Modeling and Simulation for Multi-Robot Allocation and Execution of Tasks with Temporal and Precedence Constraints

Ernesto Nunes and Maria Gini[1]

Abstract— **This work leverages the power of simulation as a tool to analyze the robustness of schedules for robots in a multi-robot system, where the robots cooperate to accomplish a set of tasks. Data collected during simulations are used to compute well-known indexes that measure the risk of delay and failure in the robots' schedules. We also build a Bayesian probabilistic model from which we can compute the probability of missing tasks' deadlines. In this model, start times of tasks are treated as random variables; the relations imposed by precedence constraints define a graphical model that captures the dependencies between tasks' random variables. We run ROS-Stage simulations for up to tens of robots and tasks on an already computed schedule to generate data to perform robustness analysis.**

I. Introduction

Multi-robot systems continue to gain traction as a viable option in warehouse automation, precision agriculture, environmental monitoring, and other application areas. Simulation software systems have been critical in improving the programming of multi-robot systems. They can also provide large amounts of data that can be used to analyze the robots' operations. In this paper we use data obtained in simulation to compute risk and probabilistic analysis for stochastic task allocation and execution when tasks have temporal and precedence constraints.

Most of the research in multi-robot task allocation and execution has proposed centralized and decentralized planners that are reactive in nature. Reactive planners absorb delays caused by exogenous events by adjusting the temporal bounds on tasks in a deterministic way, and re-planning when required. However, such planners most often ignore information that can be obtained by modeling the stochastic processes that generate the exogenous events.

We propose a simulation framework that is used to generate two types of analysis: the first is a risk-based analysis of task delay and failure. The second is a probabilistic analysis based on Bayesian principles. A graphical model is used to represent the probabilistic interdependencies between robots' tasks. Interdependencies are created by precedence relations imposed by the directed acyclic precedence graph, and by the sequencing order of tasks in each robot's schedule.

Our main contribution is an analytical framework in which risk and probabilistic analysis are performed on data generated from our simulation framework. This analysis can be used by a system user to identify changes in temporal and precedence constraints that can improve the chance of task completion. We also propose a graphical model for random variables for tasks' start times. The graphical model is preliminary, work is underway to learn and test the model.

[1] Ernesto Nunes and Maria Gini are with the Department of Computer Science and Engineering, University of Minnesota.

II. Related Work

Despite the increasing use of multi-robot simulators both in academic and industrial settings, only a few works exist that address the stochastic multi-robot task allocation problem by leveraging simulations to explicitly build probabilistic models. These models could be used during decision-making to account for the uncertainty induced by the use of robots in dynamic environments.

Work exists that proposes centralized solutions for stochastic routing and scheduling problems [1], [2], [3]. Like the allocation problem herein discussed, these problems simultaneously solve an allocation, a routing and a scheduling problem. In [3] a dynamic vehicle routing problem (VRP) is studied in which demands (or tasks) with deterministic time constraints arrive randomly, and the goal is to maximize the fraction of demand met. In their work, vehicle motion is constrained and a reachability graph is used for navigation. In [2] uncertainty in task arrival is also addressed. Their work provides theoretical analysis of a number of requirements, such as bounds on the number of vehicles used and maximum number of tasks that can be missed.

A more common way to model uncertainty in AI is by using Markov Decision Processes (MDPs). In [4], [5] MDPs are modeles in which states are locations in a map with obstacles, tasks, and robots. In [5] a state is modeled as a triplet representing the previously visited state, the amount of resources left, and the time window. Solutions to the MDP search for policies that maximize a value function over the states. Dolgov et al. [6] pose the combinatorial resource scheduling problem with uncertainty as an MDP. However, none of the works directly address the problem we solve: we deal with uncertainty generated by system dynamics during robot travel, and none of these works address problems with constraints.

Uncertainty modeling has been extended to temporally constrained problems. In [7] a set bounded uncertainty model is used to express duration uncertainty of temporal events in a Simple Temporal Network (STN) [8], and STNs with uncertainty (STNU) are introduced. STNUs have been extended by modeling uncertainty as probabilities [9], [10], [11]. The former attempts to minimize the risk of temporal inconsistencies occurring, and the latter attempts to bound the probability of not meeting a schedule. However, these models cannot be directly applied to our problem since they assume that tasks' distributions are independent. In

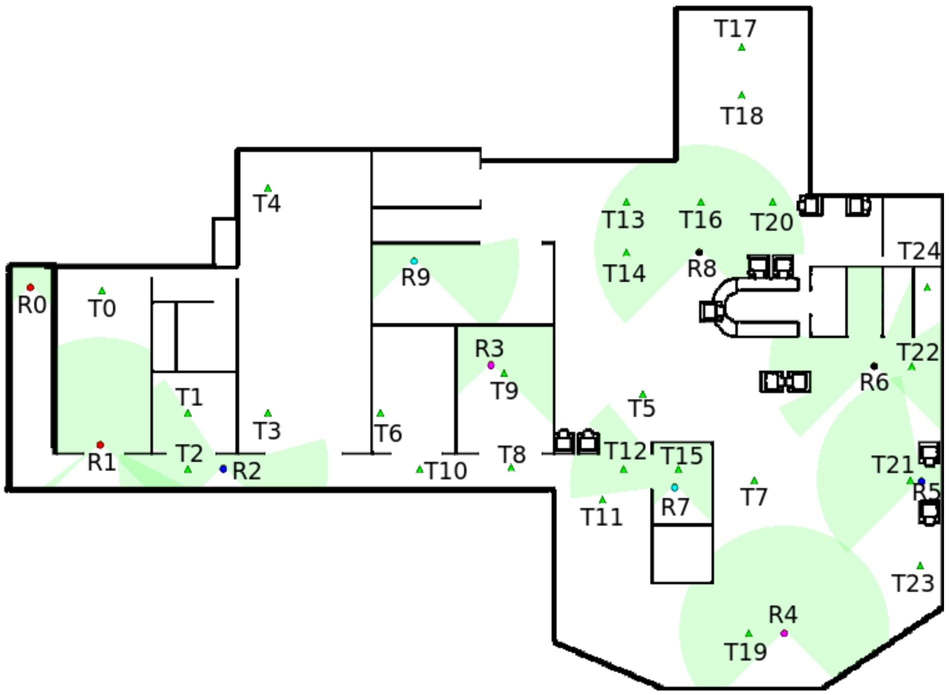

Fig. 1. ROS/Stage simulation with 10 robots (circles) and 25 tasks (triangles).

[10] probabilistic STNs (pSTNs) are introduced. Instead of requiring consistency for all probabilistic durations pSTNs require schedules for which the probability of failure can be bounded. Unfortunately, in robotics it might not be possible to find such strategies.

In [12] uncertainty is handled as a chance-constrained model. The Consensus-Based Bundle algorithm (CBBA) is proposed that solves the stochastic multi-robot task allocation problem with temporal constraints. CBBA allocates tasks with uncertainty in planning parameters to multiple robots. This work comes closest to ours, however it does not handle precedence constraints.

III. Problem Definition

We assume, as in [13], a set $\mathcal{R}$ of m robots. Each robot r_i has an initial pose, a maximum velocity, and a set of sensors. The maximum velocity is the same for all robots, but robots can travel at different speeds. We also assume a set $\mathcal{T}$ of n tasks, each with a location, an earliest start time ES_{t_j}, a latest finish time LF_{t_j}, and a duration (DU_{t_j}). Together, (ES_{t_j}, LF_{t_j}) define the bounds for the task's time window. Robots need to arrive to a task before its latest start time LS_{t_j}, which, if not specified, can be computed as $LS_{t_j} = LF_{t_j} - DU_{t_j}$. The robots have a graph representation of the environment, where the vertices are waypoints and the edges connect pairs of vertices between which there are no obstacles.

The execution framework proposed in [13] produces:

1) A multi-robot schedule, with a set of m sub-schedules $\Lambda = \{..., \Lambda_{r_i}, ...\}$ one per robot. Each schedule in Λ contains a sequence of a subset of tasks, ordered according to the time they will be performed.

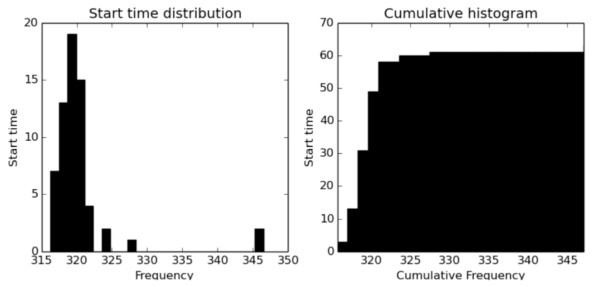

Fig. 2. Start time distribution for task T2 for 10 robots and 25 tasks.

2) the set of planned start times Δ for each task. Each Δ_{t_j} is fixed for a multi-robot schedule.

In this paper, we are interested in using a large number of simulations to compute:

1) $P(st_{t_j} \leq \Delta_{t_j})$, the probability that the start time at execution time is within the planning time;
2) $P(st_{t_j} \leq ls_{t_j})$, the probability that the start execution time is smaller than the task's latest planned start time. This is the probability that the task will not fail;
3) a discretized division of the map into areas. Finally, a distribution of the average robot traversal times per grid area.

For each task, our simulation system produces a distribution $(\mu_{st_{t_j}}, \sigma_{st_{t_j}})$ for the times in which the execution of a task starts. For the environment in Figure 1 the start time distribution for task T2 is shown in Figure 2. We use such distributions to compute measures of risk for each task, and instantiate the probabilistic model we propose, as shown later in Section V.

IV. The Simulation and Task Execution System

We implemented our prioritized Iterated Auction [14] using the Robotic Operating System (ROS) Stage plugin as described in [13]. The simulator uses a 2D representation of the robots and of the environment in which the robots operate. The system relies on ROS packages for local and global path planning (including collision avoidance), and for communication.

During the planning stage, the set Λ of robots' schedules is generated using pIA. During the execution stage, the robots execute the tasks in their schedules using the executor we proposed in [13]. During execution robots travel to tasks, execute them, and request task reassignments when they cannot perform any of their allocated tasks. Robots communicate with each other and with an auctioneer agent who keeps track of task execution across robots to ensure that precedence constraints are not violated and to reallocate tasks when reallocation is requested by a robot.

A. Robot Schedule Dispatching and Task Execution

At any time during execution a robot is either traveling to a task, executing a task, waiting to perform a task, or aborting execution. These states are directly correlated to the following execution outcomes: *succeeded* – the task was successfully executed, *aborted* – the assigned robot estimates that it cannot arrive to the task on time but there is still time to perform the task, *failed* – due to temporal constraint violation no robot in the system can do the task.

When the execution starts, a robot retrieves the first task in its schedule and travels to that task. In each ROS cycle, the robot computes the estimated start time of its next task by adding the travel time from its current location to the task's location to the already elapsed time (since the start of the traveling action). If the estimated start time is smaller than the start time computed during planning, the robot continues execution. If not, there are different cases. If the estimated new start time is smaller than the task's latest start time, the task can still be executed, but the new start time could cause inconsistencies with other tasks for which this task is a precedence constraint. Since those other tasks could have been assigned to other robots, the delayed robot needs to check with the auctioneer for any potential violation. If there are no violations, the robot assigns the estimated new start time as the task's start time and continues executing. Otherwise, the robot notifies the auctioneer that it is unable to execute the task within its temporal constraints. The auctioneer runs an auction to try to reallocate that task.

When a robot has completed the execution of a task, it marks it as succeeded, notifies the auctioneer, and proceeds to its next task. This way the auctioneer can keep track of the overall progress.

B. Auctioneer Execution Updates

As execution unfolds the auctioneer updates information about the tasks that have been completed and the tasks whose start times need to be updated due to execution delays.

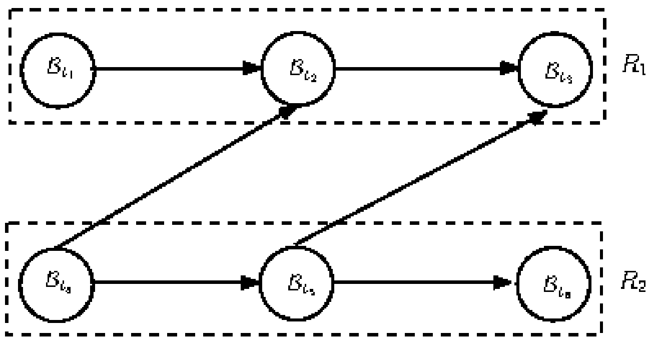

Fig. 3. Example of graphical model showing dependencies between random variables representing tasks for 6 tasks and 2 robots.

To stay updated, the auctioneer subscribes to ROS topics in which robots post their tasks' execution status. In each ROS cycle it checks if a task has been completed. If a task is marked as aborted, the auctioneer put that task up for auctions to see if any of the robots can add it to its schedule. Each robot computes its bid, as it did during planning, and sends the bid to the auctioneer, which allocates the task to the robot with the smallest bid, if any. If no robot can do the task (i.e., all robots submit ∞ as their bid value), the task and all the tasks in its induced subgraph are marked as *failed* and removed from the scheduled tasks. To keep precedence constraints consistent failed tasks are never executed.

When a robot experiences a delay it attempts to set a new value to its task start time and any task whose execution time depends on the delayed task. The auctioneer needs to check if the new start time causes inconsistencies for other robots. The auctioneer keeps a a directed acyclic graph $G_{\mathcal{P}}$ representing precedence constraints between tasks and the start and finish times of the tasks, which are updated during execution. When a robot asks the auctioneer to check for inconsistency of a potential time update, it sends the new start (and finish) times. The auctioneer proceeds by temporarily updating the finish times of all remaining tasks in the DAG. Next, it performs a topological sort on the DAG to compute a linear ordering according to the precedence constraints. The auctioneer uses the sorted graph to compute $\tilde{S}_{t_k}$ for all tasks. It then checks if $\tilde{S}_{t_k} \leq LS_{t_k}, \forall t_k$. If that is the case, the auctioneer accepts the temporal updates and send the robot an "OK" message. Otherwise, the auctioneer rejects the time updates, the robot aborts the task, and the auctioneer resets the task start time to its previous value.

V. Toward Risk and Probabilistic Models for Analysis of a Single Schedule

We expect the start times of tasks to vary across runs due to several sources of uncertainty. In this paper we focus on the delays caused by congestion (too many robots working in the same area), number and shape of fixed obstacles, and sensor and localization errors as the main sources of uncertainty.

Congestion causes the robots to reduce their speeds to avoid collisions, which in turn causes fluctuation in their travel times, and ultimately affects their arrival times to tasks. The number and shape of fixed obstacles in an environment (especially in unstructured environments) require robots to slow down to avoid collision with the obstacles, causing arrival time variations. When robots experience localization and sensor failures they might need to replan, which introduces a time overhead. In this paper we only account for transient failures such as the ones already discussed, we do not directly address permanent failures (e.g., destruction of robots).

Let $\mathcal{B}_{\mathcal{T}} = \{\mathcal{B}_{t_1}, ..., \mathcal{B}_{t_j}, ..., \mathcal{B}_{t_n}\}$ be the set of random variables for the tasks' start times, one per task. We treat $\mathcal{B}_{t_j}$ as a continuous random variable. We assume hard temporal constraints, for this reason the probability mass for each $\mathcal{B}_{t_j}$ is non-zero only within the task's time windows ($[es_{t_j}, ls_{t_j}]$).

We also consider dependencies among $\mathcal{B}_{t_j}$ variables. The structure of these dependencies is formed due to precedence constraints over the set of tasks and task sequencing in the robots schedules. The dependencies are also encoded in uncertainties of the tasks: a late arrival of a robot to a predecessor of a task might delay the arrival to that task too. We use a Bayesian network (BN) to model the dependencies (see Figure 3).

A BN is a directed acyclic graph in which random variables that share an edge are dependent. The directed edges define parent-child relationships among random variables, the same way that they describe precedence in our problem. In a BN each variable is conditionally independent of all its non-descendants given all of the variables' parents. This is useful because it allows us to model tasks in the precedence graph that are not dependent on each other (e.g tasks in different subgraphs).

In the BN each $\mathcal{B}_{t_j}$ is parameterized with a conditional probability table (CPT) that encodes the probabilities of the random variable, given its parents. In Figure 3 the CPT for $\mathcal{B}_{t_3}$ is represented as the conditional probability $p(\mathcal{B}_{t_3}|\mathcal{B}_{t_3}, \mathcal{B}_{t_5})$. If we are given an already learned BN, we can answer queries such as "what is the probability that a robot 1 will arrive to task t_3 on time, given that the robot started t_3's predecessors within their planned start times?" To answer the query we compute the following probability:

$$p(\mathcal{B}_{t_3} \leq \Delta_{t_3}|\mathcal{B}_{t_2} \leq \Delta_{t_2}, \mathcal{B}_{t_5} \leq \Delta_{t_5}) = \frac{\mathcal{P}}{\mathcal{Q}}$$

$$\mathcal{P} = p(\mathcal{B}_{t_2} \leq \Delta_{t_2}, \mathcal{B}_{t_3} \leq \Delta_{t_3}, \mathcal{B}_{t_5} \leq \Delta_{t_5}); \mathcal{Q} = p(\mathcal{B}_{t_3} \leq \Delta_{t_3})$$

The computation of $\mathcal{P}$ and $\mathcal{Q}$ is intractable, especially for a BN that encodes a large and complex precedence graph. Approximate inference could be employed using methods such as Gibbs sampling, but this would assume that the BN parameters (the CPT for each $\mathcal{B}_{t_j}$) and structure have already been learned. Fortunately in this case we would not need to learn the BN structure (which is a harder task), given that these are derived from the precedence and sequencing constraints. However, we still need to learn the BN parameters.

Training data to learn the BN parameters are obtained in simulation as historical data for tasks' start times (see Figure 2 for example). To learn the BN parameters we will use a Bayesian update approach: assign a prior probability density function to each $\mathcal{B}_{t_j}$ and use the training data (e.g Figure 2) to compute a posterior parameter distribution and the Bayes estimates. The next future step in our project is to learn the BN parameters for our data sets.

For now, we estimate probability distributions for each $\mathcal{B}_{t_j}$ following a frequency-based analysis. We run our simulation several times and collect statistics about tasks' start times, and estimate some probabilities, as we will discuss shortly.

Using the collected distributions we compute risk measures for individual tasks. This level of risk analysis gives the system user granular information about individual task's influence on the overall execution outcome. The designer can then adjust tasks' temporal constraints to improve the probability of a task being completed on time.

To measure risk we adapt two well-known risk measures, the schedule sensitivity index (SSI) and the critical delay contribution index (CDC) to our problem. Both indices have been identified as good risk measures in probabilistic project scheduling [15]. SSI measures risk as a task's variance contribution compared to the overall schedule variance. CDC measures risk as individual task's contributions the overall lateness (difference in makespan between the planned makespan and the execution makespan).

We simplify the computation of SSI and CDC as follows: the variance contribution in SSI is weighted by the number of times a robot arrived to a task after the task's planned start time (start time delay). We compute that in each simulation, and sum over all the simulations (see Equation 1). The same weights are also used to compute CDC values (see Equation 2).

When computing SSI we normalize individual task's variances by dividing them by the maximum variance over all tasks' start times. This is a departure from the original index, which used the variance of the last task's start time. This distinction is important because the variance in the last overall task's start time depends on the task's predecessors' variances. If the task has very large time window (and temporal flexibility) the time window length may attenuate or even dissipate the effects other tasks' variances have on the last task's variance.

$$SSI_{t_j} = z \cdot \sqrt{\frac{\mathrm{Var}(\vec{s}_{t_j})}{\max_{t_k \in \mathcal{T}} \mathrm{Var}(\vec{s}_{t_k})}} \tag{1}$$

$$CDC_{t_j} = z \cdot \sum_{q=1}^{\mathcal{N}} (m^{plan} - m^q) \tag{2}$$

$$z = \frac{\sum_{q=1}^{\mathcal{N}} \delta_{t_j}^q}{\mathcal{N} \sum_{t_j \in \mathcal{T}} \sum_{q=1}^{\mathcal{N}} \delta_{t_j}^q} \tag{3}$$

In Equation 1 $\delta_{t_j}^p$ is an indicator function that assumes the value of 1 if the robot assigned to execute task t_j starts the task past its planned start time Δ_{t_j}, and 0 otherwise. This value is collected in each simulation run p over all simulation runs $\mathcal{N}$. $\vec{s}_{t_j}$ is the vector of start times for task t_j, and $\vec{s}_{max}$ is the vector of tasks' start times with the maximum variance. These vectors have dimensions $\mathcal{N} \times 1$. In Equation 2 m^{plan} is the makespan robots' schedules prior to execution, and m^p is the makespan for simulation number p ($m^{plan} - m^p < 0$ indicate lateness in finishing the last overall schedule).

VI. Experiments and Preliminary Results

Given the set of schedules Λ, and other inputs, we run the simulator several times (100 times in the results herein reported) to gather statistics for tasks' start times and for the time robots spend on a grid abstraction of the map (see Figure 4).

A. *Grid Time Distribution Results*

We generate a discretization of the map by overlaying a grid over the map. We tried grids of different sizes and found that a 4×4 meter grid size yielded the best results.

Using a ROS localization package we generate timed path traces for the robots, where to each (x,y) location we attach a timestamp indicating when the robot arrived to the location. For each set of locations in the robot path that fall under the same grid we compute the time a robot spends in a grid by calculating the difference between timestamps. The differences are computed across all robot runs. The time distribution is averaged over all robots that have paths that visit the same grids. We also run extra simulations where we vary tasks' locations but keep time windows, precedence constraints, and robots' locations the same. By varying tasks' locations we create traversal time profiles for more grids on the map. The grid representation and the respective traversal times are reported in Figure 4.

The traversal time distribution (right plot in Figure 4) per grid shows that for most grids robot traversal time is under 7 minutes. Except for regions between 56-58 where robots spend nearly 10 minutes. This is partly explained by the fact that when operating in that are robots some times can not properly localize themselves and they get stuck in the same place for the rest of the simulation.

B. *Risk and Probability Results*

We also report results computed for tasks' risks and probabilities (see Figures 5, 6, 7, 8). The risks are computed according to (1) and (2). The two risks highlight different information about tasks' stochastic start times: SSI highlights the importance of tasks' variances, while CDC highlights the importance of delays of individual tasks. In our experiments the tasks with highest SSI risk are located in the grids with largest traversal times. This supports the hypothesis that tasks with high variance in their travel times, and consequently start times, are more risky. However SSI by itself might be enough because it does not measure effects of a tasks' variance on the overall risk of a schedule. For that we need the CDC risk index.

The CDC risk index shows that more tasks are at risk. Part of that is explained by the fact that CDC combines the individual (task level) and overall (schedule-level) delays

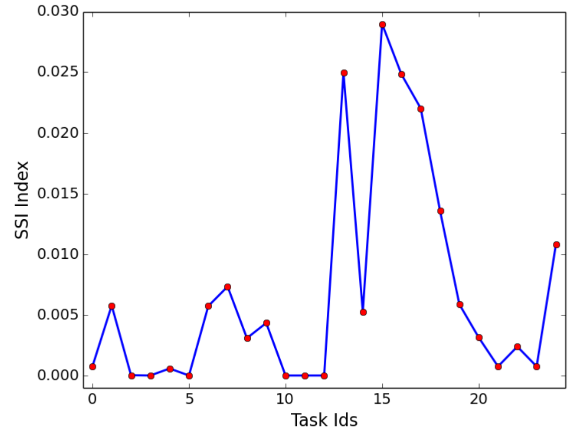

Fig. 5. Schedule sensitivity index for 25 tasks executed by 10 robots.

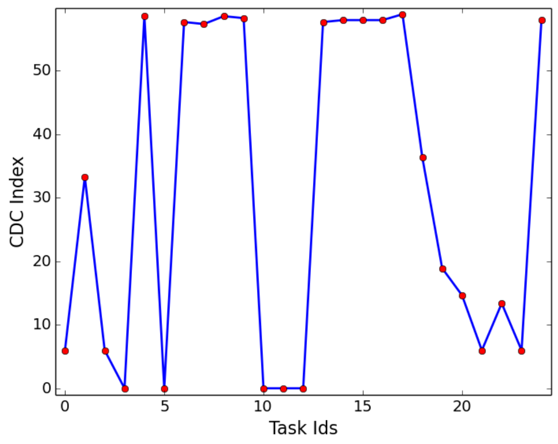

Fig. 6. Critical delay contribution for 25 tasks executed by 10 robots.

when computing risk. Tasks for which the start time at execution time exceeds the planned start time (indicating delay) are considered more risky. An interesting property is that tasks at the end of long task chains (e.g. tasks with ids 7) have large CDC risks; this is in part due to the fact that delays in earlier tasks are propagated to the later ones in the chain, even if these later tasks do not have high variance in their start times.

Our experiments also show that tasks have low probabilities of failing. Failure is the event of a task's time window constraints not being met during execution. Failure probabilities of tasks in Figure 7 are consistent with the hypothesis that more constrained tasks (e.g., task 7) are more likely to fail. This is also consistent with the delay risk reported in Figure 6. The delay probabilities are consistent with the CDC results, because CDC values use delay frequencies in their computation. The reported low probabilities in part show the robustness of our allocation and execution method. Robustness in our method is improved by schedule adjustments and task re-auctioning.

VII. Challenges and Future Directions

The work proposed so far is for a fixed schedule and fixed starting positions of robots and tasks. This level of analysis

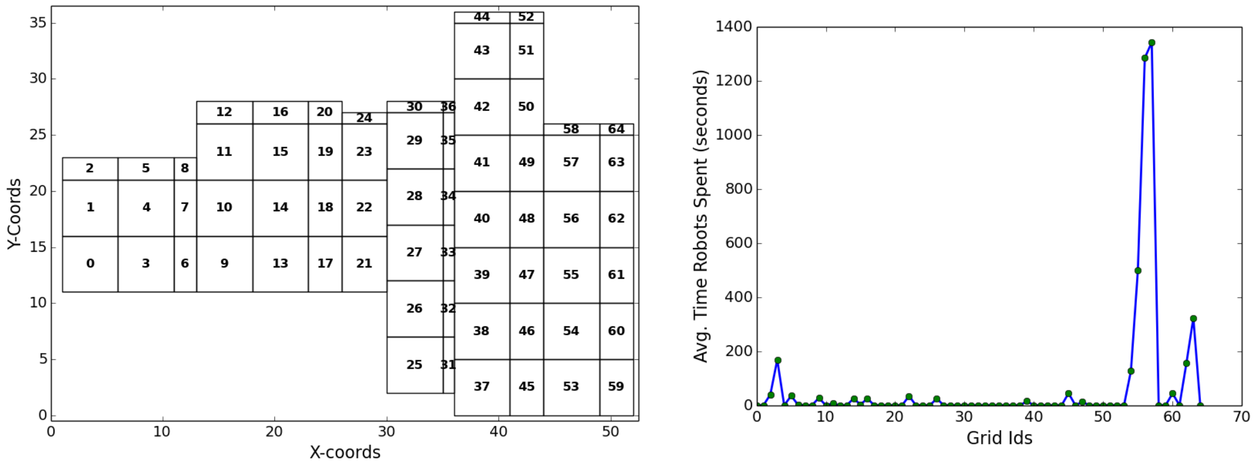

Fig. 4. (Left) 4×4 meter grid decomposition of the map. (Right) Distribution of traversal times over the grid.

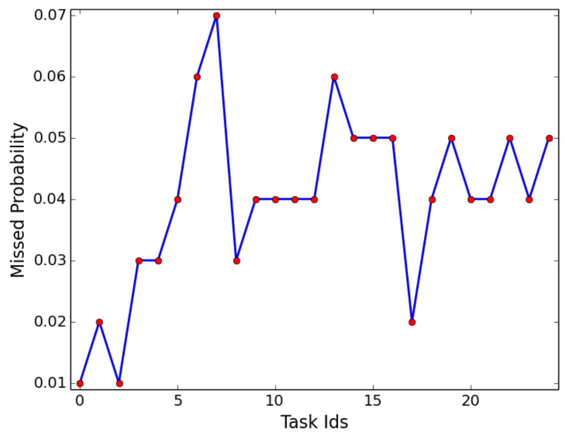

Fig. 7. Probability of task failure measured as frequencies over the number of simulations for 25 tasks and 10 robots.

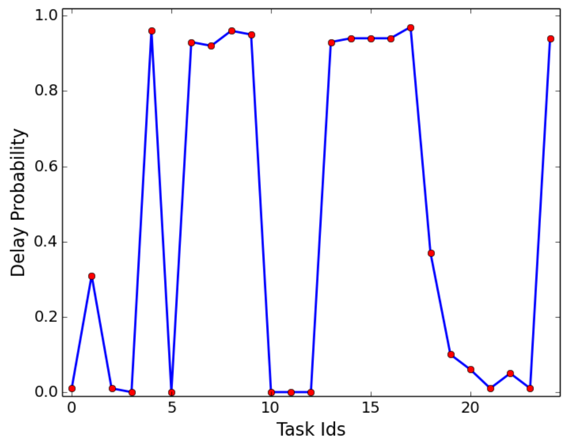

Fig. 8. Probability of task delay measured as frequencies over the number of simulations for 25 tasks and 10 robots.

is appropriate for situations when schedules, and initial positions of robots and tasks do not change over a period of time. However, the analysis might not generalize well to different maps and different configurations of tasks. A more general analysis should include simulations over different schedules and positions of robots and tasks. However, learning over the space of all possible schedules is intractable due to the exponential number of schedules.

The issue of sample complexity (the number of samples required to learn model parameters) in our problem is especially important because individual samples are expensive to obtain. Executing a schedule can take anywhere between 5-20 minutes on commodity hardware. Getting thousands or even millions of samples can take several days. However, sample collection can be made efficient by using parallel computing.

Currently we perform the analysis offline, after we run multiple simulation. We could consider models for online analysis such as dynamic BNs, or probabilistic filtering, and do the analysis online as each simulation runs.

In future work we will also study how to incorporate the risk and probabilistic analysis directly in the allocation and execution system. This raises two important questions: where in the auction and execution process should we include the risk and probabilistic analysis? and which agent should handle the probabilistic reasoning? We have taken steps to include the risk analysis to better inform the prioritization of tasks within our prioritized iterated auction. Further experiments and analysis are necessary to evaluate the effectiveness of the approach.

There are at least two ways to handle probabilistic reasoning: either the auctioneer alone updates the probabilistic models or a combination of the auctioneer and the robots. The main advantage of the former is that saves on communication and computational burden placed on robots. However, this leads to a higher degree of centralization. If robots are to also estimate probabilities, we would need to build a complex message passing system during inference to ensure that inference is performed over an accurate representation of the model. This could lead to high communication and computation complexity.

VIII. Conclusions and Future Work

We provided a first step study of how simulations can be used as an analysis tool in task allocation and execution of tasks with temporal and precedence constraints. We presented risk and probabilistic models that can be readily used to provide information about the robustness of schedules. This information could allow system designers to more easily adjust tasks' temporal constraints. We also provide a grid-based analysis of traversal times on a map. The end goal is to provide a quantitative analysis of which parts of the map cause delays in robots' schedules. Our preliminary results identify risky regions and tasks. Work is underway to extend our analysis to our auction and executor, and to consider more general offline and online probabilistic models.

Acknowledgments: Partial support provided by the National Science Foundation (under grants NSF IIP-1439728, NSF CNF-1531330) and the Doctoral Dissertation Fellowship program from the University of Minnesota.

REFERENCES

[1] X. Miao, P. Luh, D. Kleimnman, and D. Castanon, "Distributed stochastic resource allocation in teams," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 21, no. 1, pp. 61–70, 1991.

[2] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler, "A stochastic and dynamic vehicle routing problem with time windows and customer impatience," *Mobile Networks and Applications*, vol. 14, no. 3, pp. 350–364, 2009.

[3] S. Bopardikar, S. Smith, and F. Bullo, "On dynamic vehicle routing with time constraints," *IEEE Trans. on Robotics*, vol. 30, no. 6, pp. 1524–1532, 2014.

[4] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson, "Planning with deadlines in stochastic domains," in *Proc. AAAI Conf. on Artificial Intelligence*, 1993, pp. 574–579.

[5] A. Beynier and A.-I. Mouaddib, "Decentralized Markov Decision Processes for handling temporal and resource constraints in a multiple robot system," in *Distributed Autonomous Robotic Systems 6*, R. Alami, R. Chatila, and H. Asama, Eds. Springer Japan, 2007, pp. 191–200.

[6] D. A. Dolgov, M. R. James, and M. E. Samples, "Combinatorial resource scheduling for multiagent MDPs," in *Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, 2007, pp. 657–664.

[7] T. Vidal, "Handling contingency in temporal constraint networks: from consistency to controllabilities," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 11, no. 1, pp. 23–45, 1999.

[8] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.

[9] I. Tsamardinos, "A probabilistic approach to robust execution of temporal plans with uncertainty," in *Methods and Applications of Artificial Intelligence: Proc. 2nd Hellenic Conference on AI (SETN '02)*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2002, pp. 97–108.

[10] C. Fang, P. Yu, and B. C. Williams, "Chance-constrained probabilistic simple temporal problems," in *Proc. AAAI Conf. on Artificial Intelligence*, 2014, pp. 2264–2270.

[11] J. Brooks, E. Reed, A. Gruver, and J. C. Boerkoel, "Robustness in probabilistic temporal planning," in *Proc. AAAI Conf. on Artificial Intelligence*, 2015, pp. 3239–3246.

[12] S. S. Ponda, L. B. Johnson, and J. P. How, "Distributed chance-constrained task allocation for autonomous multi-agent teams," in *American Control Conf.*, 2012, pp. 4528–4533.

[13] E. Nunes, M. McIntire, and M. Gini, "Decentralized allocation of tasks with temporal and precedence constraints to a team of robots," in *Proc. SIMPAR*, 2016.

[14] M. McIntire, E. Nunes, and M. Gini, "Iterated multi-robot auctions for precedence-constrained task scheduling," in *Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, 2016, pp. 1078–1086.

[15] S. Creemers, E. Demeulemeester, and S. Van de Vonder, "A new approach for quantitative risk analysis," *Annals of Operations Research*, vol. 213, no. 1, pp. 27–65, 2014.