Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

A Sampling-Based Approach for Heterogeneous Coalition Scheduling with Temporal Uncertainty

Andrew Messing*†, Jacopo Banfi*‡, Martina Stadler‡, Ethan Stump§,
Harish Ravichandar†, Nicholas Roy‡ and Seth Hutchinson†
*Equal Contribution
†IRIM, Georgia Institute of Technology, Atlanta (GA) 30332, USA
‡CSAIL, Massachusetts Institute of Technology, Cambridge (MA) 02139, USA
§DEVCOM ARL, Adelphi (MD) 20783, USA

***Abstract*—Scheduling algorithms for real-world heterogeneous multi-robot teams must be able to reason about temporal uncertainty in the world model in order to create plans that are tolerant to the risk of unexpected delays. To this end, we present a novel sampling-based risk-aware approach for solving Heterogeneous Coalition Scheduling with Temporal Uncertainty (HCSTU) problems, which does not require any assumptions regarding the specific underlying cause of the temporal uncertainty or the specific duration distributions. Our approach computes a schedule which obeys the temporal constraints of a small number of heuristically-selected sample scenarios by solving a Mixed-Integer Linear Program, along with an upper bound on the schedule execution time. Then, it uses a hypothesis testing method, the Sequential Probability Ratio Test, to provide a probabilistic guarantee that the upper bound on the execution time will be respected for a user-specified risk tolerance. With extensive experiments, we demonstrate that our approach empirically respects the risk tolerance, and generates solutions of comparable or better quality than state-of-the-art approaches while being an order of magnitude faster to compute on average. Finally, we demonstrate how robust schedules generated by our approach can be incorporated as solutions to subproblems within the broader Simultaneous Task Allocation and Planning with Spatiotemporal Constraints problem to both guide and expedite the search for solutions of higher quality and lower risk.**

I. Introduction

Heterogeneous multi-robot systems offer the potential to solve complex large-scale problems in domains such as agriculture [1] for imaging or fertilization, military applications such as surveillance [2], and disaster recovery [3]. These problems involve interdependent tasks, require a diverse set of capabilities, and have spatiotemporal constraints. In the real world, there are numerous factors that can cause both durations of tasks and durations of task transitions to be uncertain (e.g., stale map information, battery levels causing variance in individual robots' speeds, unexpected delays). Without reasoning about possible temporal uncertainty, these factors can compound through the entire execution of a plan and result in a plan execution longer than expected.

In this work, we introduce a novel uncertainty-aware scheduling problem, the Heterogeneous Coalition Scheduling with Temporal Uncertainty (HCSTU) problem, which explicitly considers the uncertainties in the time needed to execute a task and to transition between tasks. In our setting, we assume that robots belonging to a heterogeneous team have already been assigned to one or more tasks to be carried out in different locations of a given map, such as the one shown in Fig. 1. The goal is to compute a plan in the form of a partial ordering of the tasks which minimizes the makespan. Existing approaches that deal with duration uncertainties typically either solve a determinized version of the problem or reformulate the problem into a more general form. However, such reformulations tend to be limited to small teams (less than 5 robots) and unimodal uncertainty [4], or can often lead to overly conservative or optimistic solutions [5, 6].

Fig. 1: Polypixel, a simulated urban environment used for our experiments (as seen in [7]), with its graph-based abstraction superimposed. Vertices show locations where a task has been assigned to one or more robots of a heterogeneous team. Edges represent travel paths between two locations.

To solve the HCSTU problem, we present a novel sampling-based risk-aware algorithm named Coalition Scheduling with Heuristic Sample Selection and Risk Guarantee (CS-HSSRG). Unlike approaches that either do not provide guarantees on risk [8, 9] or empirically determine a risk tolerance offset to use internally to their algorithm [10, 6], our approach allows users to specify the desired level of risk and ensures that such level will not be exceeded. Informally, risk is defined as the probability that the time needed to execute all tasks will exceed some upper bound computed by the algorithm along with the partial tasks ordering.

Our approach is agnostic to the specific underlying cause of the temporal uncertainty and the specific duration distributions. Instead, we assume access to either a model of uncertainty or a generator that can generate samples of scenarios of

possible task and task transition durations. The CS-HSSRG algorithm uses a heuristic function to select a small but representative set of such scenarios, and subsequently uses a Mixed-Integer Linear Program (MILP) to compute a schedule that satisfies the temporal constraints associated with them. The Sequential Probability Ratio Test [11] is then used to provide the theoretical guarantee that a user-specified risk tolerance is met.

In an extensive experimental campaign, we show that the bound on the risk does not lead to overly conservative solutions, and that the CS-HSSRG algorithm is extremely efficient at computing robust plans in problems with up to 15 robots when compared to two state-of-the-art methods [6, 9]. CS-HSSRG is also tested as a component for solving a class of problems that recently appeared in the literature, dubbed Simultaneous Task Allocation and Planning with Spatiotemporal Constraints (STAP-STC) problems [12]. These problems take a holistic view of heterogeneous multi-robot coordination by simultaneously considering four fundamentally intertwined coordination problems: *what* (task planning), *who* (task allocation), *when* (scheduling), and *how* (motion planning). We show that the use of CS-HSSRG for dealing with the scheduling part of the problem within the state-of-the-art multi-agent coordination planner GRSTAPS [12] results in decisions about what tasks to include in the plan and what robots to allocate to those tasks that are robust to temporal uncertainty. The proposed approach is fundamentally centralized and offline, similarly to many recent works in the same area [13, 3, 14, 12, 7], and centralization of the planner is often imposed by operational constraints. However, our approach is experimentally efficient enough to allow online replanning phases which may be triggered by different events such as subteams becoming disconnected. A comparison to fully decentralized approaches to multi-agent planning under uncertainty [15] are outside the scope of this paper as those approaches solve a fundamentally different problem.

II. Related Work

A rich body of work has addressed the multi-robot scheduling problem [16] and the closely related multi-robot task allocation (MRTA) problem [17, 18]. Our work focuses on a variant of the multi-robot scheduling problem where robots can only participate in a single task at a time, but multiple robots can form coalitions and collectively executable a single task. As such, our problem falls under the single-task (ST) robots and multi-robot (MR) tasks categorization from Gerkey and Mataric's widely used taxonomy [17] and under the Synchronization and Precedence Constraints (SP) categorization from Nunes's extension to the taxonomy [16].

In robotics, scheduling with temporal uncertainty is often formulated more generally [17, 19, 16] as either an uncertainty-based variant of a Simple Temporal Problem (STP) [5] or a Resource Constrained Project Scheduling Problem (RCPSP) [4]. For the interested reader, a detailed description of the STP and its uncertain variations can be found in [20]. Additionally, a detailed description of the RCPSP and its uncertain variants can be found in [21].

The Resource Constrained Project Scheduling Problem with Uncertainty (RCPSPU) is an extension to RCPSP that incorporates two types of uncertainty: resource uncertainty and temporal uncertainty. We focus only on the works that consider temporal uncertainty as we assume that the set of robots (i.e., our resources) is known. To tackle this problem, Varakantham *et al.* [6] present an Sample Average Approximation (SAA)-based approach dubbed SORU for generating a risk-aware schedule. SORU uses hard resource constraints and soft temporal constraints, which sometimes allow it to violate the temporal constraints in the solution. In contrast, our approach uses hard temporal constraints to avoid temporal constraint violations. Song *et al.* [9] instead use Conditional Value-At-Risk [22] as part of a branch-and-bound framework to solve the problem. Due to their demonstrated success, we use these two methods as two of the baseline for our experiments in Section V. Some common limitations of these two methods are that none of them provide a theoretical guarantee, none of them are demonstrated with more than 5 robots, and both of them are demonstrated only with uni-modal distributions for durations. In contrast, our presented approach provides a theoretical guarantee on its risk tolerance, is demonstrated to be more efficient even when solving problems with up to 15 robots, and is agnostic to the underlying duration distribution.

III. Problem Description

Consider a heterogeneous team of R robots that must collectively execute N tasks. Each task can be executed by one or more robots (i.e. a coalition), but each robot can only participate in a single task at a time (ST-MR-TA). Let task durations and robots' travel times between task locations be stochastic. The problem is for a centralized planner to compute a partial ordering of the tasks which minimizes the makespan, namely, the overall time for the robots to complete all the tasks. We first formulate this problem assuming deterministic task duration and robots' travel times, and then generalise to the stochastic case.

A. Heterogeneous Coalition Scheduling Problem

In the deterministic version of the problem, a task τ_i is defined by a duration d_i as the amount of time needed to execute τ_i for the robots that are assigned to it and an initial transition ϕ_i as the minimum amount of time needed for all of the robots assigned to τ_i to reach it from their individual initial configurations.

Tasks are temporally constrained by a set of precedence constraints $\mathcal{P}$ and a set of mutex constraints $\mathcal{M}$. A *precedence constraint ($\tau_i \prec \tau_j$)* is a temporal relationship between two tasks τ_i and τ_j that requires that the execution of τ_i concludes before τ_j starts. Each precedence constraint $\tau_i \prec \tau_j$ has a transition duration ϕ_{ij} that defines the minimum amount of time needed for all robots assigned to both τ_i and τ_j to travel from the terminal configuration of τ_i to the initial configuration

of τ_j. If there are no robots that are assigned to both τ_i and τ_j then $\phi_{ij} = 0$.

A mutex constraint is created between each pair of tasks to which the same robot is assigned. A *mutex constraint ($\tau_i \leftrightarrow \tau_j$)* between two tasks τ_i and τ_j represents the disjunction that either τ_i must conclude before τ_j starts ($\tau_i \prec \tau_j$) or τ_j must conclude before τ_i starts ($\tau_j \prec \tau_i$). If a robot is assigned to two tasks τ_i and τ_j where there already exists a precedence constraint (e.g. $\tau_i \prec \tau_j$) then the precedence constraint supersedes the mutex constraint as there is no decision to be made on the ordering of the two tasks. As such, $\mathcal{P} \cap \mathcal{M} = \emptyset$. Each mutex constraint $\tau_i \leftrightarrow \tau_j$ has a pair of transition durations ϕ_{ij} and ϕ_{ji} that define the minimum amount of time needed for all robots assigned to both τ_i and τ_j to travel from the terminal configuration of τ_i to the initial configuration of τ_j and from the terminal configuration of τ_j to the initial configuration of τ_i respectively.

The Heterogeneous Coalition Scheduling (HCS) problem is fully defined by:

- a set of task durations $\mathcal{D} = \{d_i \mid i \in \mathcal{I}\}$ where $\mathcal{I} = \{1, \ldots, N\}$,
- a set of initial task transitions $\Phi_{init} = \{\phi_i \mid i \in \mathcal{I}\}$
- a set of precedence constraints $\mathcal{P} \subseteq \mathcal{I}^2$,
- a set of mutex constraints $\mathcal{M} \subseteq \mathcal{I}^2$,
- and a set of task transition durations $\Phi = \{\phi_{ij} \mid (i,j) \in \mathcal{P}\} \cup \{\phi_{ij} \mid (i,j) \in \mathcal{M}\} \cup \{\phi_{ji} \mid (i,j) \in \mathcal{M}\}$

The HCS problem requires the selection of a set of task orderings ρ with one for each mutex constraint that minimizes the makespan C or total time needed to execute all tasks while considering the task durations and task transition durations. The HCS problem can be formulated as the following MILP:

$$
\begin{aligned}
&min \;\; C && && \text{(1a)}\\
&s.t. \;\; C \geq c_i && \forall i \in \mathcal{I} && \text{(1b)}\\
&\quad s_j \geq c_i + \phi_{ij} && \forall (i,j) \in \mathcal{P} && \text{(1c)}\\
&\quad s_j \geq c_i + \phi_{ij} - M(1-\delta_{ij}) && \forall (i,j) \in \mathcal{M} && \text{(1d)}\\
&\quad s_i \geq c_j + \phi_{ji} - M\delta_{ij} && \forall (i,j) \in \mathcal{M} && \text{(1e)}\\
&\quad s_i \geq \phi_i && \forall i \in \mathcal{I} && \text{(1f)}\\
&\quad \delta_{ij} \in \{0,1\} && \forall (i,j) \in \mathcal{M} && \text{(1g)}
\end{aligned}
$$

where s_i is the start time for the i^{th} task, c_i is the completion time for the i^{th} task ($c_i = s_i + d_i$), δ_{ij} is a boolean indicator that is 1 when the mutex constraint between the i^{th} and j^{th} tasks ($\tau_i \leftrightarrow \tau_j$) has been reduced to the precedence constraint from the i^{th} task to the j^{th} task ($\tau_i \prec \tau_j$) and 0 when the mutex constraint has been reduced to the precedence constraint from the j^{th} task to the i^{th} task ($\tau_j \prec \tau_i$).

In the above MILP, the objective function (1a) and Constraints (1b) enforce the minimization of the makespan. Constraint (1c) applies the precedence constraints and includes the time that each robot assigned to both tasks τ_i and τ_j need to transition from τ_i to τ_j. Constraints (1d) and (1e) implement the mutex constraints as big-M constraints (in both constraints, M is a large positive value): depending on the value of the mutex indicator variable δ_{ij}, only one of the two constraints can be active at any time. This disjunction makes this a Disjunctive Temporal Problem which is an NP-Hard problem [23]. Once each of the δ_{ij}'s is set then the problem becomes a Linear Program (LP) and can be solved in polynomial time. Constraints (1f) act to ensure that all robots assigned to a task can reach it from their individual initial positions before the task is scheduled to start.

The solution to the HCS problem is the set of task orderings $\rho = \{\delta_{ij} \mid (i,j) \in \mathcal{M}\}$ and the minimized makespan C. The Heterogeneous Coalition Scheduling problem also appears as a submodule of the Simultaneous Task Allocation and Planning with Spatiotemporal Constraints (STAP-STC) problem introduced in [12].

B. Heterogeneous Coalition Scheduling with Temporal Uncertainty Problem

In the Heterogeneous Coalition Scheduling with Temporal Uncertainty (HCSTU) problem, differently from its deterministic version, each of the task durations d_i, task transitions ϕ_{ij}, and initial task transition ϕ_i are represented as random variables instead of constants. We formally define a *scenario* q as a tuple $q = \langle \mathcal{D}^q,\ \Phi^q_{init},\ \Phi^q \rangle$, where $\mathcal{D}^q = \{d^q_i \mid i \in \mathcal{I}\}$ is set of task durations, $\Phi^q_{init} = \{\phi^q_i \mid i \in \mathcal{I}\}$ is a set of initial task transition durations, and $\Phi^q = \{\phi^q_{ij} \mid (i,j) \in \mathcal{P}\} \cup \{\phi^q_{ij} \mid (i,j) \in \mathcal{M}\} \cup \{\phi^q_{ji} \mid (i,j) \in \mathcal{M}\}$ is a set of task transition durations. We also define S as the space of all possible scenarios q that can sampled from the distribution over durations given by the underlying stochastic scheduling problem, $q \sim P(S)$. An immediate consequence of these definitions is that, whereas the makespan can be used to evaluate the quality of a schedule in the deterministic version of the problem, when temporal uncertainty is involved the makespan itself becomes a random variable.

Similarly to existing work [10, 6], we formulate a chance-constrained optimization problem to find a task ordering ρ that minimizes the α-robust makespan C_α such that

$$P(C_\rho > C_\alpha) \leq \alpha, \tag{2}$$

where $P(C_\rho > C_\alpha)$ denotes the probability that executing the task ordering ρ in a random scenario $q \in S$ will take longer than C_α, and α is a risk tolerance parameter provided by the user. For a given scenario q and fixed task ordering ρ, the makespan is determined by the MILP (1) by replacing all δ_{ij}'s with constants, which reduces the model to an LP.

IV. Approach

Directly optimizing C_α subject to (2) would require solving a chance-constrainted problem with probabilistic task durations and probabilistic task transition durations that create joint probabilistic constraints. This is impractical because (a) the probabilistic constraints are extremely hard to compute, and (b) the feasible region defined by the probabilistic constraints are not convex, making even checking feasibility difficult [24, 25]. For this reason, we focus our attention on sampling-based methods. We first present the Sample Average Approximation (SAA) representation of the HCSTU problem

along with its limitations. We then present our approach, Coalition Scheduling with Heuristic Sample Selection and Risk Guarantee (CS-HSSRG), which builds on elements from SAA but makes improvements for both efficiency and provides a theoretical guarantee on the risk tolerance.

A. Sample Average Approximation

The SAA representation of the HCSTU problem is given by the following MILP:

$$
\begin{aligned}
min\ & C_\alpha && && \text{(3a)}\\
s.t.\ & C^q \geq c_i^q && \forall\, i \in \mathcal{I},\ q \in \mathcal{Q} && \text{(3b)}\\
& s_j^q \geq c_i^q + \phi_{ij}^q && \forall\, (i,j) \in \mathcal{P},\ q \in \mathcal{Q} && \text{(3c)}\\
& s_j^q \geq c_i^q + \phi_{ij}^q - M(1-\delta_{ij}) && \forall\, (i,j) \in \mathcal{M},\ q \in \mathcal{Q} && \text{(3d)}\\
& s_i^q \geq c_j^q + \phi_{ji}^q - M\delta_{ij} && \forall\, (i,j) \in \mathcal{M},\ q \in \mathcal{Q} && \text{(3e)}\\
& s_i^q \geq \phi_i^q && \forall\, i \in \mathcal{I},\ q \in \mathcal{Q} && \text{(3f)}\\
& \delta_{ij} \in \{0,1\} && \forall\, (i,j) \in \mathcal{M} && \text{(3g)}\\
& C_\alpha \geq C^q - My^q && \forall\, q \in \mathcal{Q} && \text{(3h)}\\
& \alpha Q \geq \sum_{q=1}^{Q} y^q && && \text{(3i)}\\
& y^q \in \{0,1\} && \forall\, q \in \mathcal{Q} && \text{(3j)}
\end{aligned}
$$

For SAA, a scenario generator creates Q scenarios by randomly sampling from $P(S)$. Constraints (3a)-(3g) look very similar to those in the Deterministic MILP (1), but there is a set of these equations for each of the Q scenarios where a superscript of q represents that variable for the q^{th} scenario. In addition, $\mathcal{Q} = \{1, \ldots, Q\}$ and y^q is a boolean indicator variable that denotes whether or not the q^{th} scenario is used when computing the α-robust makespan C_α. Constraints (3h)-(3j) model risk tolerance by ignoring at most a fraction α of the scenarios: thanks to the big-M constraint (3h), whenever y^q is set to 1, C_α does not have C^q as lower bound.

An advantage of this approach is that it does not depend on the form of the distribution $P(S)$, or even require an explicit definition of $P(S)$, given a scenario generator. However, Constraints (3h)-(3j) add another combinatoric factor when compared with the Deterministic MILP (1), as $\binom{Q}{(1-\alpha)Q}$ combinations of scenarios have to be considered when determining the set of scenarios to ignore. This makes the approach inefficient for solving complex problems when large numbers of scenarios are needed to approximate the distributions in the problem. Also, when a scheduling algorithm is used as part of a higher level framework such as GRSTAPS it is typically run numerous times. This means the inefficiency of using SAA would be exaggerated, reducing the size of problems that the higher level framework could solve. Finally, there is no guarantee that the makespan produced by SAA is truly α-robust as specified in (2), because there is no guarantee that the Q scenarios are representative.

B. Coalition Scheduling with Heuristic Sample Selection and Risk Guarantee

Building on the SAA formulation, we now present the Coalition Scheduling with Heuristic Sample Selection and Risk Guarantee (CS-HSSRG) approach. This approach decomposes the SAA formulation into a heuristic for selecting samples which are likely to be low-cost (and therefore included in the alpha-robust makespan calculation), and a simplified MILP that we call S-SAA. The use of this heuristic removes one of the combinatoric factors and a set of boolean decision variables from the MILP that our approach needs to solve. As a result, this simplified MILP has the same complexity as the Deterministic MILP (1) and is more efficient to solve than the SAA MILP (3). Then, we use a hypothesis testing method, Wald's Sequential Probability Ratio Test (SPRT) [11], to enforce that the risk tolerance as defined in (2) is respected and that the candidate solution is truly α-robust.

Algorithm 1: Coalition Scheduling with Heuristic Sample Selection and Risk Guarantee

Input: Risk tolerance α, # of scenarios to be used in S-SAA MILP (5) η, percentage to increment the makespan δ
Output: α-robust makespan C_α, a set of task orderings ρ

1 $F \leftarrow$ {A large number of scenario samples}
// Each scenario in F is labeled with a heuristic value
2 $L \leftarrow$ {label(q) **for** $q \in F$}
3 $H \leftarrow$ The first $\lfloor (1-\alpha)|F| \rfloor$ scenarios of F sorted in ascending order of labels in L
4 $U \leftarrow$ {$\eta - 1$ random scenarios taken from H and the scenario in H with the largest label}
5 $C_\alpha,\ \rho \leftarrow$ Solve S-SAA MILP (5) with the scenarios in U
6 **while** *true* **do**
// Run the Sequential Probability Ratio Test
7 **if** $sprt(C_\alpha,\ \rho,\ \alpha)$ = *success* **then**
8 **return** $C_\alpha,\ \rho$
9 **else**
10 $C_\alpha \leftarrow C_\alpha * (1+\delta)$

The high-level pseudocode for this approach is shown in Algorithm 1. We first generate a large number of scenarios F by sampling from $P(S)$ (Line 1). We then label each scenario in F with a heuristic value that is representative of the average makespan for that specific scenario across the difference possible mutex reductions (Line 2). We use a domain independent heuristic where we compute a weighted summation of the time needed to execute each of the tasks and transitions for that specific scenario as shown in Eq. (4) below, where ψ_1-ψ_4 are weights:

$$
\begin{aligned}
label(q) = \psi_1 \sum_{i \in \mathcal{I}} d_i^q + \psi_2 \sum_{i \in \mathcal{I}} \phi_i^q \\
+ \psi_3 \sum_{(i,j) \in \mathcal{P}} \phi_{ij} + \psi_4 \sum_{(i,j) \in \mathcal{M}} (\phi_{ij}^q + \phi_{ji}^q).
\end{aligned}
\tag{4}
$$

The weights ψ_1-ψ_4 in Eq. (4) are necessary because the contributions of the task and transition durations to the makespan could be in general very different. We then sort the scenarios in F in ascending order of label, and put the first $\lfloor (1-\alpha)|F| \rfloor$ scenarios in a separate set called H (Line 3). We select η scenarios from H (Line 4), where $\eta - 1$ of the scenarios are randomly selected and the final scenario selected is the scenario with the largest heuristic value in H. Lines 2 - 4 approximate what Constraints (3h)-(3j) do in the Sample Average Approximation MILP. This allows us to use the η scenarios in a simplified version of the Sample Average

Approximation MILP (S-SAA MILP) shown below. Unlike SAA, where increasing Q increases the number of boolean decision variables, in S-SAA increasing η only adds more constants and linear constraints. This causes it to have less of an impact on the efficiency of solving the S-SAA formulation.

$$
\begin{aligned}
&min\ C_\alpha && && \text{(5a)}\\
&s.t.\ C_\alpha \geq c_i^q && \forall\, i \in \mathcal{I},\ q \in \mathcal{Q} && \text{(5b)}\\
&\quad s_j^q \geq c_i^q + \phi_{ij}^q && \forall\, (i,j) \in \mathcal{P},\ q \in \mathcal{Q} && \text{(5c)}\\
&\quad s_j^q \geq c_i^q + \phi_{ij}^q - M(1-\delta_{ij}) && \forall\, (i,j) \in \mathcal{M},\ q \in \mathcal{Q} && \text{(5d)}\\
&\quad s_i^q \geq c_j^q + \phi_{ji}^q - M\delta_{ij} && \forall\, (i,j) \in \mathcal{M},\ q \in \mathcal{Q} && \text{(5e)}\\
&\quad s_i^q \geq \phi_i^q && \forall\, i \in \mathcal{I},\ q \in \mathcal{Q} && \text{(5f)}\\
&\quad \delta_{ij} \in \{0,1\} && \forall\, (i,j) \in \mathcal{M} && \text{(5g)}
\end{aligned}
$$

The S-SAA MILP (5) selects a set of task orderings ρ that minimizes the makespan C_α for all η scenarios. Given a feasible solution of (5), however, there is no theoretical guarantee that the risk of a real scenario taking longer to execute than C_α is less than the risk tolerance α, because there is no guarantee that the η selected scenarios are representative (as in the SAA approach). As such, it is not known if the produced makespan is truly α-robust yet.

We use the SPRT [11] to test whether the produced makespan is actually α-robust. If the test passes, we can guarantee with high probability that the makespan produced is α-robust and the solution is returned by the algorithm (Lines 7 and 8). If the test fails, then the makespan is increased (Line 10) and the test is tried again until we have a makespan that is guaranteed to be α-robust. We choose to use the SPRT over other possible hypothesis testing algorithms because, as a sequential hypothesis test, the samples needed to confirm or reject the hypothesis do not need to be drawn in advance; instead, they are drawn on-demand until enough evidence to confirm or reject the hypothesis is collected. Details on the SPRT and its implementation are given below.

C. Sequential Probability Ratio Test

The Sequential Probability Ratio Test (SPRT) [11] is a sequential hypothesis test. As such, computations needed to confirm or reject the hypothesis can be lazily applied to samples as needed instead of drawing them all in advance. We consider the binomial case, where a sample can be classified into two categories: 0 and 1. Let p be the probability that a sample belongs to category 0. We deal with the problem of testing the hypothesis that p does not exceed a given value p' against the alternative hypothesis that $p > p'$. A SPRT for the binomial case can be defined by specifying a null hypothesis $H_0 \stackrel{\text{def}}{=} p = p_0$, with $p_0 < p'$, and an alternative hypothesis $H_1 \stackrel{\text{def}}{=} p = p_1$, with $p_1 > p'$. The small interval (p_0, p_1) is called the *indifference region* — the SPRT requires that we are okay with accepting either hypothesis when the true value of p is in the indifference region. Define the function $f(q)$:

$$
f(q) = \begin{cases} 1, & \textit{the sample } q \textit{ belongs to category 1;} \\ 0, & \textit{otherwise.} \end{cases} \tag{6}
$$

Given the desired type-I and type-II error rates θ[1] and β, prior to the test the following values can be calculated:

$$\lambda = \log \frac{p_1}{p_0} - \log \frac{1-p_1}{1-p_0} \tag{7}$$

$$\zeta = \log \frac{1-p_0}{1-p_1} \tag{8}$$

$$\gamma = \frac{\zeta}{\lambda} \tag{9}$$

$$l_0 = \log \frac{\beta}{1-\theta} \tag{10}$$

$$l_1 = \log \frac{1-\beta}{\theta} \tag{11}$$

The test is given as follows. At the k^{th} observation, calculate the two quantities

$$a_0 = \frac{l_0}{\lambda} + k\gamma \tag{12}$$

and

$$a_1 = \frac{l_1}{\lambda} + k\gamma. \tag{13}$$

Let $k_\top$ denote the number of observations where $f(\cdot) = 1$ in the first k samples inspected. For each time a_0 and a_1 are calculated, there are three possible outcomes:

1) accept H_0 if $k_\top \leq a_0$,
2) accept H_1 if $k_\top \geq a_1$,
3) or more observations are needed if $a_0 < k_\top < a_1$.

The third outcome means that the samples drawn so far do not provide enough evidence either to accept or reject the null hypothesis H_0. More samples need to be drawn, a_0 and a_1 need to be recomputed before checking the conditions again.

For our approach, the SPRT is given a risk tolerance parameter α, a reference makespan C_α that we are testing, and a set of task orderings ρ. We want to decide whether the risk that the time needed to execute the task ordering ρ in a random scenario exceeding C_α is less than α. Let $C(q,\rho)$ denote the makespan for scenario q when using task ordering ρ, and let $p = P(C_\rho > C_\alpha)$ denote the probability that executing the task ordering ρ in a random scenario will take longer than C_α. Therefore, we can set $p_0 = \alpha - \epsilon$, $p_1 = \alpha + \epsilon$, where ϵ is a tolerance parameter that defines the indifference region. Type-I errors occur when C_α is mistakenly rejected as not α-robust when it truly is, while type-II errors occur when C_α is mistakenly accepted as α-robust when it truly is not.

Define the Bernoulli random variable $Z_{C_\alpha,\rho}$ as follows:

$$Z_{C_\alpha,\rho} \sim Bernoulli(P(C_\rho > C_\alpha)). \tag{14}$$

We cannot draw samples from $Z_{C_\alpha,\rho}$ directly because the probability $P(C_\rho > C_\alpha)$ is unknown. Instead they are drawn from the nondeterministic function f:

$$f(C_\alpha, \rho, q) = \begin{cases} 1, & C(q,\rho) > C_\alpha; \\ 0, & otherwise. \end{cases} \tag{15}$$

[1]Wald's paper [11] uses α as the type-I error rate. We use θ to represent the type-I error rate here due to α being used for the α-robust makespan.

The nondeterminism comes from the task and transition durations being random variables themselves. Function f is our generative probabilistic model for the random variable $Z_{C_\alpha,\rho}$. Drawing samples from this probabilistic model requires drawing sample scenarios from the underlying probabilistic distribution describing the world through the previously mentioned scenario generator and then processing each one by computing $C(q,\rho)$. Separate scenarios can be processed asynchronously and in parallel to increase the speed of running the test. Furthermore, when C_α is increased and the SPRT is run again (Algorithm 1 Line 10) then previously computed $C(q,\rho)$ can be memoized and used again without the need for recomputation.

Algorithm 2: Testing hyp. $\Pr(f(C_\alpha,\rho,q)=1) \leq \alpha$

Input: Reference makespan C_α, a set of task orderings ρ, risk tolerance α, type-I error rate θ, type-II error rate β, maximum number of scenarios Q_{max}
Output: Test result

1 $\lambda, \zeta, \gamma, l_0, l_1 \leftarrow$ compute using Equations (7)-(11)
// Maximum number of scenarios with makespan $> C_\alpha$ for H_0 to be accepted with Q_{max}
2 $a_{Q_{max}} \leftarrow \frac{l_0}{\lambda} + Q_{max}\gamma$
// Total number of scenarios processed
3 $K \leftarrow 0$
// Number of scenarios whose makespan is $> C_\alpha$
4 $K_\top \leftarrow 0$
5 **while** *true* **do**
  // # of scenarios and # of scenarios whose makespan is $> C_\alpha$ since last check
6 $k,\ k_\top \leftarrow f(C_\alpha,\rho)$
7 **if** $k=0$ **then**
    // No more scenarios. Increase C_α and run test again.
8   **return** *false*
9 $K \leftarrow K+k$
10 $K_\top \leftarrow K_\top + k_\top$
11 $a_0 \leftarrow$ compute using Equation (12) for K scenarios
12 **if** $K_\top \leq a_0$ **then**
    // Accept H_0 (Accept C_α)
13   **return** *true*
14 $a_1 \leftarrow$ compute using Equation (13) for K scenarios
15 **if** $K_\top \geq a_1$ **then**
    // Accept H_1 (Reject C_α))
16   **return** *false*
17 **if** $K_\top \geq a_{Q_{max}}$ **then**
    // Unable to accept H_0 even if all remaining scenarios have a makespan $\leq C_\alpha$
18   **return** *false*

Algorithm 2 shows the pseudocode of the hypothesis test described above. Typically, the SPRT implementation either runs indefinitely before reaching a conclusion or returns a default value when all sampling resources have been used up. For our implementation, the user provides a maximum number of scenarios Q_{max} to be pre-generated and used by the test. As the guarantee on type-II error rates is desired, the algorithm terminates early if there would not be enough evidence to support accepting H_0 even when all remaining scenarios had a makespan $\leq C_\alpha$.

V. Evaluation

We empirically evaluated our approach using three sets of experiments in a simulated emergency response domain used in prior work [26, 27, 28, 29, 12]. Our first experiment examines the impact of risk tolerance, and consists of an ablation study to demonstrate that we can vary the risk tolerance and achieve plans that meet the required risk tolerance. Our second experiment is a comparison of other scheduling approaches, to show that we outperform state-of-the-art approaches in terms of mission success and computation time, even while meeting the risk tolerance. Our third experiment is to show that our approach to risk-tolerant planning can be incorporated within a heterogeneous multi-robot coordination planner, to generate coordination plans that are robust to temporal uncertainty.

In all three experiments, we use a planning domain composed of a diverse set of robots with different speeds operating in an urban environment named Polypixel (see Fig. 1). The robots need to work together to rescue wounded survivors, deliver medicine to hospitals, put out fires, and rebuild damaged infrastructure. All SPRTs were executed with type-I and type-II error rates both set to 0.05 and $\epsilon = 0.01$. Also, in all experiments, we set $|F| = 500$, $Q_{max} = 500$, $\eta = 50$ (these values were empirically determined during preliminary experiments). All approaches ran on a desktop with an AMD 3970X CPU, an A6000 GPU, and 16 GB of RAM.

For the first two experiments, we generated a set of 100 HCSTU problems from this domain by randomly sampling the number of robots, survivors, fires, and damaged buildings. Each problem had between 5-15 robots, and 10-30 tasks. We also randomized the locations of the survivors, fires, damaged buildings, hospitals, and the robots' initial location. The specific set of tasks and the allocation of robots to tasks were predetermined for each of these problems.

The first 50 problem instances simulated the possibilities of delays which directly caused temporal uncertainty and we denote this domain as DELAYS. The duration for each task was modeled as $d_i = \hat{d}_i + \mathcal{U}_{[0,300)}$ where $\hat{d}_i$ was the deterministic task duration for τ_i. The durations for transitions were computed from the length of the map edges the robot/coalition must traverse and the speed of the robot/coalition (the speed of a coalition is the speed of the slowest robot in the coalition). When traversing a map edge each robot/coalition had a 5% chance of having a $\mathcal{U}_{[0,60)}$ delay. Note that the task and transition durations were multi-modal since we applied delays to both the task durations and the edge traversal durations.

The second 50 problem instances simulated the possibility of map edges being blocked in the polypixel environment (e.g from debris) which made the traversability of the environment uncertain. This in turn indirectly created temporal uncertainty as it was unknown what was the best route between two locations on the map which made the transition durations uncertain. For each problem, each map edge had a 10% chance of being blocked. As such the resulting problem represented a variant of the Canadian Traveler Problem and we denote this domain as CTP [30]. For these problems, CS-HSSRG used the optimistic rollout policy from [31] to compute $C(q,\rho)$ for each scenario q in the SPRT. Similarly to the first 50 problem instances, the distributions for task and transition durations were multi-modal.

For the third experiment, we generated a set of 50 STAP-STC problems from the simulated emergency response domain. The problems were randomized similarly to the problems for the first two experiments, however the specific set of tasks and allocation of robots to tasks were not predetermined for these problem instances. The first 25 problem instances used the same model for unexpected delays as described above for the DELAYS domain. The second 25 problem instances used the same model for the possibility of edge blockages as described above for the CTP domain.

A. Impact of the risk tolerance

The first experiment involved an ablation study to investigate the influence of the risk tolerance α on our approach. For this experiment, we examine our algorithm with different risk tolerance levels ($\alpha \in \{0.1, 0.15, 0.2, 0.25\}$) on the 100 problem instances described previously. In Table 1, we present the max proportion of failure $\max(\bar{p}(C_\rho > C_\alpha))$ and average α-robust makespan C_α. For the proportion of failure $\bar{p}(C_\rho > C_\alpha)$, we generated 10,000 random scenarios and then computed the makespan for each scenario when executed with the solution task order ρ. The proportion of failure is the proportion of those scenarios whose makespan is larger than the solution C_α and approximates the true risk $P(C_\rho > C_\alpha)$. The max proportion of failure $\max(\bar{p}(C_\rho > C_\alpha))$ is the maximum $\bar{p}(C_\rho > C_\alpha)$ over the 100 problem instances.

Table 1: Results for different risk levels

	DELAYS		CTP	
α	$\max(\bar{p}(C_\rho > C_\alpha))$	C_α	$\max(\bar{p}(C_\rho > C_\alpha))$	C_α
0.1	0.091	3643.33	0.098	4023.63
0.15	0.136	3528.84	0.122	3881.42
0.2	0.154	3303.83	0.188	3710.10
0.25	0.224	3122.35	0.238	3510.83

In Table 1, we observe that as the risk tolerance α increases, the average α-robust makespan tends to decrease. This makes sense as the looser risk tolerance allows CS-HSSRG to ignore more slower scenarios at the expense of the higher risk of not providing an upper bound. We also observe that the max proportion of failure $\max(\bar{p}(C_\rho > C_\alpha))$ tends to be close to α, demonstrating that CS-HSSRG does not result in overly conservative makespans despite providing a theoretical guarantee on its α-robust makespan.

B. Comparison with other Scheduling Approaches

For our second experiment, we fixed $\alpha = 0.1$ and benchmarked our approach on the 100 problems described previously against two state-of-the-art RCPSPU approaches in SORU [6] and the Conditional Value-At-Risk Branch-and-Bound framework from [9] (CVAR-BNB), as well as the solving the Sample Average Approximation (SAA) formulation (3) directly. To generalize the problem to the RCPSPU for SORU and CVAR-BNB, we represented the individual robots as resources and each task or transition duration is represented as an activity. Each approach was given a 60s timeout for each problem instance. If an approach hit the timeout then the best feasible approach it generated was used.

We compared all four approaches on several metrics including the percentage of solutions where a temporal constraint was violated, the percentage of solutions where $\bar{p}(C_\rho > C_\alpha) > \alpha$, the percentage of successful solutions ($\bar{p}(C_\rho > C_\alpha) \leq \alpha$), the average computation time, the percentage of the problems where the approach hit the 60s timeout, the average $\bar{p}(C_\rho > C_\alpha)$, and the average C_α. The average C_α was only computed over problems in which all four approaches computed solutions with $\bar{p}(C_\rho > C_\alpha) \leq \alpha$.

1) SAA: As can be seen in Table 2, SAA did not violate any temporal constraints, but generated solutions that violated the risk tolerance 40% of the time for the DELAYS problems and 68% of the time for the CTP problems. It also had an average proportion of failure $\bar{p}(C_\rho > C_\alpha)$ of 0.097 and 0.112 respectively, which were either close or over the desired risk tolerance. These risk tolerance violations were likely because SAA does not get to choose representative samples like CS-HSSRG and additionally does not use statistical testing to guarantee its "α-robust makespan" is actually α-robust. On the other hand, CS-HSSRG used its heuristic sample selection and the SPRT to successfully generate risk tolerant solutions on 100% of the problems for both domains. SAA did on average compute a smaller C_α than any of the other approaches including CS-HSSRG. On average for these problems, its C_α was 90% of the C_α generated by CS-HSSRG. As SAA frequently violated the risk tolerance, it is likely that when it doesn't violate the risk tolerance the C_α generated has a less conservative risk resulting in the lower average C_α. When computing a solution, SAA on average took 41.4x the time to compute solution as CS-HSSRG did. Additionally, it hit the timeout on 13% of problems in total.

2) SORU: SORU was the only approach that generated solutions which violated temporal constraints. This is caused by SORU using soft temporal constraints. It did not have any risk tolerance violations, which is caused by its extremely conservative solutions with an average risk of 0.003 and 0.011 respectively for the two domains. This extremely conservative risk resulted in it having the highest average C_α. The average C_α generated by SORU was 120% of the C_α generated by CS-HSSRG. SORU also took the longest of the four approaches on average and took on average 60.5x as long to compute a solution as CS-HSSRG. Additionally, it hit the 60s timeout on 73% of all the problems.

3) CVAR-BNB: Similar to SAA, CVAR-BNB did not violate any temporal constraints, but violated the risk tolerance on some of the problems (28% and 50% respectively). The risk tolerance violations are likely because CVAR-BNB does not get to choose representative samples like CS-HSSRG and additionally does not use statistical testing to guarantee its "α-robust makespan" is actually α-robust. CVAR-BNB on average generated C_α that was 97% of the C_α generated by CS-HSSRG, however on average it took 10.1x as long to compute its solution. Also, while it was quicker than both SAA and SORU on average, it did hit the 60s timeout on 2% of the total problems.

Table 2: Summary of comparison results ($\alpha = 0.1$)

	DELAYS				CTP			
	Ours	SAA	SORU [6]	CVAR-BNB [9]	Ours	SAA	SORU	CVAR-BNB
TC Failure (%)	**0**	**0**	8[1]	**0**	**0**	**0**	12[1]	**0**
$\bar{p}(C_\rho > C_\alpha) > \alpha$ (%)	**0**	40	**0**	28	**0**	68	**0**	50
Success (%)	**100**	60	92	72	**100**	32	88	50
Avg. $\bar{p}(C_\rho > C_\alpha)$	0.079 ± 0.020	0.097 ± 0.053	0.003 ± 0.009	0.088 ± 0.043	0.071 ± 0.016	0.112 ± 0.029	0.011 ± 0.010	0.096 ± 0.045
Avg. C_α [2]	3794.71 ± 1305.60	**3376.19** ± **1953.00**	4554.90 ± 1436.70	3679.04 ± 2390.73	4365.24 ± 1620.32	**4268.57** ± **1800.43**	5813.46 ± 2132.32	4316.96 ± 2532.38
Avg Comp. Time (s)	**0.822** ± 0.426	34.057 ± 24.288	49.760 ± 17.154	8.313 ± 1.913	**1.014** ± 0.34	42.02 ± 19.88	57.64 ± 9.41	10.93 ± 8.31
Timeout (%) [3]	**0**	10	64	2	**0**	16	82	2

[1]SORU uses hard resource constraints and soft temporal constraints, which sometimes allow it to violate the temporal constraints.
[2]Computed using trials for which all four approaches' $\bar{p}(C_\rho > C_\alpha) \leq \alpha$
[3]Upon hitting the 60s timeout, a feasible solution was retrieved if one had been found by the approach.

4) Ours: In comparison to the other three approaches, CS-HSSRG was the only approach to never have a temporal constraint violation or a risk tolerance violation and was the only approach to never hit the 60s timeout. Additionally, it on average generated C_α that were within 10% of SAA, 3% of CVAR-BNB, and were 20% faster than SORU.

Furthermore, CS-HSSRG is an order of magnitude faster than all other approaches (on average 41.4x faster than SAA, 60.5x faster than SORU, and 10.1x faster than CVAR-BNB), demonstrating that it is significantly more efficient than the state-of-the-art without sacrificing solution quality.

C. Incorporating CS-HSSRG within Heterogeneous Multi-agent Coordination

For our third experiment, we demonstrated how CS-HSSRG's ability to reason about uncertainty and risk can be used within a heterogeneous multi-robot coordination planner to make decisions about what tasks to select and who to allocate to those tasks. To show the applicability of CS-HSSRG in this context, we used GRSTAPS [12], which is a state-of-the-art heterogeneous, multi-robot coordination planner, and we used CS-HSSRG as the scheduling layer of GRSTAPS. Specifically, we used CS-HSSRG with $\alpha = 0.1$ to compute an α-robust makespan of the schedule, and replaced the normal makespan in the Normalized Schedule Quality (NSQ) heuristic and as the path cost for the task planning layer. We compared this variant of GRSTAPS which we will called GRSTAPS^{TU} against the original GRSTAPS which we called GRSTAPS^{DET}. For each problem, GRSTAPS^{DET} was provided a determinized version of the problem where $d_i = \mathbb{E}[d_i]$, $\phi_{ij} = \mathbb{E}[\phi_{ij}]$, and $\phi_i = \mathbb{E}[\phi_i]$. Again, the goal was to show that CS-HSSRG could allow heterogeneous, multi-robot coordination planning that is robust to temporal uncertainty.

Table 3: Results for $\bar{p}(C_\rho > C_\alpha)$

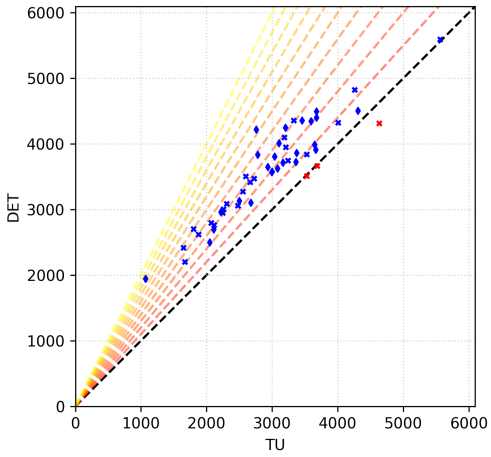
	DELAYS	CTP
DET	0.5275 ± 0.3271	0.7805 ± 0.2175
TU	0.0858 ± 0.0061	0.0864 ± 0.0063

We measured the quality of the solution task orderings ρ produced by each approach. For each problem instance, we generated 10,000 random scenarios and then computed the makespan for each scenario when executed with the solution task ordering. We then took the average makespan over the 10,000 scenarios. The average makespans can be seen in Fig. 2a. In Fig. 2b, we show the sum of the average makespans and the time needed to compute the solution, since the latter could take a significant part of the actual mission duration in a real-world setting. Additionally, we use the makespans computed for these 10,000 realizations to compute $\bar{p}(C_\rho > C_\alpha)$ which is shown in Table 3.

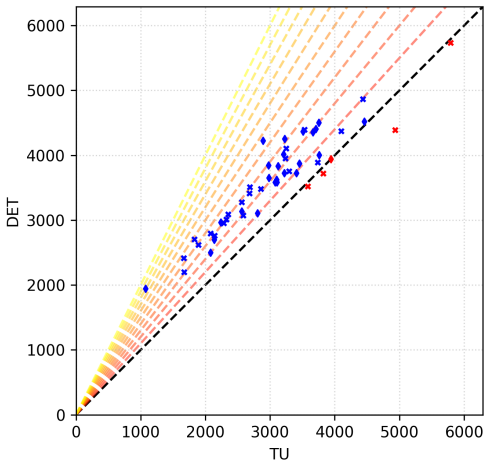
As can be seen, by reasoning about uncertainty, GRSTAPS^{TU} was able to create better task orderings that on average result in better makespans. GRSTAPS^{DET} uses the expected values of individual distributions while attempting to create a solution that minimizes makespan, however, it does not consider anything more about the distribution such as variance or modality. This can cause it to select tasks, allocations, and task orderings that have low expected values for duration, but have a high risk of having large durations. Due to the large number of decisions it has to make (which tasks, which robots, which task orderings), there are a lot of possibilities for the individual risks of a task or transition taking longer than expected to be realized in a scenario and can result in high overall risk. This can be seen in Table 3 where GRSTAPS^{DET} has high average and standard deviation for the estimate of overall risk $\bar{p}(C_\rho > C_\alpha)$. On the other hand while CS-HSSRG does not directly consider the distributions, it indirectly considers the distributions through the sampling and sample selection method and then it directly considers the risk through the SPRT. As both approaches generated and expanded similar numbers of search nodes in both the task planning and task allocation layers of GRSTAPS (less than 0.1% difference on average), it is likely that the improvement in solution quality is a result of CS-HSSRG providing better search guidance to both the task planning and task allocation layers of GRSTAPS. This better guidance results in higher quality, low risk solutions.

VI. Conclusion

In this paper, we introduced a novel sampling-based risk-aware multi-robot scheduling algorithm named Coalition Scheduling with Heuristic Sample Selection and Risk Guarantee (CS-HSSRG). We showed how CS-HSSRG could solve for multi-robot schedules with guarantees of probability of completion in the presence of temporal uncertainties, providing significant performance gains for these problems relative to existing methods without making any assumptions about the



(a) Average makespan when using the solution task ordering ρ.



(b) Sum of the average makespan when using the solution task ordering ρ and computation time.

Fig. 2: The blue (red) markers are problem instances where GRSTAPS^{TU} did better (worse) than GRSTAPS^{DET}. Each successive dashed colored line is the result from GRSTAPS^{DET} being 10% worse than the result from GRSTAPS^{TU}.

underlying uncertainty distribution. Additionally, we showed that our approach could be incorporated in a multi-robot coordination system to provide overall multi-robot plans that are robust to temporal uncertainty.

Nevertheless, our approach does have some limitations that must still be addressed: the user must determine the number of scenarios to be used in the S-SAA MILP η and the maximum number of scenarios used by the SPRT Q_{max}. As mentioned earlier, increasing η or Q_{max} increases the quality of the solution and the likelihood that the risk tolerance can be achieved when the SPRT is run for the first time in Algorithm 1. However, increasing these parameters also increases the computation time needed to solve the problem. In our experiments, we empirically determined η and Q_{max} through some preliminary experiments, measuring the number of SPRT failures. Future work may focus on fast learning-based methods able to automatically select the scenarios to be used in the S-SAA MILP, varying η based on the confidence of the prediction.

Acknowledgments

This material is based upon work supported under the DCIST CRA by the Army Research Laboratory under Cooperative Agreement Number W911NF-17-2-0181.

References

[1] J. Liu and R. K. Williams, "Coupled temporal and spatial environment monitoring for multi-agent teams in precision farming," *IEEE Conference on Control Technology and Applications*, pp. 273–278, 2020.

[2] C. J. McCook and J. M. Esposito, "Flocking for heterogeneous robot swarms: A military convoy scenario," in *Proceedings of the Annual Southeastern Symposium on System Theory*, 2007, pp. 26–31.

[3] G. Neville, A. Messing, H. Ravichandar, S. Hutchinson, and S. Chernova, "An Interleaved Approach to Trait-Based Task Allocation and Scheduling," in *Proc. IROS*, 2021.

[4] J. Blazewicz, J. K. Lenstra, and A. H. Kan, "Scheduling subject to resource constraints: classification and complexity," *Discrete Applied Mathematics*, vol. 5, no. 1, pp. 11–24, 1983.

[5] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, pp. 61–95, 1991.

[6] P. Varakantham, N. Fu, and H. C. Lau, "A proactive sampling approach to project scheduling under uncertainty," in *Proc. AAAI*, 2016, pp. 3195–3201.

[7] J. Banfi, A. Messing, C. Kroninger, E. Stump, S. Hutchinson, and N. Roy, "Hierarchical Planning for Heterogeneous Multi-Robot Routing Problems via Learned Subteam Performance," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4464–4471, 2022.

[8] P. Lamas and E. Demeulemeester, "A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations," *Journal of Scheduling*, vol. 19, no. 4, pp. 409–428, 2016.

[9] W. Song, D. Kang, J. Zhang, and H. Xi, "Risk-aware proactive scheduling via conditional value-at-risk," in *Proc. AAAI*, 2018.

[10] N. Fu, P. Varakantham, and H. C. Lau, "Robust partial order schedules for RCPSP/max with durational uncertainty," in *Proc. ICAPS*, 2016, pp. 124–130.

[11] A. Wald, "Sequential Tests of Statistical Hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117–186, 1945.

[12] A. Messing, G. Neville, S. Chernova, S. Hutchinson, and H. Ravichandar, "GRSTAPS: Graphically Recursive Simultaneous Task Allocation, Planning, and Schedul-

ing," *International Journal of Robotics Research*, vol. 41, no. 2, pp. 232–256, 2022.

[13] E. Bischoff, F. Meyer, J. Inga, and S. Hohmann, "Multi-Robot Task Allocation and Scheduling Considering Cooperative Tasks and Precedence Constraints," in *IEEE International Conference on Systems, Man and Cybernetics*, 2020.

[14] D. Matos, P. Costa, J. Lima, and A. Valente, "Multiple Mobile Robots Scheduling Based on Simulated Annealing Algorithm," in *International Conference on Optimization, Learning Algorithms and Applications*, 2021.

[15] C. Amato, "Decision-making under uncertainty in multi-agent and multi-robot systems: Planning and learning," in *Proc. IJCAI*, 2018, pp. 5662–5666.

[16] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, 2017.

[17] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, 2004.

[18] G. Korsah, A. Stentz, and M. Dias, "A comprehensive taxonomy for multi-robot task allocation," *International Journal of Robotics Research*, vol. 32, pp. 1495–1512, 2013.

[19] Y. Zhang and L. E. Parker, "Multi-robot task scheduling," in *Proc. ICRA*, 2013.

[20] M. Saint-Guillain, T. S. Vaquero, S. A. Chien, J. Agrawal, and J. Abrahams, "Probabilistic Temporal Networks with Ordinary Distributions: Theory, Robustness and Expected Utility," *Journal of Artificial Intelligence Research*, vol. 71, pp. 1091–1136, 2021.

[21] F. Habibi, F. Barzinpour, and S. J. Sadjadi, "Resource-constrained project scheduling problem: review of past and recent developments," *Journal of Project Management*, pp. 55–88, 2018.

[22] D. Duue and J. Pan, "An Overview of Value at Risk," *Journal of Derivatives*, vol. 4, no. 3, pp. 7–49, 1997.

[23] K. Stergiou and M. Koubarakis, "Backtracking algorithms for disjunctions of temporal constraints," *Artificial Intelligence*, vol. 120, no. 1, pp. 81–117, 2000.

[24] J. Luedtke, S. Ahmed, and G. L. Nemhauser, "An integer programming approach for linear programs with probabilistic constraints," *Mathematical Programming*, vol. 122, no. 2, pp. 247–272, 4 2010.

[25] J. Brooks, E. Reed, A. Graver, and J. C. Boerkoel, "Robustness in probabilistic temporal planning," in *Proc. AAAI*, 2015.

[26] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, "RoboCup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 6, pp. 739–743, 1999.

[27] P. Bechon, M. Barbier, G. Infantes, C. Lesire, and V. Vidal, "HiPOP: Hierarchical Partial-Order Planning," *STAIRS*, pp. 51–60, 2014.

[28] A. Whitbrook, Q. Meng, and P. W. Chung, "A novel distributed scheduling algorithm for time-critical multi-agent systems," in *Proc. IROS*, vol. 2015-Decem, 2015, pp. 6451–6458.

[29] W. Zhao, Q. Meng, and P. W. Chung, "A Heuristic Distributed Task Allocation Method (PIA)," *IEEE Transactions on Cybernetics*, vol. 46, no. 4, pp. 902–915, 4 2016.

[30] D. Fried, S. E. Shimony, A. Benbassat, and C. Wenner, "Complexity of Canadian traveler problem variants," *Theoretical Computer Science*, vol. 487, pp. 1–16, 2013.

[31] P. Eyerich, T. Keller, and M. Helmert, "High-quality policies for the Canadian traveler's problem," in *Proc. AAAI*, 2010.