In *Proceedings of the IJCAI'16 Workshop on Autonomous Mobile Service Robots,*
New York City, USA July 2016

Multirobot Symbolic Planning under Temporal Uncertainty

Shiqi Zhang, Yuqian Jiang, Guni Sharon, and Peter Stone
Department of Computer Science, UT Austin
szhang@cs.utexas.edu; jiangyuqian@utexas.edu; gunisharon@gmail.com; pstone@cs.utexas.edu

Abstract

Multirobot symbolic planning aims at computing plans, each in the form of a sequence of actions, for a team of robots to achieve their individual goals while minimizing overall cost. Solving this problem requires to model the limited resources of a working environment (e.g., corridors that allow at most one robot at a time) and the possibility of action synergy (e.g., multiple robots going through a door after a single door-opening action). However, it is a challenge to plan for resource sharing and realizing synergy in a team of robots due to the robots' noisy action durations. This paper, for the first time, focuses on the problem of multirobot symbolic planning under temporal uncertainty (MSPTU). We present an algorithm inspired by simulated annealing for MSPTU problems. The algorithm has been evaluated using multirobot navigation tasks in simulation. We observed significant improvements in reducing overall cost compared to baselines in which robots do not communicate or model temporal uncertainty.

1 Introduction

Intelligent service robots, including CoBots [Veloso *et al.*, 2015], Dora [Hanheide *et al.*, 2015], KeJia [Chen *et al.*, 2010], and Segbots [Khandelwal *et al.*, 2015], have been developed for different research purposes. A fundamental capability for such robots is the ability to navigate through their environments. At the lowest level, navigation consists of finding continuous paths through local space that avoid obstacles and smoothly reach designated waypoints. However, especially in large-scale environments, it can be useful to select such waypoints by reasoning at a higher level of abstraction.

To this end, symbolic planning techniques allow a robot to compute a sequence of actions, by reasoning about action preconditions and effects, to bring about state transitions in order to achieve a goal that is unreachable using individual actions. For instance, the action of going through a door into a room is preconditioned by the robot being beside the door and the door being open; and the effect is the robot's position being changed to the new room. When action costs are further incorporated into this planning process, robots can compute optimal plans that maximize overall utility (or minimize overall cost).

When multiple robots share a physical environment, their plans might interact such that their independently-computed optimal plans become suboptimal at runtime, due to constrained resources such as narrow corridors that allow at most one robot to pass. On the other hand, a team of robots have the potential to leverage synergies in their plans by coordinating amongst themselves. For instance, when a robot knows its teammate is going to take an expensive door-opening action, it makes sense for the robot to plan to follow its teammate through the door instead of opening it separately. A key challenge to planning for resource sharing and leveraging synergy is the inherent uncertainty in the durations of robots' actions at runtime, which is largely overlooked in existing research. This paper proposes a novel method to meet this challenge.

The first main contribution of this paper is the introduction of the Multirobot Symbolic Planning under Temporal Uncertainty (**MSPTU**) problem. We then present a novel algorithm inspired by simulated annealing search [Kirkpatrick *et al.*, 1983] for coordinating multiple robots under this problem setting, and experimentally evaluate it through comparisons against baselines in which robots do not coordinate their plans, or coordinate but do not model temporal uncertainty.

2 Related Work

This work is closely related to existing research in the areas of symbolic planning, multirobot task scheduling, scheduling under uncertainty, and MDP-based multirobot planning.

Symbolic planning aims at computing a sequence of actions to achieve goals that are possibly unreachable through individual actions. Since the development of STRIPS [Fikes and Nilsson, 1972], many action languages have been developed for symbolic planning by describing preconditions and effects of actions, including PDDL [Ghallab *et al.*, 1998]. In parallel, BC is an action language that is particularly attractive for robotic applications because it can represent recursive fluents, indirect action effects and defaults [Lee *et al.*, 2013] (we use BC in this work). However, these action languages do not support the capability of reasoning about noisy action durations, which is critical for multirobot planning toward sharing resources and constructing synergy at runtime.

A multirobot scheduling problem's input includes a set of robots and a set of tasks, and the output is a schedule that is

for each task an allocation of one or more time intervals to one or more robots [Brucker, 2007; Zhang and Parker, 2013]. Recent work on multirobot scheduling further considers temporal uncertainty [Brooks *et al.*, 2015]. However, scheduling algorithms are not developed for symbolic planning and hence cannot generate action sequences in large, complex domains.

In parallel of symbolic planning, (PO)MDP-based planning techniques have been extensively studied in the literature. Existing (PO)MDP-based research has studied planning with concurrent actions [Mausam and Weld, 2008; Smith and Weld, 1999], planning under temporal uncertainty [Guo and Hernández-Lerma, 2009; Younes and Simmons, 2004], incorporating temporal logic into navigation task planning [Fentanes *et al.*, 2015], and planning for multirobot systems [Khandelwal *et al.*, 2015; Zhang *et al.*, 2013]. Such algorithms are good at handling non-deterministic action outcomes using probabilities and planning toward maximizing long-term reward. In contrast, symbolic planning techniques, such as STRIPS, PDDL and BC, fall into a very different planning paradigm, where the input are action preconditions and effects, non-deterministic action outcomes are handled by plan monitoring and replanning, and the output is a sequence of actions that can be easily interpreted by human users (compared to a policy in (PO)MDP). Symbolic planning techniques have been widely used in service robot applications (e.g., [Chen *et al.*, 2010], [Zhang *et al.*, 2015] and [Khandelwal *et al.*, 2014]).

Existing work has investigated the problem of concurrent task assignment and planning of trajectories for a team of robots [Turpin *et al.*, 2014; Ma and Koenig, 2016]. Given N robots and N goal locations, the algorithms aim to find a suitable assignment of robots to goals and the generation of collision-free, time parameterized trajectories for each robot. However, their approach is only applicable to navigation problems and they do not model noisy action durations. This paper, for the first time, focuses on multirobot symbolic planning under temporal uncertainty.

3 Algorithm

In this section, we first define the MSPTU problem (§ 3.1) and present our symbolic planner for single-robot navigation tasks (§ 3.2). We then introduce our *Poisson*-based model of noisy action durations (§3.3) and how to compute conditional plan cost in two-robot systems (§ 3.4). Finally, we introduce our MSPTU algorithm (§3.5). The problem definition and our MSPTU algorithm are not restricted to specific applications.

3.1 Problem definition

In this paper, we assume each robot can work on at most one task at a time and each task requires only one robot, which corresponds to the "single-robot", "single-task" problems following Gerkey and Matarić's taxonomy [Gerkey and Matarić, 2004]. We assume the robots are homogeneous, sharing the same set of sensing and actuating capabilities. The performance of an MSPTU algorithm is evaluated by episode in this paper. At the beginning of an episode, each robot has exactly one task assigned and the tasks are not transferable. The end of an episode is identified by the time of the slowest robot finishing its task. A *central controller* runs our MSPTU algorithm to compute plans for all robots and robots do not have to make any decision themselves (i.e., robots work in a centralized system). We assume that robots have a noise-free communication channel that enables coordination.

Given a domain that includes N robots, an MSPTU problem is of the form $\langle \mathcal{D}, \mathcal{A}, \mathcal{S}, \mathcal{G}, \mathcal{R} \rangle$:

- $\mathcal{D}$ is a description of objects (including robots) in the domain, their properties, and their relations.
- $\mathcal{A}$ is a description of robot actions, including their preconditions, effects and costs.
- $\mathcal{S}$ is a set of states in which each is the initial state of a robot: $s_i \in \mathcal{S}$ is the state of the ith robot and $|\mathcal{S}| = N$. A robot's state does not include the state of other robots.
- $\mathcal{G}$ is a set of goal states in which each corresponds to a robot: $g_i \in \mathcal{G}$ is the goal state of the ith robot and $|\mathcal{G}| = N$.
- $\mathcal{R}$ is a set of constrained resources, each associated with a cost of violation, that can be obtained by at most M robot at a time, where $M < N$.

Domain description $\mathcal{D}$ is about a static environment that includes all robots in it, e.g., two rooms are accessible via a door that is being closed. Action description $\mathcal{A}$ focuses on robot capabilities of making changes in the domain, e.g., a door-opening action can change a door's property from "closed" to "open". An action can be executed only if all preconditions are fulfilled. A robot's initial state, $s \in \mathcal{S}$, and goal, $g \in \mathcal{G}$, are specified by values of domain properties. $\mathcal{D}$ and $\mathcal{A}$ correspond to the rigid and dynamic laws of action languages respectively (details in Section 3.2).

Under temporal uncertainty, robots can only succeed in sharing constrained resources probabilistically. We say robots fail in sharing a resource, if K robots physically compete for a constrained resource that can only be shared by M robots and $K > M$. It is difficult but necessary to model the consequences of such failed cases in planning. For instance, two robots competing for a narrow corridor may cause collisions, detours, and many other consequences. For the sake of simplicity, we model collaboration failures (violations of constraints) with fixed costs.

The goal of solving an MSPTU problem is to compute symbolic plans, each in the form of a sequence of actions, for a team of robots to achieve their individual goals while minimizing expected overall cost. Therefore, the output of an MSPTU algorithm is a set of plans, one for each robot. Since MSPTU problems model the temporal uncertainty in the durations of robots' actions, it is necessary to replan when uncertainty has been reduced considerably at runtime, e.g., after a robot finishes a time-consuming action.

3.2 Single-robot symbolic planning using BC

We use action language BC [Lee *et al.*, 2013] in this work because it can formalize defaults (e.g., without knowing the contrary, we believe office doors are closed over weekends) and recursively defined fluents (e.g., two rooms are *accessible* to each other if each of them is *accessible* to a third room). However, the algorithms developed in this paper are

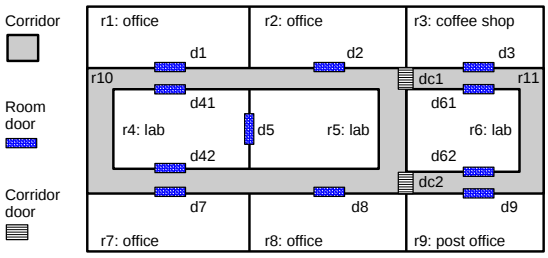

Figure 1: A domain map used in simulation experiments. Opening a corridor door is an expensive action.

not restricted to specific action languages or symbolic planners. It should be noted that planning for single-robot navigation tasks using BC has been studied, e.g., [Khandelwal *et al.*, 2014], and we adapt that formulation for multirobot settings.

Figure 1 shows the domain we use for instantiating our approach and in experiments. In domain description $\mathcal{D}$, corridors ($r10$ and $r11$), labs ($r4$, $r5$ and $r6$), coffee shop ($r3$), and offices ($r1$, $r2$, $r7$, $r8$ and $r9$) are rooms. Room doors ($d1$, $d2$, $\cdots$) and corridor doors ($dc1$ and $dc2$) are doors. The following rules define the ownership between rooms and doors.

$$hasdoor(r1, d1).\ \ hasdoor(r10, dc1).\ \ \cdots$$
$$\textbf{default}\ \ \neg hasdoor(R, D).$$

where, R and D represent a room and a door respectively. The last rule above is a *default* for reasoning with incomplete knowledge: it is believed that room R does not have door D unless there is evidence supporting the contrary.

Action description, $\mathcal{A}$, includes the rules that formalize the preconditions and effects of actions that can be executed on each robot. We use fluents *open*(D), *facing*(D), *beside*(D), and *loc*(R) to represent door D is open, the robot is facing door D, the robot is beside door D, and the robot is in room R. Robot identities are not included in the representation of a robot's location, *loc*(R), because a robot's state does not model the state of other robots. Robot actions include *approach*(D), *opendoor*(D), *cross*(D), and *waitforopen*(D), where *waitforopen*(D) enables a robot to wait for another robot to open door D and is only useful in multirobot systems. Due to space limit, we arbitrarily select action *cross*(D) and present its definition as below. Crossing door D changes the robot's location from R_1 to R_2, the room on the other side of door D. The last three rules below describe the executability, e.g., *cross*(D) cannot be executed if door D is not open.

$cross(D)$ **causes** $\neg facing(D)$.
$cross(D)$ **causes** $loc(R_2)$ **if** $loc(R_1),\ acc(R_1, D, R_2)$.
nonexe $cross(D)$ **if** $loc(R),\ \neg hasdoor(R, D)$.
nonexe $cross(D)$ **if** $\neg facing(D)$.
nonexe $cross(D)$ **if** $\neg open(D)$.

Given a planning goal, a planner can find many solutions. We select the one that minimizes the overall cost. As an example, if the robot is initially placed in the post office, $r9$, and the goal is to move to lab $r6$, our planner will produce a plan:

$$approach(d9).\ opendoor(d9).\ cross(d9).$$
$$approach(d62).\ opendoor(d62).\ cross(d62).$$

In implementation, to model the progress of navigation actions (*approach*(D), in our case). We discretize distance by representing each corridor using a set of grid cells. Accordingly, each *approach*(D) action is replaced by a sequence of actions that lead the robot following waypoints.

3.3 Modeling noisy action durations

This subsection presents a novel model for representing and reasoning about temporal uncertainty in noisy action durations. In this paper, we consider only the uncertainty from action *approach*(D) and deriving this action's probability density function (PDF) builds on the following assumptions:

1. Unless explicitly delayed, all robots move at the same velocity, v. Unless specified otherwise, $v=1$ in this paper.
2. A human obstacle appears within every unit distance at a known rate, and their appearances are independent of each other. We use λ to denote this rate.
3. While taking action *approach*(D), each obstacle appearance causes a delay for a known amount of time, δ, for example, by forcing the robot to stop and say "excuse me".

Following Assumptions 1 and 2, we can use a Poisson distribution to model the number of delays caused by human appearances in a unit time and its corresponding PDF is:

$$\hat{f}(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{1}$$

where e is Euler's number and k is the number of delays.

Proposition 1: If X and Y are two independent discrete random variables with a Poisson distribution: $X \sim Poisson(\lambda_1)$ and $Y \sim Poisson(\lambda_2)$, then their sum $Z = X + Y$ follows another Poisson: $Z \sim Poisson(\lambda_1 + \lambda_2)$ [Knill, 1994].

According to Proposition 1, when a robot travels for time t (instead of unit time), the number of delays, k', accumulates over time and follows another Poisson distribution with a PDF of $f(k', \lambda')$. Following Assumption 1, parameter λ' is a function of traveled distance d:

$$\lambda'(d) = \lambda \cdot t(d) = \lambda \cdot d/v \tag{2}$$

Since k' follows a Poisson distribution, we can compute the overall time needed for traveling a distance of d:

$$t = t^{act} + t^{del} = d/v + k' \cdot \delta \tag{3}$$

where $t^{act} = d/v$, as a linear function of distance d, represents the acting time, and $t^{del} = k' \cdot \delta$ is the delayed time.

Using Equations 2 and 3, we can see the overall navigation time t follows a (non-Poisson) distribution with PDF:

$$f\big(t, \lambda'(d)\big) = \frac{\big(\lambda'(d)\big)^{\frac{t-d/v}{\delta}} \cdot e^{-\lambda'(d)}}{\big(\frac{t-d/v}{\delta}\big)!} \tag{4}$$

Figure 2 visualizes two example PDFs. For instance, it is the most likely that traveling distance $d = 50$ at velocity $v = 1$ takes 60 time units while modeling possible delays (instead of 50 in obstacle-free domains). It also shows that a longer distance produces more uncertainty in completion time. We also collected navigation time using a real robot, and the results suggest that a shifted Poisson distribution can well represent noisy durations of navigation actions (Figure 3).

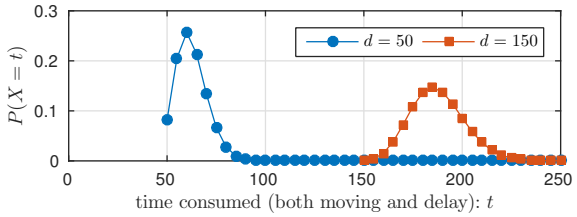

Figure 2: Two example PDFs of *shifted* Poisson distributions used for modeling the noisy durations of navigation actions: $v=1$, $\delta=5$, and $\lambda=0.05$. Comparing the two PDF curves, we can see a longer distance brings more uncertainty.

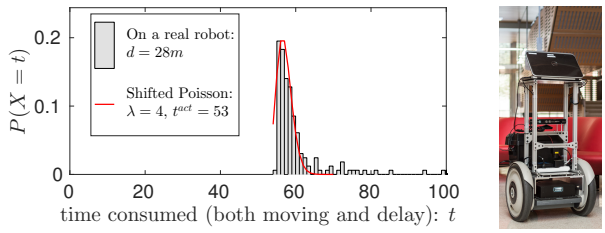

Figure 3: To experimentally justify the use of Poisson distribution, we used a real robot to navigate in a corridor ($28m$) for 164 times. The **Left** shows the histogram of traveling time and a shifted Poisson that well models the action's noisy durations (with parameters properly set), and the **Right** shows the Segway-based robot.

We further remove the parameter of λ' and substitute d/v with t^{act}, so Equation 4 is converted into Equation 5, modeling action completion time. For action *approach*(D), $\lambda'>0$, meaning that the time of action completion can be delayed. For other actions, $\lambda'=0$.

$$f(t,\lambda') = \frac{(\lambda')^{\frac{t-t^{act}}{\delta}} \cdot e^{-\lambda'}}{(\frac{t-t^{act}}{\delta})!} \tag{5}$$

We use $Dist(t^{act},\lambda')$ to represent a distribution over possible lengths of completion time with a density function of Equation 5. Modeling noisy action durations in this way paves the way to further investigating how uncertainty is accumulated over plans that include a sequence of actions. For instance, $t^{act}=50$ and $\lambda'=2.5$ correspond to the (blue) circle-mark curve in Figure 2. Since (we assume) non-navigation actions do not introduce extra uncertainty at running time, Equation 3 can be directly applied to modeling the distribution over possible lengths of time consumed by a sequence of actions including potentially both navigation and non-navigation actions. A plan of form $\langle a_0, a_1, \cdots \rangle$ can be represented as below to further model the distribution over possible lengths of completion time of each action. We call p an *extended* plan (or simply plan).

$$p: \langle (a_0, t_0^{act}, \lambda_0'), (a_1, t_1^{act}, \lambda_1'), \cdots \rangle$$

According to Proposition 1, the time consumed by executing the first K actions in plan p follows a distribution of:

$$Dist(\sum_{k=0}^{K-1} {t_k}^{act}, \sum_{k=0}^{K-1} \lambda_k')$$

Therefore, $Dist(t^{act},\lambda')$ represents a novel distribution that can model the temporal uncertainty that accumulates over a sequence of actions in robot navigation problem. Note that other application domains may require very different representations (PDFs) for modeling their noisy action durations, and this subsection simply presents a concise PDF representation for navigation actions.

3.4 Computing conditional plan cost

In a two-robot system that includes robots R and R', p and p' are robots' extended plans. The *conditional plan cost* of p' given p is the estimate of total cost robot R' will consume, if R and R' simultaneously execute their plans, p and p', respectively. Different from single-robot planning, we have to consider possible collisions and door-sharing behaviors (and any conflicts or synergies in general) in computing conditional plan costs. It should be noted that we use integrals in this section primarily for the cleanness of representations. In implementation, the integrals are replaced by summation operations, because action completions only happen at specific time instances (e.g., Figure 2). We first compute the probability of robot R''s navigation action a' overlapping p's navigation action a over time (parameter λ omitted from PDFs):

$$Pr^{ovlp}(a,a') = 1 - \int_0^\infty\int_{t_2}^\infty f_1^s(t_1) f_2^c(t_2)\, dt_1\, dt_2 - \int_0^\infty\int_{t_1}^\infty f_1^c(t_1) f_2^s(t_2)\, dt_2\, dt_1 \tag{6}$$

where f_1^s and f_1^c are the PDFs of starting and completion times of action a; f_2^s and f_2^c are the PDFs of starting and completion times of action a'. The first double integral computes the probability of the completion of a' being earlier than the start of a, and the second computes the probability of the start of a' being after the completion of a.

In modeling door-sharing behaviors, we assume: 1) all doors are automatic in such a way that they give enough going-through time for the robots that are ready to go through the door; and 2) a door closes automatically when it detects no more robot awaiting. We use $t^{wait}(a,a')$ to represent the time of robot R' waiting for R to open door D, where a' is R''s action and is *waitforopen*(D).

$$t^{wait}(a,a') = \int_0^\infty\int_{t_2}^\infty (t_1 - t_2) f_1^c(t_1) f_2^s(t_2)\, dt_1\, dt_2$$

where f_1^c is the PDF of the completion time of action a; and f_2^s is the PDF of the start time of action a'.

It is possible that robot R has finished the action of going through door D before robot R' arrives. In this case, robot R' may have avoided closer doors and has to reopen the door. We compute the probability of such failures:

$$Pr^{fail}(a,a') = \int_0^\infty\int_{t_1}^\infty f_1^c(t_1) f_2^s(t_2)\, dt_2\, dt_1 \tag{7}$$

where f_1^c is the time of robot R completing action a, the action of crossing door D, and f_2^s is the time of robot R' starting the action of *waitforopen*(D), i.e., a'.

Algorithm 1 presents our algorithm for computing conditional plan cost (in our navigation domain). While computing the cost of *waitforopen*(D), we need to consider both

Algorithm 1 Computing conditional plan cost (in navigation)

Input: Plan p', where $|p'| = K'$
Input: Plan set P that the cost of p' is conditioned on
Input: μ : collision cost
Input: ω : *waitforopen*(D) failure cost
Input: ρ : value of time
Output: C': overall cost of plan p'

1: C'=0
2: **for each** $p \in P$ **do**
3: **for each** $a \in p$ **do**
4: **for each** $a' \in p'$ **do**
5: $C' \leftarrow C' + cost(a')$
6: **if** a is *opendoor*(D) and a' is *waitforopen*(D) **then**
7: $C' \leftarrow C' + \rho \cdot t^{wait}(a, a') \cdot \big(1 - Pr^{fail}(a, a')\big)$
8: $C' \leftarrow C' + \omega \cdot Pr^{fail}(a, a')$
9: **else if** both a and a' are navigation actions **then**
10: $C' \leftarrow C' + \mu \cdot Pr^{ovlp}(a, a')$
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **return** C'

the cases that have synergy and those that failed (Lines 7-8). Although the form of temporal uncertainty varies significantly over different robot actions, this approach can be easily applied to other domains for sharing limited resource and constructing "*wait-for-action*"-style synergies, as long as the PDFs of the actions' durations are available. In the next section, we present our main algorithm that addresses general MSPTU problems.

3.5 Our MSPTU algorithm

Planning for a team of robots under temporal uncertainty in a joint-action, continuous-time search space is intractable from a practical point of view, even if the number of robots and the lengths of individual plans are within a reasonable range.

Algorithm 2 shows our novel algorithm for N-robot MSPTU problems, where $N \geq 2$. This algorithm is inspired by *simulated annealing* search for approximating the global optimum of overall system utility [Kirkpatrick *et al.*, 1983]. The input includes the robots' initial and goal states, and a set of parameters that have been defined in Algorithm 1. The input also includes M, the number of other robots being considered while computing conditional plan cost, and Θ that represents how many rounds of "negotiations" the robots can have before finalizing their plans. The output is a set of plans, one for each robot. Algorithm 2 has $O(\Theta \cdot N)$ complexity for operations of computing conditional plan costs, where N is the number of robots.

The first for-loop (Lines 2-5) is used for computing one plan for each robot, ignoring the existence of the other robots. If $\Theta = 0$, meaning that there is no collaboration among the robots, the independently-computed optimal plans are returned. Otherwise, the program enters the second for-loop (Lines 7-14), where α is a *negotiation depth* that incrementally grows by $1/\Theta$ in each iteration. The loop continues until α reaches 1. Intuitively, the negotiation depth measures how much a robot considers its teammates: when $\alpha = 0$, it totally ignores its teammates; when $\alpha = 1$, it considers its teammates as important as itself, in terms of plan cost. In the inner for-loop (Lines 9-13), we first compute plan for the jth robot while minimizing the conditional plan cost given the plans computed for the previous M robots (Line 10). We then update the jth robot's plan and plan cost. In the end of the program, a set of plans is returned.

Algorithm 2 Our MSPTU algorithm

Input: $\mathcal{S}$, a set of N states, and, $\mathcal{G}$, a set of N goals ($N \geq 2$)
Input: μ, ω and ρ, as defined in Algorithm 1
Input: M: number of other robots conditioned in Alg-1, $M < N$
Input: Θ: number of iterations, $\Theta \geq 0$
Output: $[p_1, p_2, \cdots, p_N]$

1: Initialize an empty plan *queue* of size M: P^M
2: **for each** $i \in [1, N]$ **do**
3: Compute plan p_i and corresponding cost $cost(p_i)$ (§3.2)
4: Push back p_i to P^M
5: **end for**
6: **if** $\Theta \neq 0$ **then**
7: **for each** $i \in [1, \Theta]$ **do**
8: $\alpha = i/\Theta$, where α is the negotiation depth
9: **for each** $j \in [1, N]$ **do**
10: $[p'_j,\ cost(p'_j)] \leftarrow$ Alg-1 $(p'_j,\ P^M,\ \alpha\mu,\ \alpha\omega,\ \rho)$
11: $p_j \leftarrow p'_j$ and $cost(p_j) \leftarrow cost(p'_j)$
12: Push back p_j to P^M
13: **end for**
14: **end for**
15: **end if**
16: **return** $[p_1, p_2, \cdots, p_N]$

Although the output of Algorithm 2 includes plans for all robots to achieve their goals, the robots do not necessarily follow the plans all the way to the end of episodes. The temporal uncertainty of a plan is reduced after an action of the plan is completed. We recompute plans for all robots after one of the robots finishes its current action. However, we have noticed that it makes sense to activate replanning only if the change in uncertainty is significant and only for the robots who have potential to find better plans given the uncertainty change. We leave further exploration of this issue to future work.

4 Experiments

Our MSPTU algorithm has been implemented and evaluated using a team of mobile robots working on navigation tasks in an indoor office environment. We use two types of simulation in our experiments: an abstract simulator and GAZEBO [Koenig and Howard, 2004]. In the abstract simulator, navigation actions' noisy durations are sampled from a shifted Poisson distribution (Eqn. 4); collision cost is 40; *waitforopen*(D) failure cost is 12; and collisions are possible only if both robots are taking navigation actions. In GAZEBO, we add human walkers into the environment to simulate the process of walking people causing delays to robot navigation actions, and the performance is evaluated based robots' average completion time (no arbitrary costs are used). We use CLINGO4 for solving BC programs [Gebser *et al.*, 2014].

GAZEBO simulation: Figure 4 (**Left**) shows the GAZEBO simulation environment used in the first set of experiments. Two robots need to navigate from their initial positions, labeled by green dots in the bottom, to their goal positions, labeled by red dots on the top. The two robots start at the

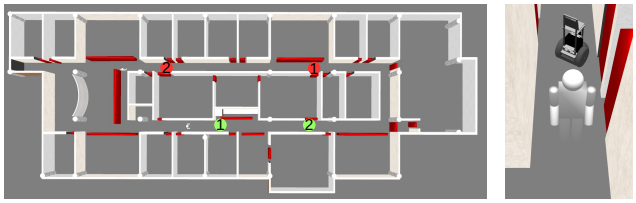

Figure 4: GAZEBO simulation environment: **Left** shows a top-down view, and **Right** shows a robot and a human walker.

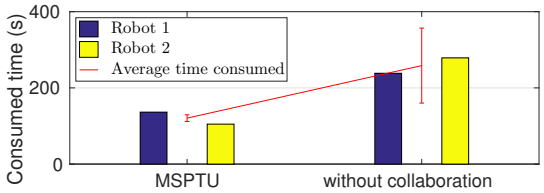

Figure 5: Average time consumed by two robots doing navigation tasks in the GAZEBO environment, where the initial and goal positions of the two robots are labeled with green (bottom) and red (top) dots respectively in Figure 4.

same time, and we record the completion time for each of the robots. Figure 4 (**Right**) shows a screenshot of a walker blocking the way of the robot. Figure 5 shows the results collected from the experiments conducted in GAZEBO. Without collaboration, robot $R2$ takes an average of 278 seconds to achieve its goal, and our MSPTU successfully reduces $R2$'s average execution time to about 104 seconds. While considering both robots, MSPTU reduces the completion time from more than 250 seconds to less then 120 seconds, and the improvement is significant.

Abstract simulation: Figure 6 reports the results of experiments evaluating the probabilistic model of noisy action durations. The head start of $R1$ varies in a relatively small range in x-axis (± 6). $R1$ starts at a position close to the top corridor door and needs to move to its goal position on the other side of the door. When robot $R1$ has a head start of -1, the overall cost of our MSPTU algorithm is smaller than the baseline by more than ten (reduced from more than 50 to less than 40). Focusing on this performance improvement, we find robot $R2$ can either open the bottom door by itself or follow the first robot through the corridor door on the top (trajectories shown as dashed lines). Without modeling temporal uncertainty (baseline), $R2$ is not aware of the risk of being delayed while moving upward. In contrast, using our MSPTU algorithm, $R2$ dynamically evaluates the uncertainty from its teammate and itself, and is able to balance the risk and potential benefit to select the best path.

To evaluate the scalability of our MSPTU algorithm, we increase the number of robots from two to three. The initial and goal positions of these robots are shown in Figure 7. Each robot can have a head start of 0, 15 or 30 (randomly selected). After removing duplicates (e.g., head starts of $[0, 0, 0]$ and $[15, 15, 15]$ are equivalent), there are 19 combinations in total. For each combination, we conducted 10 trials. We vary the number of iterations Θ (1 and 2) and the number of other robots conditioned in Algorithm 1 (1 and 2), so there are 4 combinations of parameters, corresponding to 4 configurations of MSPTU. The average cost over the three robots are

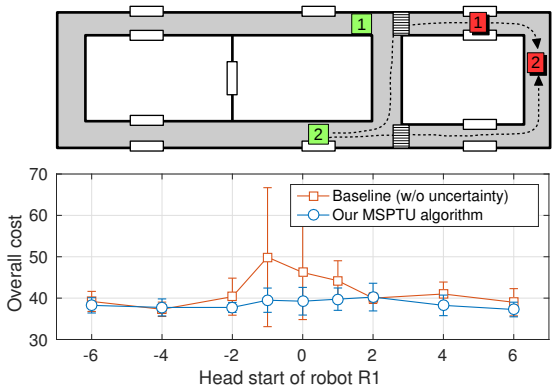

Figure 6: Planning for a two-robot system using our MSPTU algorithm and a baseline that does not model temporal uncertainty. Dashed lines show two possible ways for robot $R2$. The way on the top is risky for $R2$, because it can be delayed in moving upward, making $R2$ too late to follow $R1$ through the top corridor door.

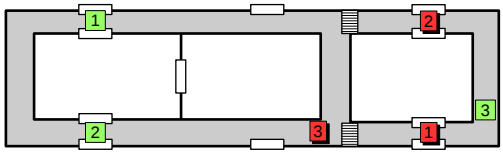

Figure 7: Initial and goal positions while planning for a three-robot system using our MSPTU algorithm. Start time of each of the three robots can be delayed by 0, 15, or 30 units (randomly decided).

presented in Table 1. The reduction of average cost by considering plans of all other robots is significant, regardless of Θ's value: p-value=0.03 when $\Theta = 1$, and p-value=0.02 when $\Theta = 2$. However, running two iterations does not produce significant improvement in reducing the average cost. The results indicate that $\Theta = 1$ is the best choice in setting the number of iterations in the current settings.

Table 1: Results of planning for three robots using our MSPTU algorithm: average cost (over three robots) with standard deviation.

Number of iterations: Θ	Number of other robots conditioned in Algorithm 1	
	Only previous (1)	All others (2)
1	71.77 (25.17)	57.86 (3.55)
2	71.44 (24.97)	56.38 (2.75)

5 Conclusions

In this paper, we introduce the multirobot symbolic planning under temporal uncertainty (MSPTU) problem. Although symbolic planning techniques have been widely used on single-robot systems for plan generation, when multiple robots share a physical environment, their independently-computed optimal plans might become suboptimal, due to constrained resources and missed potential to leverage synergies. As the first work focusing on the MSPTU problem, we develop a novel algorithm inspired by simulated annealing using an action language for symbolic planning, and models noisy action durations using Poisson-like distributions. The algorithm has been evaluated using multirobot navigation tasks in simulation. We observe significant improvements against baselines where robots do not coordinate their plans, or coordinate but do not model temporal uncertainty.

Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287), ONR (21C184-01), and AFOSR (FA9550-14-1-0087). Peter Stone serves on the Board of Directors of, Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

[Brooks *et al.*, 2015] Jeb Brooks, Emilia Reed, Alexander Gruver, and James C Boerkoel Jr. Robustness in probabilistic temporal planning. In *National Conference on Artificial Intelligence (AAAI)*, 2015.

[Brucker, 2007] Peter Brucker. *Scheduling algorithms*. Springer, 2007.

[Chen *et al.*, 2010] Xiaoping Chen, Jianmin Ji, Jiehui Jiang, Guoqiang Jin, Feng Wang, and Jiongkun Xie. Developing high-level cognitive functions for service robots. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 989–996, 2010.

[Fentanes *et al.*, 2015] Jaime Pulido Fentanes, Bruno Lacerda, Tomas Krajnik, Nick Hawes, and Marc Hanheide. Now or later? predicting and maximising success of navigation actions from long-term experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1112–1117. IEEE, 2015.

[Fikes and Nilsson, 1972] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.

[Gebser *et al.*, 2014] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014.

[Gerkey and Matarić, 2004] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research (IJRR)*, 23(9):939–954, 2004.

[Ghallab *et al.*, 1998] Malik Ghallab, Craig Knoblock, David Wilkins, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David E Smith, Sun Ying, and Daniel Weld. Pddl-the planning domain definition language. 1998.

[Guo and Hernández-Lerma, 2009] Xianping Guo and Onésimo Hernández-Lerma. *Continuous-time Markov decision processes*. Springer, 2009.

[Hanheide *et al.*, 2015] Marc Hanheide, Moritz Göbelbecker, Graham S Horn, Andrzej Pronobis, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Charles Gretton, Richard Dearden, Miroslav Janicek, et al. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 2015.

[Khandelwal *et al.*, 2014] Piyush Khandelwal, Fangkai Yang, Matteo Leonetti, Vladimir Lifschitz, and Peter Stone. Planning in Action Language $\mathcal{BC}$ while Learning Action Costs for Mobile Robots. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[Khandelwal *et al.*, 2015] Piyush Khandelwal, Samuel Barrett, and Peter Stone. Leading the way: An efficient multi-robot guidance system. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1625–1633, 2015.

[Kirkpatrick *et al.*, 1983] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Knill, 1994] Oliver Knill. Probability and stochastic processes with applications. *Havard Web-Based*, 1994.

[Koenig and Howard, 2004] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.

[Lee *et al.*, 2013] Joohyung Lee, Vladimir Lifschitz, and Fangkai Yang. Action language bc: Preliminary report. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 983–989. AAAI Press, 2013.

[Ma and Koenig, 2016] Hang Ma and Sven Koenig. Optimal target assignment and path finding for teams of agents. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2016.

[Mausam and Weld, 2008] Mausam and Daniel S Weld. Planning with durative actions in stochastic domains. *J. Artif. Intell. Res.(JAIR)*, 31:33–82, 2008.

[Smith and Weld, 1999] David E Smith and Daniel S Weld. Temporal planning with mutual exclusion reasoning. In *IJCAI*, volume 99, pages 326–337, 1999.

[Turpin *et al.*, 2014] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112, 2014.

[Veloso *et al.*, 2015] Manuela M. Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. CoBots: Robust symbiotic autonomous mobile service robots. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[Younes and Simmons, 2004] Håkan LS Younes and Reid G Simmons. Solving generalized semi-markov decision processes using continuous phase-type distributions. In *The AAAI Conference on Artificial Intelligence*, 2004.

[Zhang and Parker, 2013] Yu Zhang and Lynne E Parker. Multi-robot task scheduling. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2992–2998. IEEE, 2013.

[Zhang *et al.*, 2013] Shiqi Zhang, Mohan Sridharan, and Christian Washington. Active visual planning for mobile robot teams using hierarchical POMDPs. *IEEE Transactions on Robotics*, 29(4):975–985, 2013.

[Zhang *et al.*, 2015] Shiqi Zhang, Fangkai Yang, Piyush Khandelwal, and Peter Stone. Mobile robot planning using action language bc with an abstraction hierarchy. In *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, Lexington, KY, USA, September 2015.