, it will converge towards $opt(t_f)$.

B. Scheduling with Re-Planning

With the coordination space and corresponding path from Method A (Algorithm 1), we have a solution that avoids operator collisions. We now seek a solution that yields a better result if we are able to alter the trajectories of the robots. Extend the notation for Problem 1 as follows: First, denote U^i to be the set of intervals $U^i = \{[b_1^i, c_1^i], ..., [b_k^i, c_k^i]\}$ where for each interval $[b_j^i, c_j^i]$, an operator is not available to supervise robot $\mathscr{A}^i$. Second, consider the dangerous regions $d \in \mathscr{C}^i_{free}$ to be obstacles within $\mathscr{O}^i$ when the robot does not have an


(a)

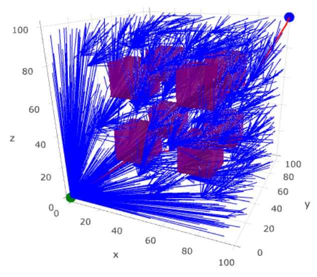
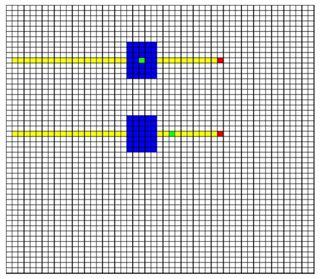
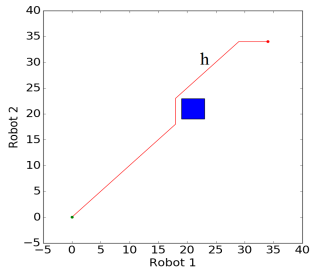
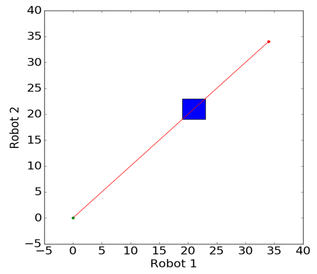

(b)

Fig. 1. *RRT** Examples
(a) *RRT** visualization for two robots avoiding operator attention collisions (purple). The axes correspond to the positions of robots 1 and 2 along their trajectories. The red line represents a valid path through the coordination space that can be mapped back into policies for the robots. (b) Similar example as (a) for three robots.


(a)



(b)



(c)

Fig. 2. Example Coordination Space

(a) 2 Robots (red), and trajectories (yellow) in a 2-dimensional Environment. Blue areas require an operator's supervision. (b) 2-dimensional Coordination Space resulting from (a). The axes correspond to the positions of robots 1 and 2 along their trajectories. The red line represents a valid path through the coordination space that can be mapped back into policies for the robots. The blue squares are areas of operator attention collision. (c) Coordination space from (b), with the desired (optimal) policy shown as the red line.

operator to supervise it $d \in O^i | t \in U^i$. Third, define a procedure *replan*, that can re-plan the trajectory of robot such that the robot will not perform critical maneuvers during times U^i, and returns a new S^i and τ^i. As an example, *replan* can be a path-planning algorithm that avoids dangerous regions of the environment when an operator will not be available). For illustrative purposes, an example of *replan* can be found in Section V.

We solve this scheduling and path-planning problem by comparing the path h through the coordination space with a desired path h_{des}, a path from the origin to the opposite corner (the shortest path through X). Building on the example found in Figure 2(a, b), we can visualize an ideal path as the one in Figure 2(c), which is currently not possible due to an operator conflict. As we follow our ideal path through the coordination space, we may encounter an obstacle at point $x \in X$ such that $h_{des}(x) \bigcap X_{obs} \neq \emptyset$. In this case, we must either plan around the obstacle, tuning the velocity of the involved robot(s) as in the solution for Problem 1 or create alternate plans for the robots so that the number of robots requesting supervision during that time can be satisfied by the p operators, and potentially reducing the total mission run-time by not having to wait for an operator to become available.

As before, we have to consider conflicts within the resulting coordination space S. We then allow for the possibility of re-planning trajectories to eliminate overlapping supervision requests. However, by altering the trajectories of robots to avoid an operator attention collision, we alter *when* the robots involved will reach later parts of their trajectory, affecting when they will require operator supervision in subsequent sections. This in turn will modify X, potentially moving, creating, or removing later obstacles. As an illustration, in Figure 3(a), we have two robots, which are expected to simultaneously enter regions requiring supervision. The resulting coordination space in Figure 3(b) shows that we can resolve this by stopping robot 1 (represented by the vertical segment of h) until robot 2 has exited the region. We can attempt to circumvent this conflict by having robot 2 plan a trajectory that partially avoids the conflict. However, the additional time needed to circle the region will cause it to encounter its second critical section at a later time - the same time as robot 1 (Figure 3(c)) - resulting in another operator conflict obstacle that must be dealt with.

In order to handle the complexity of the problem, this paper uses a heuristic approach as shown in Algorithm 4. We begin by constructing a coordination space X and path h through X_{free} (lines 1, 2) using *RRT** as in Solution 1. This provides us with our standard solution. We then create what would be an optimal solution h_{opt}, given by no robots requiring supervision (lines 3, 4). Next, we check to see if the optimal solution is feasible by looking for any collisions with obstacles in the actual coordination space (line 5), which can be done using Algorithm 3. If the optimal solution is invalid, we can correct it in two ways:

1) Accept the obstacle, and alter our policy (as in Solution 1).
2) Re-plan the robots that have a conflict in order to eliminate the obstacle.

We now describe how to re-plan the robots. Begin by sampling along h_{opt} with Algorithm 2; when a collision is encountered, denote this as an obstacle $z \in X_{obs}$, where $z_{\mathscr{A}}$ is the set of robots and their segments z_S that are responsible for the conflict (line 6). By selecting the obstacle z that is closest to the origin, we will preserve any previous work, and only affect the policy that remains *after* encountering

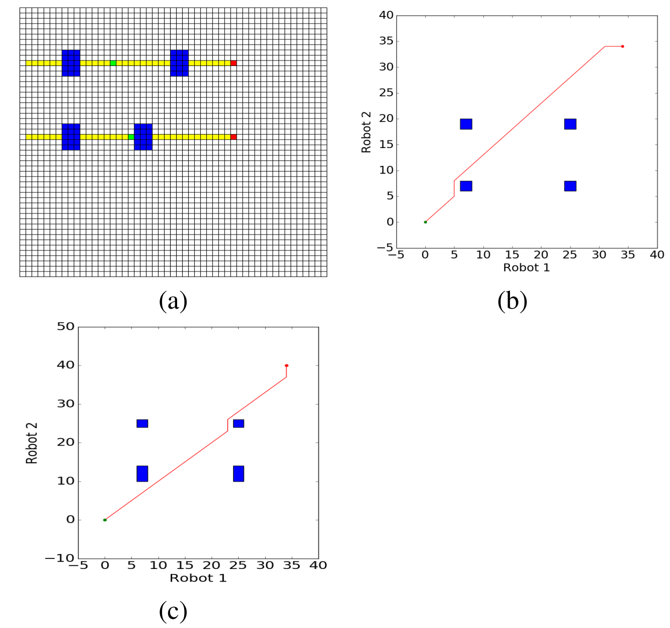


Fig. 3. Shifting Conflict Regions

(a) Environment and expected trajectories; (b) Original Coordination Space; (c) Final Coordination Space

this obstacle. For each robot involved in the collision, use the mapping defined in Solution 1 to determine the times U^i during which it needs supervision (the times corresponding to intersecting the obstacle) in lines 8, 9. Then use *replan* and U^i to re-plan the remaining portion of the robot's trajectory, returning and saving a new set of segments during which it requests supervision, and trajectory τ^j (lines 10, 11). If no modification is possible, the original trajectory is returned. Given that there are $|z_{\mathscr{A}}|$ robots involved in the collision represented by obstacle z, we need to remove $g = |z_{\mathscr{A}}| - p$ robots from this collision in order to remove the obstacle. In line 12, we select the g robots involved in z that have the shortest trajectory lengths (minimum effect on total mission time). Next, we compare a path through the new coordination space with the g re-planned robots, to the path through the original coordination space (lines 13 - 18). If the new path given by using the re-planned robot trajectories is shorter than the original solution, we update our solution (lines 19, 20), otherwise, we maintain the current robot trajectories and instead add that obstacle to the obstacles in S_{opt} (lines 21, 22) so that the robot velocities will be tuned.

Once there are no more obstacles intersecting the optimal policy h_{opt}, we are left with a policy that will result in no operator conflicts.

V. Simulation

In this section, we describe our implementation of RRT^* and re-scheduling, and provide both simulated and physical experiments.

Algorithm 4 Heuristic_Re-Planning(s, p)

```
1:  $X = [0, l^1] \times \cdots \times [0, l^m]$
2:  $h \leftarrow RRT^*(X, (0, \ldots, 0), (l^1, \ldots, l^m), S, p)$
3:  $S_{opt} \leftarrow \{\emptyset\}$
4:  $h_{opt} \leftarrow \text{line}(X, (0, \ldots, 0), (l^1, \ldots, l^m))$
5:  while $h_{opt} \bigcap X_{obs} \neq \{\emptyset\}$ do
6:      $z \leftarrow \text{First\_Obstacle}(h_{opt}, S, p)$
7:      $alternatives \leftarrow \emptyset$
8:      for $\mathscr{A}^i \in z_{\mathscr{A}}$ do
9:          $U^i_{test} \leftarrow \text{Unsupervised\_Times}(h_{opt}, \mathscr{A}^i, z_S)$
10:         $S^j, \tau^j \leftarrow r(\mathscr{A}^i, U^i_{test})$
11:         $alternatives \rightarrow alternatives \bigcup (S^i, S^j, \tau^j)$
12:     $alternatives = min(g, alternatives)$
13:     $S_{alt} \leftarrow S$
14:     for $(S^i, S^j, \tau^j) \in alternatives$ do
15:         $S_{alt} \leftarrow (S_{alt} \setminus S^i) \bigcup S^j$
16:     $h_{alt} \leftarrow RRT^*(X, (0, \ldots, 0), (l^1, \ldots, l^m), S_{alt}, p)$
17:     if $len(h_{alt}) \leq len(h)$ then
18:         $S \leftarrow S_{alt}$
19:         $h \leftarrow h_{alt}$
20:     else
21:         $S_{opt} \leftarrow z_S$
22:         $h_{opt} \leftarrow RRT^*(X, (0, \ldots, 0), (l^1, \ldots, l^m), S_{opt}, p)$
23: Output: $h_{opt}$
```

A. Software Simulation for Scheduling with Re-Planning

Here, we describe and display our simulation. We also provide an example *replan* algorithm to allow robots to re-plan their trajectory around the unsafe areas if an operator will not be available during certain times.

Example Re-plan Algorithm: The *replan* algorithm functions as follows: Intuitively, we are striving for the shortest path between x^i_{init} and x^i_{final} within the physical environment. This can be achieved by using the A^* algorithm [2], [18]. However, the shortest path may pass through obstacles requiring supervision. Given the times during which an operator will *not* be available, U^i, we can modify A^* as follows: Denote the starting time of the mission as $T_i = 0$. Augment A^*'s nodes with an additional *time* parameter. As we expand each node, we set it's neighbors *time* attribute to $time + travel\_time$ where *time* is the current time-stamp, and $travel\_time$ is the the time required to move from the current node to the neighbor. This will give an estimate of when the robot will reach a location within the environment. If the neighbor is inside a conflict *and* the neighbor's *time* lies within U^i, then we treat the neighbor as an obstacle that cannot be visited. This implementation of A^* nicely solves the problem, and will result in a path that leads out of and around conflicts during the operator-denied intervals, as seen in Figure 4.

We now present an example to illustrate our work in Figure 4. We show a known 2-dimensional $\mathbb{R}^2$ environment with three robots. The environment contains a set of areas (shown in blue) where the ground may be too spongy to support a robot,

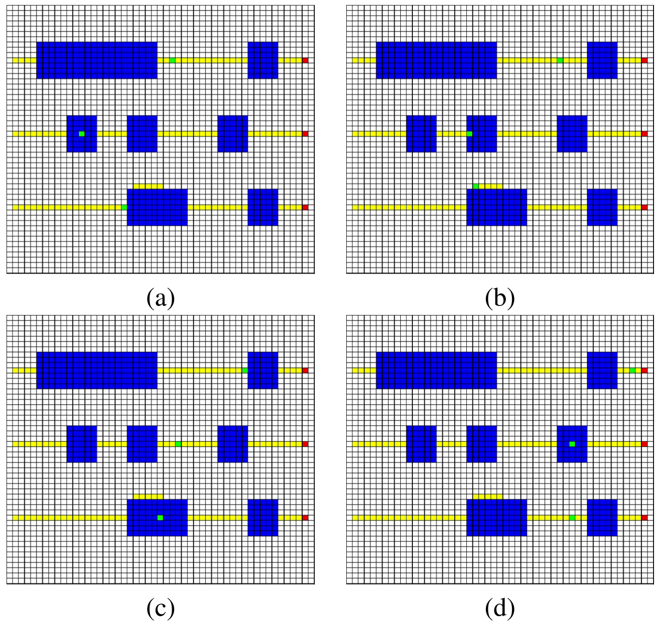


Fig. 4. Example Simulation Environment

Simulation Environment; Robots 1, 2, 3 are arranged from top to bottom. (a) Robot 3 is paused to allow Robot 2 to receive supervision. (b) Robot 3 has re-planned around the critical region, allowing Robot 2 to receive supervision. (c) Robot 1 is paused to allow Robot 3 to receive supervision. (d) All robots proceed to their goal locations.

therefore an operator must supervise any robots inside of these areas in case the robot starts to sink.

The example has also been designed to show several operator attention "collision" types. As all three robots move from left to right, we see the following potential operator requests:

- $\mathscr{A}^1$ requiring an operator
- $\mathscr{A}^1$ and $\mathscr{A}^2$ simultaneously requiring an operator
- $\mathscr{A}^1, \mathscr{A}^2, \mathscr{A}^3$ simultaneously requiring an operator
- $\mathscr{A}^3$ requiring an operator while $\mathscr{A}^1$ and $\mathscr{A}^2$ leave their critical regions
- $\mathscr{A}^2$ requiring an operator
- $\mathscr{A}^1$ and $\mathscr{A}^2$ simultaneously requiring an operator

In Figure 5, we see the coordination space corresponding to the solution *without* re-planning (a, b), and *with* re-planning (c, d). Many of the collisions in Figure 5(c) have resolved by tuning the velocity of stopping one or more of the robots involved. Comparing the initial and final coordination spaces, we see that an additional set of collisions has been introduced. This arises from $\mathscr{A}^3$ re-planning around part of its supervision segments, entering and leaving a dangerous region, creating two instances where it will require supervision. This allows the mission to finish faster, at $t = 67$, rather than if the robot had simply paused, which would finish at $t = 69$.

The simulation was set up using a planar 2-dimensional world. This world was then populated with a set of dangerous regions with random locations and dimensions, and robots were assigned random starting and goal locations such that $q_I^i, q_G^i \in C_{free}^i$, and trajectories taking them from their start to goal as in Figure 6. A set of 10 environments were generated,

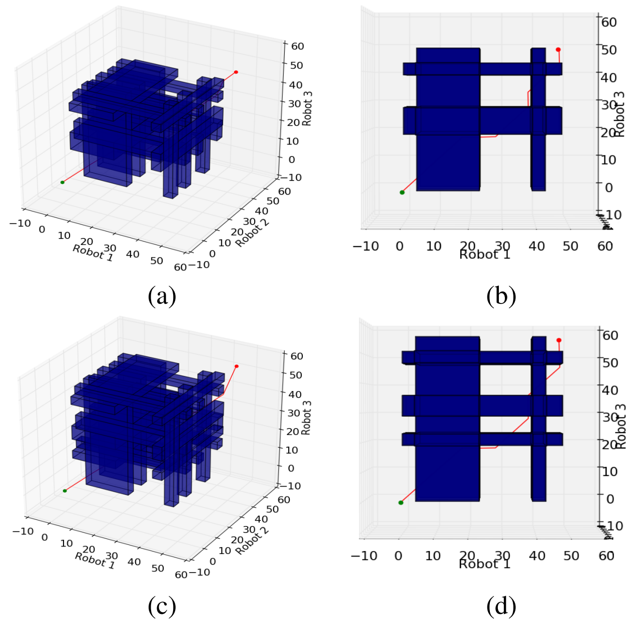


Fig. 5. Example Simulation Coordination Space

(a) Original Coordination Space; (b) Side view of (a); (c) Final Coordination Space; (d) Side view of (c)

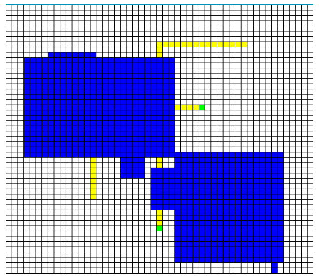


Fig. 6. Example Random Environment

Example of a randomly-generated environment and trajectories intersecting critical regions.

where each trial used a newly generated environment and set of robot starting and goal positions. Each of the 10 environments used in trials consisted of 2, 4, or 8 robots, moving at 1 cell/second. Each of these was in turn solved using both Scheduling and Scheduling with Re-Planning, with 1, 2, 4, or 8 operators. The results are shown in Table I.

We find a marked increase in savings as the number of robots increases, as re-scheduling removes a greater number of obstacles. In cases where the number of operators is equal to or greater than the number of robots there are no savings as expected, since this would not result in any attention obstacles to avoid. Over the course of the trials, all tests with 2, and 4 robots completed successfully. Of the trials with 8 robots and 1 operator, no solutions could be found with the provided RRT^* parameters. With 2 operators, 30% completed, and 60% for 4 operators. This is likely due to the relatively low number of samples taken (25,000) when running Attention RRT^*, and the large *steer* length, which did not allow for finding paths in

TABLE I
AVERAGE TIME SAVINGS WHEN RE-PLANNING VS ATTENTION RRT^*

Robots	Operators	Average Savings
2	1	1.126
2	2	0
2	4	0
2	8	0
4	1	1.937
4	2	3.402
4	4	0
4	8	0
8	1	NA
8	2	0.218
8	4	5.284
8	8	0

narrow gaps between obstacles. Tuning these parameters lies outside the scope of this paper, but is nonetheless an interesting problem we hope to look into in future work.

B. Hardware Experiment for Scheduling with Re-Planning

In this section, we further motivate the efficacy of the approach by applying it to hardware rather than being limited to simulations. The hardware demonstration assumed a single operator spread across three robots. The example was implemented using a hardware/software test-bed composed of line-following robots in a discrete grid environment, and a distributed software system. The robots' bodies and electronics are based on the open-source Arduino Controlled Servo Robot reference design [22]. The reference design was augmented using three infrared light detectors for use in line detection, and an XBee DigiMesh [31] radio transceiver for use in robot-controller communication.

The software component of the test-bed is distributed across one controller and all three robots. The robots observe the world through line sensors, which inform them of whether they are:

1) centered on a grid line
2) left of a grid line
3) right of a grid line
4) at a grid intersection (i.e. a world-state)
5) in an unknown part of the environment

The robots maintain knowledge only of their own world-state (position and orientation) using a deterministic finite state machine of infinite size, the transition function of which is implemented using a second transition-state machine that ensures – given some assumptions about the uniformity of the environment – that the robots' inter-state path does not deviate from a grid line (i.e. that while transitioning between world-states, their transition-state is one of the first three of the five previously enumerated states). State transitions are performed upon the instruction of the controller, on which the presented algorithms run. The controller does not maintain knowledge of the robots' world-states, but can query each robot individually as needed to determining if collision obstacles exist.

The simulation performed in Figure 7 began with a model environment as in Figure 7(a), where each of the 3 robots were assigned trajectories, expecting to traverse areas of the

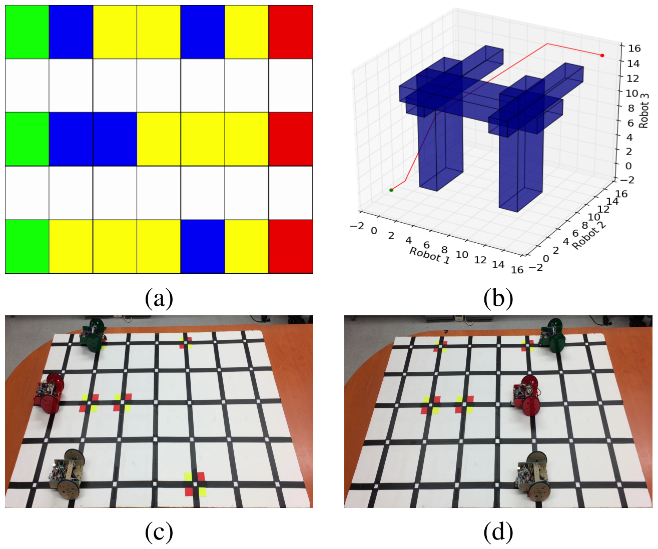


Fig. 7. Hardware Experiment Example

(a) Simulation Environment; (b) Resulting Coordination Space; (c) Corresponding hardware simulation at $t = 1$; (d) Hardware simulation at $t = 5$

environment that require operator supervision (shown in blue), and ending at their goal (red). A single operator is available to oversee robots inside of critical areas. The physical environment represents the supervision-required areas as red/yellow squares, in the same positions as in the simulation. We then generated a coordination space (b), which provides an attention-collision-free set of policies that enable the robots to execute their trajectories while guaranteeing that a single operator would not have their attention split among multiple robots. The robots received and executed their corresponding policies, moving and pausing when appropriate, such that no more than a single robot entered a supervision region at a time. Further experiments and videos can be found at: http://users.cis.fiu.edu/~jabobadi/oa/

The hardware experiments show a successful run using our methodology to coordinate three different robots provided a single operator. The robots finished in the shortest time possible, and the operator was not overburdened.

VI. CONCLUSION

In this work, we provide a methodology for converting a set of robot trajectories and supervision requests into a policy that guides the robots in such a way that operators can oversee critical sections of robot plans without being over-allocated. Moreover, we indicate how to decide whether re-planning a robot would yield a better plan compared to slowing down or stopping to wait for an operator.

In the future, we would like to incorporate how robot movements affect an operator's effectiveness in overseeing them [6]. In particular, we are especially interested in incorporating operator context-switching time. This may be accomplished by extending obstacles towards the origin in order to resemble the extra time needed for an operator to adjust to their new task. Another possible avenue for further work is incorporating uncertainty of a robot's trajectory. Here, in the case of uncertainty regarding when the operator will be required, we can uniformly expand all obstacles. The resulting coordination space will yield a path h which has a more conservative solution. This may also be combined with a receding horizon approach to allow for rapidly changing robot plans in a more dynamic environment, compared to a controlled industrial setting.

Future avenues of research will include uncertainty models on how long operators take to complete their task. We also plan to extend the configuration space use into a phase that can include the robot dynamics and can model situations where the robot needs some time to accelerate/decelerate and do more realistic motions.

Acknowledgment

Mr. Zanlongo would like to thank FIU's Applied Research Center (ARC) and the Department of Energy Office of Environmental Management for providing the opportunity to be a DOE Fellow under the Science and Technology Workforce Development Initiative and facilitating research opportunities under the DOE-FIU Cooperative Agreement DE-EM0000598.

References

[1] Christopher G Atkeson, Benzun P Wisely Babu, Nandan Banerjee, Dmitry Berenson, Christoper P Bove, Xiongyi Cui, Mathew DeDonato, Ruixiang Du, Siyuan Feng, Perry Franklin, et al. No falls, no resets: Reliable humanoid behavior in the darpa robotics challenge. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 623–630. IEEE, 2015.

[2] J. Beardwood, J.H. Halton, and J.M. Hammersley. The shortest path through many points. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 55, pages 299–327. Cambridge Univ Press, 1959.

[3] J.W. Crandall, M.L. Cummings, M. Della Penna, and Paul M.A. de Jong. Computing the effects of operator attention allocation in human control of multiple robots. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(3):385–397, 2011.

[4] M.L. Cummings and P.J. Mitchell. Operator scheduling strategies in supervisory control of multiple uavs. *Aerospace Science and Technology*, 11(4):339–348, 2007.

[5] Mathew DeDonato, Velin Dimitrov, Ruixiang Du, Ryan Giovacchini, Kevin Knoedler, Xianchao Long, Felipe Polido, Michael A Gennert, Taşkın Padır, Siyuan Feng, et al. Human-in-the-loop control of a humanoid robot for disaster response: A report from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):275–292, 2015.

[6] Anca D Dragan, Shira Bauman, Jodi Forlizzi, and Siddhartha S Srinivasa. Effects of robot motion on human-robot collaboration. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 51–58. ACM, 2015.

[7] Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[8] B.P. Gerkey and M.J. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[9] Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.

[10] K.K. Hauser. Minimum constraint displacement motion planning. In *Robotics: Science and Systems*, 2013.

[11] Evert Helms, Rolf Dieter Schraft, and M Hagele. rob@ work: Robot assistant in industrial environments. In *IEEE International Workshop on Robot and Human Interactive Communication*, pages 399–404. IEEE, 2002.

[12] T. Hughes. Human-automation coordination in multi-uav control. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 6315, 2008.

[13] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104, 2010.

[14] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[15] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.

[16] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goals. In *IEEE International Conference on Robotics and Automation*, pages 2847–2852, April 1996.

[17] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14(6):912–925, December 1998.

[18] James Matthews. Basic a* pathfinding made simple. *AI Game Programming Wisdom*, pages 105–113, 2002.

[19] R. Murphy and J. Shields. The role of autonomy in dod systems. Technical report, Technical report, Department of Defense, Defense Science Board Task Force Report, 2012.

[20] Robin R Murphy. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):138–153, 2004.

[21] Keiji Nagatani, Seiga Kiribayashi, Yoshito Okada, Kazuki Otake, Kazuya Yoshida, Satoshi Tadokoro, Takeshi Nishimura, Tomoaki Yoshida, Eiji Koyanagi, Mineo Fukushima, et al. Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots. *Journal of Field Robotics*, 30(1):44–63, 2013.

[22] Oomlout. Arduino controlled servo robot (serb), 2008.

[23] L.E. Parker. Multiple mobile robot systems. In *Springer Handbook of Robotics*, pages 921–941. Springer, 2008.

[24] S.S. Ponda, L.B. Johnson, A. Geramifard, and J.P. How. Cooperative mission planning for multi-uav teams. In *Handbook of Unmanned Aerial Vehicles*, pages 1447–1490. Springer, 2015.

[25] Sarvapali Ramchurn, Joel Fischer, Yuki Ikuno, Feng Wu, Jack Flann, and Antony Waldock. A study of human-agent collaboration for multi-uav task allocation in dynamic environments. 2015.

[26] Julie Shah, James Wiken, Brian Williams, and Cynthia Breazeal. Improved human-robot team performance using chaski, a human-inspired plan execution system. In *6th international conference on Human-robot interaction*, pages 29–36. ACM, 2011.

[27] Pete Trautman. Probabilistic tools for human-robot cooperation. In *Human Agent Robot Teamwork Workshop HRI*, 2012.

[28] Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[29] J.J. Wang, Y.F. Zhang, L. Geng, J.Y.H. Fuh, and S.H. Teo. Mission planning for heterogeneous tasks with heterogeneous uavs. In *International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 1484–1489. IEEE, 2014.

[30] Ronald Wilcox, Stefanos Nikolaidis, and Julie Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. *Robotics Science and Systems VIII*, pages 441–448, 2012.

[31] Product Manual XBee and XBee-PRO DigiMesh. 2.4 oem rf modules, digi international, 2010.

[32] Holly A Yanco, Adam Norton, Willard Ober, David Shane, Anna Skinner, and Jack Vice. Analysis of human-robot interaction at the darpa robotics challenge trials. *Journal of Field Robotics*, 32(3):420–444, 2015.

[33] Erkang You and Kris Hauser. Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input. In *Robotics: science and systems*, volume 7, page 354, 2012.

[34] Sebastian A Zanlongo, Mahbubur Rahman, Franklin Abodo, and Leonardo Bobadilla. Multi-robot planning for non-overlapping operator attention allocation. In *IEEE International Conference on Robotic Computing*, pages 109–112. IEEE, 2017.