Multirobot Symbolic Planning under Temporal Uncertainty

Shiqi Zhang[1,2], Yuqian Jiang[1], Guni Sharon[1], and Peter Stone[1]
[1] Department of Computer Science, The University of Texas at Austin
[2] Department of Electrical Engineering and Computer Science, Cleveland State University
s.zhang9@csuohio.edu; jiangyuqian@utexas.edu; gunisharon@gmail.com; pstone@cs.utexas.edu

ABSTRACT

Multirobot symbolic planning (MSP) 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 MSP problems requires modeling limited domain resources (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, the temporal uncertainty that propagates over actions, such as delays caused by obstacles in navigation actions, makes it challenging to plan for resource sharing and realizing synergy in a team of robots. This paper, for the first time, introduces the problem of MSP under temporal uncertainty (MSPTU). We present a novel, *iterative inter-dependent planning* (IIDP) algorithm, including two configurations (*simple* and *enhanced*), for solving general MSPTU problems. We then focus on multirobot navigation tasks, presenting a full instantiation of IIDP that includes a new algorithm for computing conditional plan cost under temporal uncertainty and a novel *shifted-Poisson* distribution for accumulating temporal uncertainty over actions. The algorithms have been implemented both in simulation and on real robots. We observed a significant reduction in overall cost compared to baselines in which robots do not communicate or model temporal uncertainty.

CCS Concepts

•**Computing methodologies** → **Robotic planning; Multi-agent planning; Planning under uncertainty;**

Keywords

Multirobot task planning; Planning under temporal uncertainty; Intelligent mobile robotics

1. INTRODUCTION

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). As a result, symbolic planning techniques have been widely used in applications of intelligent mobile robots, such as indoor service tasks [6, 30, 15, 18] and search and rescue [19, 28].

When multiple robots share a physical environment (such as our Segway-based robots, BWIBots [18], that are shown in Figure 1), it is necessary to model how their plans interact with each other and to construct action synergies as possible. On one hand, the robots' 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*. While such conflicts can be resolved locally at runtime (e.g., two robots detecting a conflict can "negotiate" to decide who gives way to the other [32]), we argue a better solution is to avoid such conflicts at planning time. On the other hand, communications within 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*. One of the key challenges to planning for such resource sharing and leveraging synergy is the inherent temporal uncertainty in the durations of robots' actions at runtime, which is largely overlooked in existing research. The main contribution of this paper includes:

- The introduction of the multirobot symbolic planning under temporal uncertainty (**MSPTU**) problem, and
- A novel, Iterative Inter-Dependent Planning (**IIDP**) algorithm inspired by simulated annealing [20] for MSPTU problems.

Figure 1: Three of our Segway-based mobile robot platforms.

IIDP is generally applicable to MSPTU problems and is not restricted to specific symbolic planners, forms of noisy action durations or application domains. [1] IIDP has two configurations that have different trade-offs between computational complexity and

[1]A MSP problem is a special form of a MSPTU problem, assuming no temporal uncertainty, so IIDP is applicable to MSP problems.

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

plan quality: *simple*-IIDP (**S-IIDP**) and *enhanced*-IIDP (**E-IIDP**). The biggest advantage of IIDP is that it does not increase a MSPTU problem's dimensionality beyond a single agent, while still being able to produce near-optimal solutions (Section 4). IIDP is named as being "inter-dependent" because an important step in IIDP calls an external algorithm that computes an optimal plan for a robot under the condition of other robots' current plans (i.e., this planning process depends on existing plans). This inter-dependent planning algorithm is required by IIDP, but can be independently developed.

As the secondary contribution of this paper, we present a full instantiation of the IIDPs, as applied to a multirobot navigation task, where a new (domain-specific) conditional planning algorithm is developed. The aim of using this task domain includes demonstrating the need of modeling temporal uncertainty in MSP problems, illustrating the whole process of instantiating IIDP, and experimentally evaluating the performance of IIDP in a real-world problem.

This paper is composed of two parts: the IIDP algorithm and the (domain specific) inter-dependent planning algorithm, where the former utilizes the latter. The algorithms have been implemented both in simulation and on real robots using a multirobot navigation problem. Evaluations were conducted via comparisons against baselines in which robots do not coordinate their plans, or coordinate but do not model temporal uncertainty. We observe no significant difference when robots' individually computed plans do not overlap in time or space. When such overlaps exist, we observe IIDP enables a team of robots to avoid going into the same corridor at planning time, leverage action synergy by sharing door-opening actions, and significantly reduce the overall plan cost.

2. RELATED WORK

Single-robot symbolic planning: Since the development of action language STRIPS [9], many action languages have been developed for symbolic planning by describing preconditions and effects of actions, including PDDL [13] that is arguably the most widely used. PDDL was developed for and maintained by the International Planning Competition (IPC) community since 1998. BC is an action language that is particularly attractive for robotic applications because it can represent recursive fluents, indirect action effects and defaults [24] (we use BC in this work). However, none of these action languages support the capability of reasoning about noisy action durations, which is critical for multirobot planning toward sharing resources and constructing synergy at runtime.

Multirobot symbolic planning: Action languages, including PDDL, have been used for symbolic planning for a team of robots [10, 1, 21, 5, 32, 33]. However, they either do not model possible runtime conflicts (assuming that plans computed can be successfully executed to the end without any interruptions) [21, 5] or aim at resolving conflicts *locally* at runtime [10, 1, 32, 33]. As an example of locally resolving conflicts, two robots that compete for a narrow corridor can "negotiate" to make sure one robot gives way to the other [32]. Such conflict-resolving actions can be very expensive in practice, yielding locally optimal solutions. We model noisy action durations, as one of the factors that cause runtime conflicts, and avoid such conflicts (in probability) at planning time.

Multirobot scheduling: A multirobot scheduling problem's input includes a set of robots and a set of tasks. The output is a schedule that is for each task an allocation of one or more time intervals to one or more robots [4, 37, 7]. Recent work on multirobot scheduling further considers temporal uncertainty (in a multirobot navigation task) [3]. However, scheduling algorithms generally do not reason about actions' preconditions and effects, and hence cannot be used for generating action sequences in complex domains that require reasoning about actions.

Multirobot probabilistic planning: Contemporaneously with 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 [26, 27], planning under temporal uncertainty [14, 34], incorporating temporal logic into navigation task planning [8], and planning for multirobot systems using a single (PO)MDP [16], multiple (PO)MDPs [35], and DEC-POMDPs [2]. Such algorithms are good at handling nondeterministic 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. Therefore, symbolic planning, c.f., (PO)MDP-based, is more suitable to problems where there are many potential goals and human-interpretable plans are required.

Multirobot motion planning: Existing work has investigated the problem of multirobot concurrent task assignment and motion (trajectory) planning [29, 25]. 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. Although such motion planning algorithms are complimentary to our multirobot symbolic task planning algorithm, their methods are applicable to problems that require only navigation actions and they do not model noisy action durations (assuming no runtime delays).

Generally, there is a growing body of work on mixing symbolic and probabilistic techniques in reasoning and planning research. This paper introduces probabilistic temporal uncertainty (modeling noisy action durations) into the multirobot setting of symbolic planning, and for the first time, focuses on the problem of multirobot symbolic planning under temporal uncertainty.

3. DEFINITION OF MSPTU PROBLEMS

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 [12]. We assume the robots are homogeneous and the tasks are not transferable. The performance of an MSPTU algorithm is evaluated by episode and the end of an episode is identified by the time of the slowest robot finishing its task. A *central controller* runs the 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.

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 i^{th} 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 i^{th} 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 $\mathcal{D}$ includes the environmental information that does not change over time. For instance, two rooms being directly accessible to each other should be included in $\mathcal{D}$ (whereas through-door accessibility should not, because it can be changed by robot actions). 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"*. 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 (examples in § 5.1).

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, at the planning phase, 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.

4. IIDP ALGORITHM

Multirobot planning requires the modeling of *joint* actions that each correspond to a vector of actions, one for each robot, and *asynchronous* action executions at runtime. The exponentially increasing number of joint actions and possible interdependencies of concurrent actions make optimal multirobot planning NP-hard. The complexity of multirobot planning is analyzed in [31]. When temporal uncertainty is further incorporated, the problem (MSPTU) becomes extremely difficult, even if the number of robots and the length of individual plans are within a reasonable range. In this section, we aim to provide a general solution to MSPTU problems that cannot be solved using existing methods.

Algorithm 1 shows our novel, iterative inter-dependent planning (IIDP) algorithm for MSPTU problems. This algorithm is inspired by *simulated annealing* search for approximating the global optimum of overall system utility [20].

> *Informally,* IIDP *iteratively computes and saves the conditional "optimal" plan for each robot given other robots' current plans. In each iteration, the discount of conflict penalty increases (from zero in the first iteration to one in the last iteration) and the discount of collaboration-failure penalty decreases (from infinite to one).*

The input of IIDP includes the robots' initial and goal states ($\mathcal{S}$ and $\mathcal{G}$) that can be described using an action language, such as BC in our case, and the number of other robots being considered while computing conditional plan cost (M). The intuition of having parameter M is that a robot might be interested in negotiating with only a subset of its teammates [2]. Θ is an important parameter that represents how many rounds of *negotiations* the robots can perform before finalizing their plans, where a negotiation means a robot updates its plan based on plans of (not necessarily all) its teammates. The value of Θ has a significant influence on the performance of IIDP and will be discussed later. The output is P^N, a set of N plans, one for each robot.

[2]There is big room for research in specifying this subset of teammates, which is not covered in this paper. For instance, in navigation tasks, it makes sense to consider the robots' spatial closeness.

Algorithm 1 IIDP: our algorithm for MSPTU problems

Input: $\mathcal{S}$, a set of N states, and, $\mathcal{G}$, a set of N goals ($N \geq 2$)
Input: M: number of other robots considered in conditional planning, $M < N$
Input: Θ: number of rounds of "negotiations", $\Theta \geq 0$
Output: $P^N : [p_1, p_2, \ldots, p_N]$
1: Initialize a plan *queue* of size M: Q^M, where $p_i \in Q^M,\ i \in \{1, 2, \ldots, M\}$
2: Initialize a plan *array* of size N: P^N, where $p_i \in P^N,\ i \in \{1, 2, \ldots, N\}$
3: **for each** $i \in \{0, 1, \ldots, \Theta\}$ **do**
4: $\quad \alpha = i/\Theta$, where we define $\alpha = 0$ when $\Theta = 0$
5: $\quad$ **for each** $j \in \{1, 2, \ldots, N\}$ **do**
6: $\quad\quad$ Dequeue from Q^M
7: $\quad\quad P^N(j) = \text{argmin}_{p'_j}\Big(\mathcal{C}(p'_j, Q^M, \alpha)\Big)$, where $s_j \xrightarrow{p'_j} g_j, s_j \in \mathcal{S}, g_j \in \mathcal{G}$
8: $\quad\quad$ Enqueue p_j into Q^M
9: $\quad$ **end for**
10: **end for**
11: **return** $P^N : [p_1, p_2, \cdots, p_N]$

We first initialize an empty plan queue (FIFO) of length M ($M < N$) that is used for saving the plans from M teammates. Then we enter a for-loop that has $\Theta + 1$ iterations (Lines 3-10), where α is a *negotiation depth* that incrementally grows by $1/\Theta$ in each iteration (Line 4). 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 (conflicts have no cost and collaboration failures have an infinite cost); when $\alpha = 1$, it considers its teammates as important as itself (costs are not discounted). In the inner for-loop (Lines 5-9), we compute a plan p_j for the j^{th} robot depending on existing plans of other robots, save it in P^N as $P^N(j)$, and enqueue this new plan to plan queue Q^M. Inter-dependent planning is conducted in Line 7, where we compute the optimal conditional plan for the j^{th} robot while minimizing the conditional plan cost given the current plans of its M teammates (saved in plan queue Q^M). The $s \xrightarrow{p} g$ symbol represents that plan p leads state transitions from initial state s to goal g. In the end of the program, a set of plans is returned as P^N.

It should be noted that in Line 7: the operation of $\mathcal{C}$ for computing inter-dependent plan cost requires the modeling of action preconditions, effects, costs, and noisy durations, which is highly domain dependent and hence its development is independent of IIDP. The operation of argmin requires a symbolic task planner for computing a sequence of actions while minimizing the overall plan cost.

Two configurations: simple and enhanced IIDPs.

When $\Theta = 0$, the robots compute plans independently, because we define $\alpha = 0$ when $\Theta = 0$ in Line 4. When $\Theta = 1$, there is only one round of negotiation and we call this configuration *simple*-IIDP (**S-IIDP**). The performance of S-IIDP is sensitive to the order of the robots being planned for, because a given robot considers all robots in front of it (in plan queue Q^M) but none of the robots after it. An extreme case is that the N^{th} robot's updated plan is not considered by any of its teammates. When $\Theta > 1$, there are multiple rounds of negotiations and we call this configuration *enhanced*-IIDP (**E-IIDP**).

IIDP has $O(\Theta \cdot N \cdot C)$ complexity where C is the complexity of a single inter-dependent planing operation ($\mathcal{C}$ in Line 7) and N is the number of robots. We do not discuss the complexity of $\mathcal{C}$, because the development of $\mathcal{C}$ algorithms is independent of IIDP. Section 5 presents a novel algorithm for $\mathcal{C}$ within the context of multirobot navigation tasks.

Although the output of Algorithm 1 includes plans for all robots,

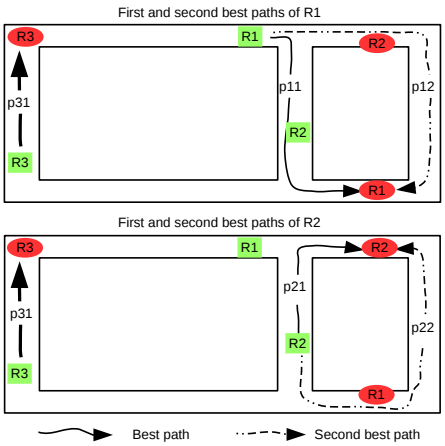


Figure 2: In this example, the overall cost of $p12$ and $p21$ is smaller than the cost of $p11$ and $p22$: $C(p12)+C(p21) < C(p11)+p(22)$. E-IIDP always suggests $p12$ and $p21$ (optimal) despite the planning order, outperforming independent planning and (Brute-force) S-IIDP.

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.

An illustrative example of E-IIDP.

Figure 2 shows a three-robot symbolic planning problem where the robots' start and end points are marked with green rectangles and red ellipses respectively. The two subfigures show the best two plans of robots $R1$ and $R2$; $R3$ has only one dominant plan. [3]

In case of no communication among the robots (Θ=0), $R1$ takes $p11$ and $R2$ takes $p21$, causing a collision with a large probability. S-IIDP is more competitive: in an arbitrary order, we compute a conditional plan for each robot while considering plans of all other robots. In case of planning order $R2$-$R1$-$R3$ (S-IIDP is sensitive to ordering), S-IIDP suggests $p11$ and $p22$ to $R1$ and $R2$, which is suboptimal.

Using E-IIDP with M=2, the initial plans are $p11$, $p21$ and $p31$ after the first iteration ($i=0$). Given a reasonably large Θ, *despite the planning order*, $R1$ will switch to $p12$ when α is increased to some level, and then the plans remain to the end. It should be noted that, in this example, S-IIDP is guaranteed to find the optimal solution *only if it tries all possible orderings* (Brute-force S-IIDP), which has $O(N! \cdot N \cdot C)$ complexity for the general case, whereas E-IIDP has $O(\Theta \cdot N \cdot C)$ complexity. As a result, E-IIDP has much lower complexity than Brute-force S-IIDP in this example: E-IIDP $= O$(Brute-force S-IIDP), while both produce optimal solutions. Although in this case both Brute-force S-IIDP and E-IIDP do find optimal solutions, it is not the case that either is always optimal. Indeed, it is possible to generate examples of each outperforming the other. Such examples are presented in **Appendix** A that is available in an extended version of this paper.[4]

[3]Symbolic plans are simply represented as navigation trajectories for the sake of explanation. We assume collisions occur (in probability) only if both robots are moving.

[4]The extended version of this paper is hosted at: `www.cs.utexas.edu/~pstone/Papers/bib2html/b2hd-AAMAS17-Zhang.html`

5. AN INSTANTIATION OF IIDP

In this section, we instantiate IIDP using a multirobot navigation task. Symbolic task planning techniques are needed because robots need to reason about between-room accessibilities and plan to open doors as needed. The single-robot version of this domain (without modeling noisy action durations) has been studied in existing research [17, 36]. In this section, we present our symbolic planner, *shifted-Poisson* distributions for accumulating temporal uncertainty over actions, and a novel algorithm for computing conditional plan cost under temporal uncertainty.

5.1 Single-robot symbolic planning using BC

We use action language BC [24] for symbolic planning in this work because it can formalize defaults and recursively defined action effects (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 not restricted to specific action languages or symbolic planners. We adapt existing formulations of BC-based, single-robot navigation tasks [17, 36] for multirobot settings. For instance, we use the following rules to define the ownership between rooms and doors:

$hasdoor(r1, d1).\ hasdoor(r1, d2).\ hasdoor(r2, d2).\ \cdots$
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. In implementation, to model the progress of navigation actions (*approach*, in our case), we discretize distance by representing each corridor using a set of grid cells. Accordingly, each *approach* action is replaced by a sequence of actions that lead the robot to follow waypoints. We use CLINGO4 for solving BC programs [11].

5.2 Modeling noisy action durations

In single-robot systems, following the plan generated by a symbolic planner, such as our BC-based planner, a robot can execute actions to optimally achieve the goal. When multiple robots share an environment, their plans might interact such that their independently-computed optimal plans become suboptimal at runtime. In order to

leverage such interactions toward sharing resources and constructing action synergy, it is necessary to model the temporal uncertainty (in noisy action durations) that propagates over actions.

This subsection presents a novel model for representing and reasoning about temporal uncertainty in the noisy durations of navigation actions. This representation is used for not only modeling individual actions' noisy durations but also accumulating the uncertainty over a sequence of actions. In this paper, we consider only the temporal uncertainty from navigation action *approach*(D). Deriving the probability density function (PDF) of *approach*(D) builds on the following assumptions:

1. Unless explicitly delayed, the robots move at constant 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. [5]

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:

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

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: $Z \sim Poisson(\lambda_1+\lambda_2)$ [22].

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 $\hat{f}(k',\lambda')$. 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 *shifted* Poisson distribution with PDF:

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

Figure 3(a) 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 shown in Figure 3(b) suggest that a shifted Poisson distribution can well represent noisy durations of navigation actions (with parameters properly set).

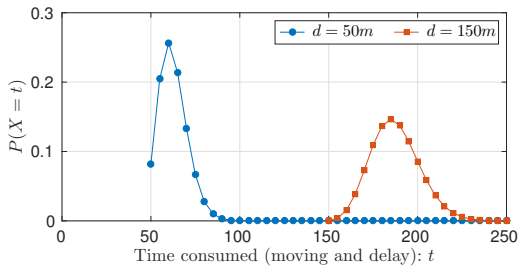
We further remove the parameter of λ' and substitute d/v with t^{act}, and use $Dist(t^{act},\lambda')$ to represent a distribution over possible lengths of completion time. Modeling noisy action durations in this way paves the way to further investigating how uncertainty is accumulated over plans that include a sequence of actions. For instance, $t^{act} = 50$ and $\lambda' = 2.5$ correspond to the (blue) circle-mark curve in Figure 3(a). 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).

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

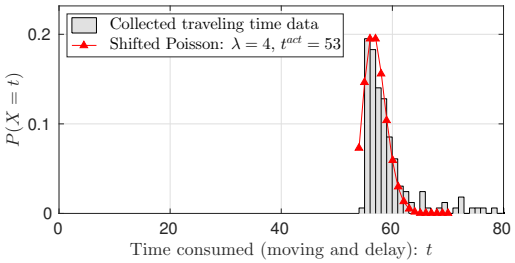
According to Proposition 1, the time consumed by executing the first K actions in plan p follows a distribution of:

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

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, as an illustrative example and for the purpose of evaluating IIDP, simply presents a concise PDF representation for navigation actions.



(a) Noisy durations of navigation actions in simulation.



(b) Noisy durations of navigation actions on a real robot.

Figure 3: (a) PDFs of two *shifted* Poisson distributions used for modeling the noisy durations of navigation actions: $v = 1$, $\delta = 5$, and $\lambda = 0.05$. A longer distance brings more uncertainty; and (b) A real robot navigates in a reasonably busy corridor ($28m$) for 164 times. The shifted Poisson well models the action's noisy durations.

5.3 Computing inter-dependent plan cost: $\mathcal{C}$

In a two-robot system that includes robots R and R', p and p' are robots' extended plans. The *inter-dependent 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 inter-dependent plan costs. We first compute the probability of robot R''s navigation action a' overlapping p's nav-

[5] For example, such delays can be caused by forcing the robot to stop and say "excuse me" as is done by CoBots [30].

igation action a over time (parameter λ omitted from PDFs), while the overlapping in space is handled by the symbolic task planner:

$$Pr^{ovlp}(a, a') = 1 - \int_0^\infty \int_{t_2}^\infty f_1^s(t_1) f_2^c(t_2)\, dt_1\, dt_2 - \int_0^\infty \int_{t_1}^\infty f_1^c(t_1) f_2^s(t_2)\, dt_2\, dt_1 \quad (5)$$

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. [6]

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).

$$t^{wait}(a, a') = \int_0^\infty \int_{t_2}^\infty (t_1 - t_2) f_1^c(t_1) f_2^s(t_2)\, dt_1\, dt_2 \quad (6)$$

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:

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

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

Algorithm 2 Computing inter-dependent plan cost (navigation)

Input: Plan p', whose cost will be evaluated
Input: Plan set P, on which the cost of p' is dependent
Input: α: negotiation depth
Input: (optional configuration parameters) μ : collision cost, ω : *waitforopen*(D) failure cost, and ρ : value of time
Output: C': overall cost of plan p'

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

Algorithm 2 presents our algorithm for computing inter-dependent plan cost (in our navigation domain). While computing the cost of *waitforopen*(D), we need to consider both the cases that have synergy and those that failed (Lines 6-7). 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.

[6]In implementation, the integrals are replaced by summation operations, because action completions only happen at specific time instances (e.g., Figure 3).

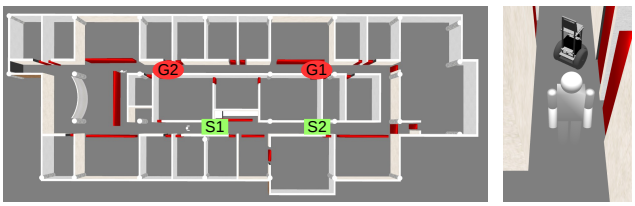


Figure 4: GAZEBO simulation environment (and a picture of a human walker blocking a robot).

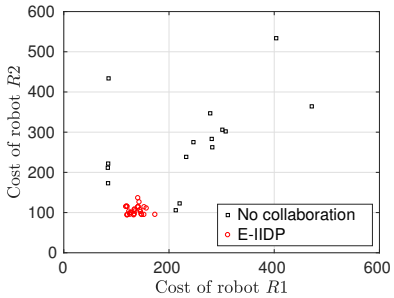


Figure 5: Costs of robots $R1$ and $R2$ in 45 trials collected using GAZEBO simulation environment.

6. EXPERIMENTS

We have implemented IIDP (specifically, its multirobot navigation instantiation) both in simulation and on robots. Simulation experiments were conducted using a realistic multirobot simulation environment (GAZEBO [23]) and a much faster abstract simulator that does not have an interface for visualization or a physics engine for simulating collision consequences. Noisy action durations in the abstract simulator are sampled from predefined distributions. Experiments were conducted to investigate how different values of Θ (S-IIDP vs. E-IIDP) affect the performance in reducing overall cost, to evaluate the necessity of modeling noisy action durations using our probabilistic model, and to study the performance of IIDPs in systems that include varying numbers of robots.

Gazebo simulation (No collaboration vs. E-IIDP).

Figure 4 (**left**) shows our GAZEBO testing environment. We add human walkers (**right**) to simulate the process of walking people causing delays to robot navigation actions. The floormap is divided into grid cells and taking a symbolic action (to one of the four directions) enables the robot to move to one of the nearby cells given no obstacles. Two robots need to navigate from their initial positions (green rectangles) to their goal positions (red ellipses). The two robots start at the same time, and we record the completion time for each of the robots. The performance is evaluated based on the robots' overall completion time.

Table 1: Average cost consumed (time) and standard deviation from GAZEBO simulation experiments (reported in Figure 5).

	E-IIDP	No collaboration
Robot-1	136.34 (13.18)	238.36 (117.04)
Robot-2	104.85 (10.74)	278.72 (112.55)
Average	120.60	258.54

Experiments in GAZEBO were conducted to visualize and validate the whole process of multirobot plan generation and execution, and to compare IIDP (enhanced configuration) to a baseline that computes plans for the robots independently (no collabora-

tion). The results in the form of execution costs of two robots are shown in Figure 5. We can see a cluster of red circles in the bottom-left, which indicates the E-IIDP algorithm reduces the overall plan cost. The blue squares on the left, for instance, correspond to the trials where robot $R2$ avoids $R1$ by taking a big detour (locally optimal solution). Table 1 shows the averages of the same set of results collected from GAZEBO. Considering both robots, E-IIDP ($M=1$, $\Theta=2$) significantly reduces the average completion time from more than 250 seconds to about 120 seconds.

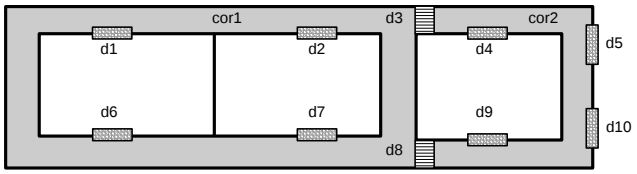


Figure 6: Abstract simulation environment, where action durations are generated by sampling from pre-specified distributions.

Abstract simulation (S-IIDP vs. E-IIDP).

In order to run a lot more trials, we use an 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. [7] Figure 6 shows the domain map.

The first set of experiments in abstract simulation was conducted to evaluate how the number of rounds of negotiations (Θ in Algorithm 1) affects the overall cost. Figure 7 reports the results: $\Theta=0$ means no collaborations between robots (baseline); $\Theta = 1$ and $\Theta = 2$ correspond to S-IIDP and E-IIDP respectively. Each data point corresponds to results from 50 trials (the same for the following results unless stated otherwise). For instance, when robot $R1$ is delayed by 5 time units, we see E-IIDP reduces the overall cost from more than 80 to lower than 50, and enables $R1$ and $R2$ to share door-opening actions. When one robot starts much earlier than the other (two ends of these curves), the overall costs are all about 45 no matter whether collaborations are enabled or not, because they can hardly cause collisions or share doors. Comparing the triangle and square curves in both subfigures, we find that E-IIDP enables more action synergies than S-IIDP (via sharing door-opening actions), even when the overall cost reduction introduced by such synergies is small.

Abstract simulation (with/without uncertainty).

Our next set of experiments evaluate the need for modeling temporal uncertainty, where the baseline does not model the noise in action durations (optimistically assuming no delays in navigation actions). $R1$ needs to move from $d3$ in corridor $cor1$ to door $d4$, so the best plan for $R1$ is to open and go through door $d3$ in any case. The head start of $R1$ varies in a relatively small range (± 6).

Figure 8 reports the results of these experiments. When $R1$ has a head start of -1, the overall cost of our E-IIDP 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. Without modeling

[7] If two robots try to pass each other, there is a significant risk that they will bump into each other and become entangled. In contrast, at least in our environment, we find that most people give way to the robots by standing close to the wall.

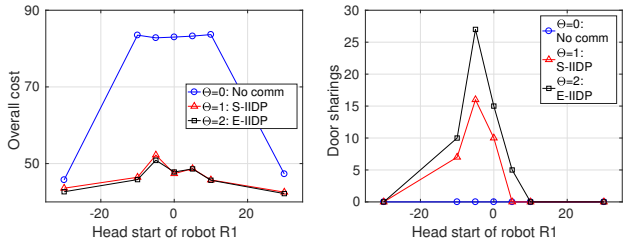


Figure 7: Planning for a two-robot system (evaluating Θ): $R1$ and $R2$ need to move from $d2$ to $d9$ and from $d7$ to $d4$ respectively.

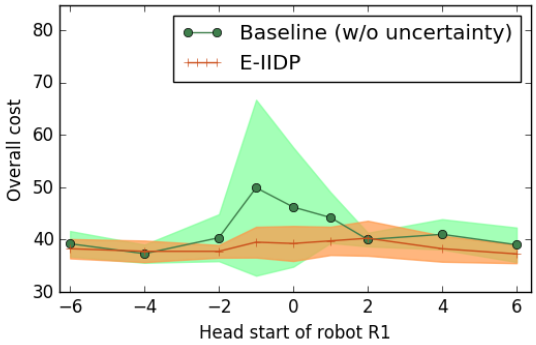


Figure 8: Planning for a two-robot system (evaluating the need of modeling temporal uncertainty): $R1$ and $R2$ need to move from $d3$ ($cor1$ side) to $d4$ and from $d7$ to $d5$ respectively.

temporal uncertainty (baseline), $R2$ is not aware of the risk of being delayed while moving upward. As a consequence, $R2$ will be moving to door $d3$ (hoping to follow $R1$ through door $d3$), even if it is in a risky situation that a single delay on the way will make it too late to catch up with $R1$'s door-opening action. The big variance in overall cost for the baseline corresponds to the fact that the trials where robot $R2$ succeeds in following $R1$ through the door and the trials where it fails produce very different overall costs. E-IIDP models the noisy action durations and enables $R2$ to dynamically evaluate the uncertainty from its teammate and itself, and is able to balance the risk and potential benefit to select the best path.

Abstract simulation (three-robot experiments).

In a team that includes more than two robots, IIDP has the option to consider only a *subset* of its teammates in conditional planning (specified by M in Algorithm 1). This set of experiments was conducted in a team of three robots to evaluate how the value of M affects the performance of IIDP. Accordingly, we adjust the value of Θ (S-IIDP vs. E-IIDP) and the value of M. Robots $R2$ and $R3$ have different head starts before $R1$'s plan execution. The subfigures of Figure 9 report the results of nine different head start combinations. In each subfigure, the x-axis corresponds to one of the four IIDP configurations, and the y-axis corresponds to the overall cost. We do not see significant differences over the four IIDP configurations in most head start combinations. This corresponds to our expectation that, when the robots' plans do not have (much) overlap in time, their plan executions do not affect each other and it is unlikely to have collisions or construct synergy. In the middle-left and bottom-middle subfigures, we see considering two other robots (instead of one) significantly reduces the cost of robot $R3$ and the overall cost.

Table 2 shows the performance of the four IIDP configurations. The reduction of average cost by considering plans of all other robots is significant, regardless of Θ's value: p-value=0.03 when

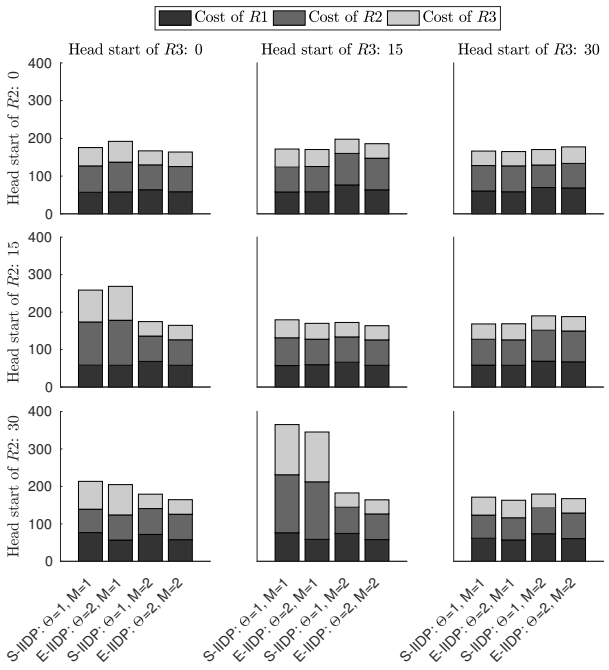


Figure 9: Planning for a three-robot system (overall cost under four configurations of IIDP): $R1$, $R2$ and $R3$ need to move from $d1$ to $d9$, from $d6$ to $d4$, and from $d10$ to $d8$ ($cor1$ side) respectively.

Table 2: Mean and standard deviation values of the four configurations in Figure 9. Given $M = 2$, the average overall cost using E-IIDP is significantly different from that of S-IIDP (v-value=0.0128).

	Number of teammates considered in conditional planning	
	$M = 1$	$M = 2$
S-IIDP : $\Theta = 1$	207.81 (66.24)	179.28 (9.81)
E-IIDP : $\Theta = 2$	205.35 (62.25)	***171.03 (9.99)***

$\Theta = 1$, and p-value=0.02 when $\Theta = 2$. Given all other robots are considered in conditional planning, E-IIDP performs significantly better than S-IIDP (bold font). However, when only one other robot is considered, we do not see a significant difference between E-IIDP and S-IIDP (the left two columns). Therefore, E-IIDP with $M = 2$ performs significantly better than all three other configurations.

Robot trial.

Collecting statistical results using multiple robots on navigation tasks can be difficult in practice, because the robots sometimes take a very long time to finish a trial, especially when the collaborations are not successful, and robot collisions can cause physical damage to the robot platforms and sometimes to the environment. However, in order to demonstrate that our methods can be used to enable two real robots to collaborate by constructing an action synergy, we present an illustrative (successful) trial in the real world.

We have implemented the two configurations of IIDP and all actions formalized in action language $\mathcal{BC}$, including *approach*(D), *opendoor*(D), *cross*(D), and *waitforopen*(D), on a team of real robots. Runtime communications are manually conducted for now. Figure 10 shows the occupancy-grid map, where robot $R1$ and $R2$ are required to move from $S1$ to $G1$ and from $S2$ to $G2$ respectively. The floor is separated into two areas by two doors in the middle, $D1$ and $D2$. Without collaboration, $R1$ and $R2$ will choose to ask help from humans to open and go through $D1$ and $D2$ respectively, which is not a globally optimal solution.

E-IIDP models the possible delays in navigation actions, enabling $R2$ to balance the potential benefit of following $R1$ through door $D1$ and the risk of not being able to catch up $R1$'s door-opening action. As a result, constructive action synergy is realized: while $R1$ is taking the expensive "call for open" action, $R2$ moves to $D1$ and waits until $R1$ "opens" $D1$ (with human help). This strategy is better than the one suggested by the no-collaboration baseline and produces more reliable collaborations between robots by modeling the temporal uncertainty. Figure 11 shows a picture of $R2$ (robot on the right) waiting to follow $R1$ (robot on the left) through door $D1$, using E-IIDP. A video of this trial is available at:
`https://youtu.be/ADbH3sppLHQ`

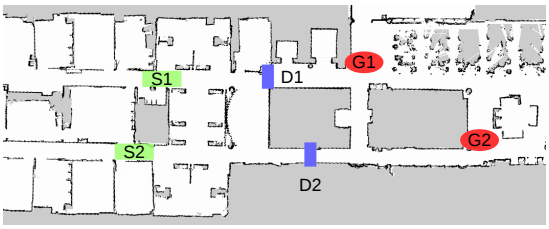


Figure 10: Floor map of the real-world environment.

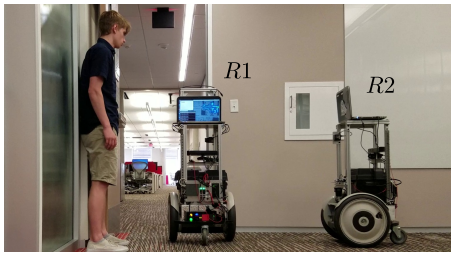


Figure 11: Using E-IIDP, two robots construct action synergy by sharing a door-opening action: robot $R1$ asks for help from a human for opening the door and is executing the *gothrough* action, while robot $R2$ is waiting to follow $R1$ through the door.

7. CONCLUSIONS

We introduce the multirobot symbolic planning under temporal uncertainty (MSPTU) problem, and develop a novel, iterative inter-dependent planning (IIDP) algorithm inspired by simulated annealing. IIDP has two configurations, *simple*-IIDP (S-IIDP) and *enhanced*-IIDP (E-IIDP). We instantiate IIDP with a multirobot navigation problem, where we introduce *shifted Poisson* distributions and present a novel algorithm for computing conditional plan cost. In experiments, we see E-IIDP brings significant improvements against baselines where robots do not coordinate their plans, or coordinate but do not model temporal uncertainty, and E-IIDP enables more collaborations, compared to S-IIDP.

Acknowledgments

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, IIS-1637736, IIS-1651089), ONR (21C184-01), AFOSR (FA9550-14-1-0087), Raytheon, Toyota, AT&T, and Lockheed Martin. 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

[1] R. Alur, S. Moarref, and U. Topcu. Counter-strategy guided refinement of gr (1) temporal logic specifications. In *Formal Methods in Computer-Aided Design (FMCAD), 2013*, pages 26–33. IEEE, 2013.

[2] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling. Planning for decentralized control of multiple robots under uncertainty. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1241–1248. IEEE, 2015.

[3] J. Brooks, E. Reed, A. Gruver, and J. C. Boerkoel Jr. Robustness in probabilistic temporal planning. In *National Conference on Artificial Intelligence (AAAI)*, 2015.

[4] P. Brucker. *Scheduling algorithms*. Springer, 2007.

[5] J. Buehler and M. Pagnucco. A framework for task planning in heterogeneous multi robot systems based on robot capabilities. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[6] X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, and J. Xie. Developing high-level cognitive functions for service robots. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 989–996, 2010.

[7] B. Coltin and M. Veloso. Scheduling for transfers in pickup and delivery problems with very large neighborhood search. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[8] J. P. Fentanes, B. Lacerda, T. Krajnik, N. Hawes, and M. 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.

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

[10] I. Filippidis, D. V. Dimarogonas, and K. J. Kyriakopoulos. Decentralized multi-agent control from local ltl specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 6235–6240. IEEE, 2012.

[11] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014.

[12] B. P. Gerkey and M. 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.

[13] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. E. Smith, S. Ying, and D. Weld. Pddl-the planning domain definition language. 1998.

[14] X. Guo and O. Hernández-Lerma. *Continuous-time Markov decision processes*. Springer, 2009.

[15] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrová, J. Young, J. Wyatt, D. Hebesberger, T. Körtner, et al. The strands project: Long-term autonomy in everyday environments. *IEEE Robotics and Automation Magazine*, 2016.

[16] P. Khandelwal, S. Barrett, and P. 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.

[17] P. Khandelwal, F. Yang, M. Leonetti, V. Lifschitz, and P. Stone. Planning in Action Language $\mathcal{BC}$ while Learning Action Costs for Mobile Robots. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[18] P. Khandelwal, S. Zhang, J. Sinapov, M. Leonetti, J. Thomason, F. Yang, I. Gori, M. Svetlik, P. Khante, V. Lifschitz, et al. Bwibots: A platform for bridging the gap between ai and human–robot interaction research. *The International Journal of Robotics Research*, 2017.

[19] B. Kim, C. Chacha, and J. Shah. Inferring robot task plans from human team meetings: A generative modeling approach with logic-based prior. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[21] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 855–862. IEEE, 2013.

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

[23] N. Koenig and A. 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.

[24] J. Lee, V. Lifschitz, and F. Yang. Action language bc: Preliminary report. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 983–989. AAAI Press, 2013.

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

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

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

[28] K. Talamadupula, J. Benton, P. W. Schermerhorn, S. Kambhampati, and M. Scheutz. Integrating a closed world planner with an open world robot: A case study. In *AAAI*, 2010.

[29] M. Turpin, N. Michael, and V. Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112, 2014.

[30] M. M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal. CoBots: Robust symbiotic autonomous mobile service robots. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[31] G. Wagner and H. Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1 – 24, 2015.

[32] K. W. Wong and H. Kress-Gazit. Let's talk: Autonomous conflict resolution for robots carrying out individual high-level tasks in a shared workspace. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 339–345. IEEE, 2015.

[33] K. W. Wong and H. Kress-Gazit. Need-based coordination

for decentralized high-level robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

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

[35] S. Zhang, M. Sridharan, and C. Washington. Active visual planning for mobile robot teams using hierarchical POMDPs. *IEEE Transactions on Robotics*, 29(4):975–985, 2013.

[36] S. Zhang, F. Yang, P. Khandelwal, and P. Stone. Mobile robot planning using action language bc with an abstraction hierarchy. In *Proceedings of the 13th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR)*, Lexington, KY, USA, September 2015.

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

Appendix *A*

We present examples as evidences that neither Brute-force Simple IIDP (Brute-force S-IIDP) nor Enhanced IIDP (E-IIDP) always finds the optimal plan. It should be noted that (similar to Figure 2 in the main paper) symbolic plans are simply represented as navigation trajectories for the sake of explanation.

We make the following assumptions in the examples:

1. Robots collide at a node if they arrive at the same time step (even when they are moving in the same direction).
2. Costs of collisions are cumulative, i.e. the cost of two collisions is higher than the cost of one collision. This allows the algorithm to minimize the probability of collisions, if not possible to avoid them completely.
3. Robot i starts at S_i and plans to go to G_i.

Figure 12 shows an example of E-IIDP outperforms Brute-force S-IIDP (**Left**) and an example of Brute-force S-IIDP outperforms E-IIDP (**Right**).

In the left diagram of Figure 1, the individual plans can possibly "collide" at nodes A and D. E-IIDP successfully avoids both collisions by suggesting plan (S_1->A->C->G_1, S_2->B->D->G_2), producing the optimal solution. Brute-force S-IIDP will try two orderings: 1->2 and 2->1. S-IIDP with planning order 1->2 generates the plan (S_1->A->D->G_1, S_2->B->D->G_2), which results in a collision at D. Similarly, S-IIDP with the planning order 2->1 generates a plan that causes a collision at A. In this example, E-IIDP outperforms Brute-force S-IIDP.

The right diagram shows a three-robot example where S-IIDP finds the optimal plan and E-IIDP fails. Brute-force S-IIDP is able to find the optimal plan (with ordering 1->2->3): (S_1->A->D->G_1, S_2->B->E->G_2, S_3->C->F->G_3). E-IIDP produces a locally optimal solution (S_1->A->D->G_1, S_2->A->D->G_2, S_3->B->E->G_3), where robots 1 and 2 cannot switch to collision-free plans given the plan of robot 3, while robot 3 has no incentive to change plan. In this example, Brute-force S-IIDP outperforms E-IIDP.

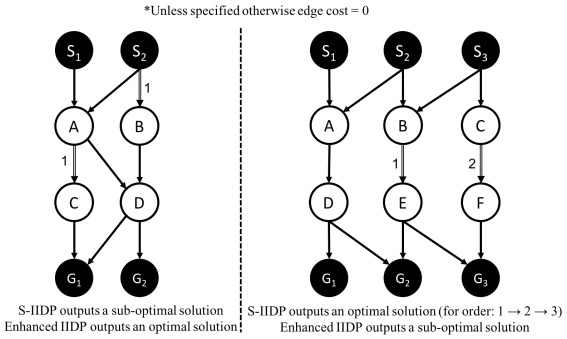


Figure 12: Two examples with avoiding collisions. The left diagram shows a case where E-IIDP outperforms Brute-force S-IIDP. The right diagram shows a three-robot example where S-IIDP finds the optimal plan and E-IIDP fails.

Figure 13 shows an example where robots can realize action synergy by sharing door-opening actions. In the main paper, results reported in Figure 7 show that E-IIDP enables more door-sharing behaviors than S-IIDP and E-IIDP has much lower complexity than Brute-force S-IIDP. In this example, we present a situation where Brute-force S-IIDP (with orderings 3->2->1 or 3->1->2) outperforms E-IIDP. Detailed information about the plans suggested by Brute-force S-IIDP and E-IIDP has been embedded in the figure.

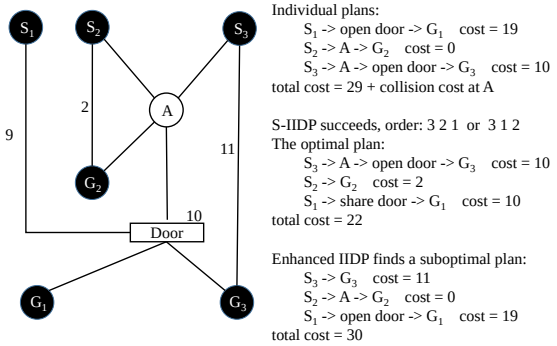


Figure 13: An example in the context of sharing doors. Brute-force S-IIDP (with orderings 3->2->1 or 3->1->2) outperforms E-IIDP in this case by planning robots 1 and 3 to share the door.