



Multi-robot planning with conflicts and synergies

Yuqian Jiang¹ · Harel Yedidsion¹  · Shiqi Zhang² · Guni Sharon³ · Peter Stone¹

Received: 3 March 2019 / Accepted: 15 March 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Multi-robot planning (MRP) 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 MRP 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*). Optimally solving MRP problems is hard as it is a generalization of the single agent planning domain which is known to be NP-hard, and frequently requires considering the states of all the robots, resulting in an exponentially growing joint state and action space. In many MRP domains, robots encounter situations where they have conflicting needs for constrained resources, or where they can take advantage of what each other is doing to form synergies. In this article, we formulate the problem of multi-robot planning with conflicts and synergies (MRPCS), and develop a multi-robot planning framework, called *iterative inter-dependent planning* (IIDP), for representing and solving MRPCS problems. Within the IIDP framework, we develop the algorithms of *increasing dependency* and *best alternative* which exhibit different trade-offs between plan quality and computational efficiency. Extensive experiments covering the suggested algorithms have been performed using both an abstract-domain simulator, where we can automatically generate a variety of domain configurations, and a practical MRPCS instantiation that focuses on multi-robot navigation. In the navigation domain, we model plan costs with temporal uncertainty, and present a novel *shifted-Poisson* distribution for accumulating temporal uncertainty over actions. In comparison to baseline approaches, our algorithms produce significant reductions in overall plan cost, while avoiding search in the joint state space. In addition, we present a complete demonstration of the implementation of the model on a team of real robots.

Keywords Multi-robot planning · Planning under temporal uncertainty · Intelligent mobile robotics

1 Introduction

Multi-agent planning has become increasingly important as more and more systems have many autonomous agents interacting in the same environment. Autonomous vehicles and service robots in buildings are just a couple of examples. In some multi-agent systems, each agent has its own goal and planning knowledge, and generates plans to achieve its goal while maximizing its own utility. Planning techniques such as *symbolic planning* allow an agent 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

✉ Harel Yedidsion
harel@cs.utexas.edu

Yuqian Jiang
jiangyuqian@utexas.edu

Shiqi Zhang
szhang@cs.binghamton.edu

Guni Sharon
guni@tamu.edu

Peter Stone
pstone@cs.utexas.edu

¹ Department of Computer Science, The University of Texas, Austin, USA

² Department of Computer Science, The State University of New York, Binghamton, USA

³ Computer Science & Engineering, Texas A&M University, Texas, USA



Fig. 1 Three of our Segway-based mobile robot platforms

process, agents can compute optimal plans that maximize utility (or minimize cost).

When multiple robots share a physical environment (such as our Segway-based robots, BWIBots Khandelwal et al. (2017), that are shown in Fig. 1), it is necessary to model how their plans interact with each other, to avoid conflicts and to construct action synergies as much as possible.

The agents' 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. Figure 2 shows an example of a conflict, where two robots need to go from $S1$ to $G1$ and $S2$ to $G2$ respectively. If the robots act according to independently-computed optimal plans, they will block each other in a narrow corridor, causing a conflict.

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. This scenario is demonstrated in Fig. 3 which shows the occupancy-grid map, where robot r_1 and r_2 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, r_1 and r_2 will plan to perform the expensive door opening procedure independently of each other and go through $D1$ and $D2$ respectively. The globally optimal solution is, while r_1 is taking the expensive "call for open" action, r_2 moves to $D1$ and waits until r_1 "opens" $D1$ (with human help).

One of the key challenges to planning for such resource sharing and leveraging synergy is the inherent temporal uncertainty in the duration of robots' actions at runtime, which is largely overlooked in existing research. While such conflicts can be resolved locally at runtime [e.g., two robots detecting a conflict at runtime can "negotiate" to decide who gives way to the other Wong and Kress-Gazit (2015)], we

argue a better solution is to avoid such conflicts at planning time.

The contributions of this article include:¹

- The formulation of the multi-robot planning with conflicts and synergies (MRPCS) problem;
- A novel, Iterative Inter-Dependent Planning (IIDP) framework for coordinating single-robot plans, which is the main contribution of this article;
- Two algorithms that implement the IIDP framework for solving MRPCS problems;
- A model of temporal uncertainty in MRPCS problems;
- Systematic evaluations of the framework and algorithms, including demonstrations on real robots.

IIDP is generally applicable to MRPCS problems via utilizing single robot planners. However, IIDP is not restricted to specific planners, forms of noisy action durations, or application domains. The IIDP framework provides a high-level coordination protocol for a team of planning agents, to communicate their plans and produce plans with overall lower costs than independently generated plans, while efficiently using constrained resources.

The general structure of an IIDP algorithm suggests that each robot iteratively computes and saves the conditionally optimal plan given other robots' current plans. Each robot then exchanges plans with the others and re-plans accordingly. The iterative "negotiation" and re-planning process continues until the robots either reach an agreement or use up the maximum number of iterations.

While the general IIDP framework can accommodate a diverse set of algorithms varying in the decision process they employ to re-plan, we introduce and focus on a study of two specific IIDP algorithms in this article. The proposed IIDP algorithms have different trade-offs between computational complexity and plan quality. The first is the BEST ALTERNATIVE (BA) algorithm, which aims to iteratively find the agent with the best alternative plans, and have this agent change its plan. The second algorithm is the INCREASING DEPENDENCY (ID) algorithm, which increasingly considers the cost of interactions in a finite number of iterations. The agents start from completely ignoring interactions with other agents in the first iteration, and end with full consideration of interactions in the last iteration. Intuitively, this is a multi-round negotiation process, where the agents proceed from independent planning to full consideration of conflict costs and synergy benefits.

¹ This article is based upon a previous conference publication (Zhang et al. 2017). It extends the conference paper in formalizing a framework for multi-robot planning, proposing a novel algorithm, and testing the algorithms on larger scale problems of up to 50 agents as opposed to 3. None of the material (including the conference paper) has appeared in any other journal submission.

Fig. 2 An example showing a conflict between the individually optimal plans of two robots

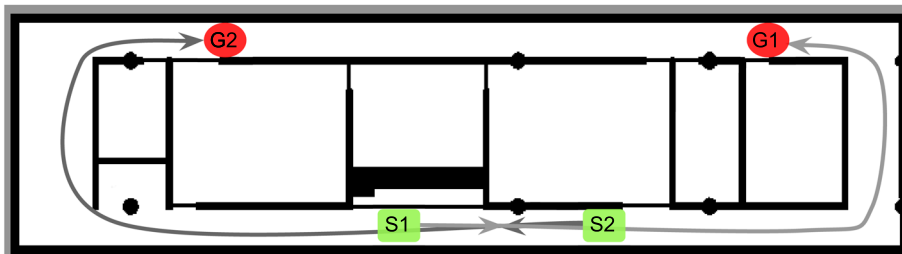
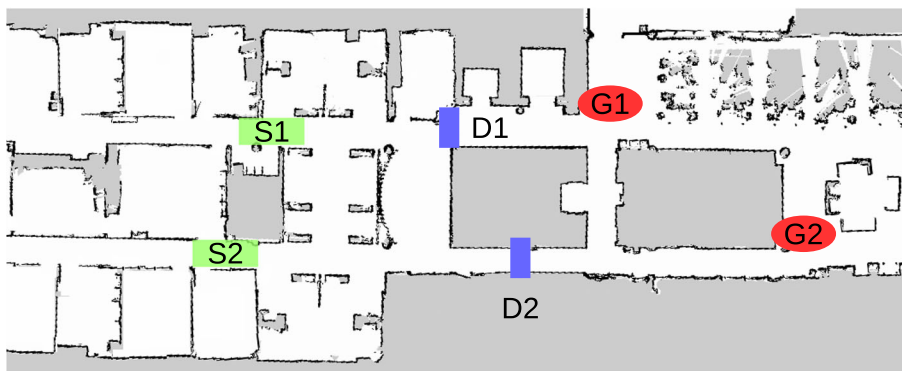


Fig. 3 Floor map showing a potential synergy between two robots going from S1 to G1 and S2 to G2 respectively. D1 and D2 are closed doors



The most important property of IIDP is that it does not increase an MRPCS problem's dimensionality beyond that of a single agent and never considers the joint-agent state-action space, while still being able to produce good quality solutions (Sect. 4). IIDP is named as being "inter-dependent" because an important step in the implementation of IIDP algorithms 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). While this inter-dependent planning algorithm is an important component of any IIDP realizations, the IIDP framework is general and is not restricted to any specific inter-dependent planning algorithm.

As a secondary contribution of this article, we model and solve a specific MRPCS problem, namely *multi-robot navigation*, using the IIDP framework. The aim of applying MRPCS to this specific domain includes illustrating the whole process of instantiating IIDP, demonstrating the need of modeling temporal uncertainty in MRPCS problems, comparing the two new IIDP algorithms (BA and ID), and evaluating the performance of IIDP in a real-world problem.

In both the instantiated real world problem and the abstract virtual domain, we observe that our IIDP algorithms reduce the overall plan cost compared to baseline algorithms. In the experiments with real robots, the use of the IIDP framework enables a team of robots to avoid going into the same corridor at planning time, and leverages action synergy by sharing door-opening actions.

Although we primarily use a team of indoor mobile robots for demonstration and evaluation purposes in this article, the developed framework and algorithms are general enough to

be applicable to a variety of other domains. For instance, one setting where multi-robot interaction and coordination are crucial, is the autonomous vehicle planning domain (Dresner and Stone 2008). Here conflicts can manifest as traffic jams while synergies can be formed by platooning, which in turn can yield energy saving, lower congestion, and improved safety (Tsugawa and Kato 2010).

The remainder of this article is organized as follows. Section 2 discusses existing research on multi-agent planning, and provides context for our contribution. Section 3 defines the MRPCS problem. Section 4 presents our IIDP framework as well as the two algorithms developed under this framework. Section 5 provides technical details of instantiating IIDP using the specific domain of multi-robot navigation, including modeling temporal uncertainty in navigation actions. Section 6 details the experiment setup and results, including demonstrations on a real multi-robot system. Section 7 concludes this article, and provides a few open problems for future work.

2 Related work

IIDP algorithms solve problems by iteratively using a two-stage process: in the first stage, each agent employs a single agent planner, while in the second stage the agents exchange plans and negotiate over dependencies. We begin the review of related work with single-agent planners followed by a detailed discussion of multi-agent planners.

Planning was one of the earliest research areas in artificial intelligence, and aims at helping an agent to plan a

sequence of actions to accomplish complex tasks that are impossible using individual actions. The key to formalizing a planning problem is the description of actions. Action description languages (or action languages) have been developed for describing preconditions and effects of actions, and can be used for describing planning domains. The STRIPS language (Fikes and Nilsson 1972), as part of the Shakey project (Nilsson 1984), is the earliest action language. We refer the interested reader to a journal article that reviewed early action languages (Gelfond and Lifschitz 1998), where the focus was mainly on developing languages that are more natural and concise in describing planning domains. Nowadays, PDDL Ghallab et al. (1998) is arguably the most widely used action language that is supported by a variety of planning algorithms and implementations (Helmert 2006; Hoffmann and Nebel 2001). 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 (Lee et al. 2013), and BC programs can be solved using systems such as Clingo Gebser et al. (2014). While we use BC in this work, it should be noted that IIDP, as an efficient multi-robot planning framework, is not restricted to any action languages or planning systems.

Multi-agent planning Multi-agent planning (MAP) is a variant of the automated planning domain where plans must be computed for multiple agents acting in a joint state space. A line of publications offered a formal planning language description for representing MAP, including multi-agent extension of STRIPS (Boutilier and Brafman 2001), Multi-agent Planning Language (MAPL) (Brenner 2003), Concurrent STRIPS (CSTRIPS) (Oglietti and Cesta 2004), MA-STRIPS (Brafman and Domshlak 2008), and MA-PDDL (Kovács et al. 2012). A competition of distributed and multi-agent planning took place in 2015 as part of the workshop on distributed and multi-agent planning. The problems in that competition were formulated in the MA-PDDL language. Among the solvers that participated, the ADP planner (Crosby et al. 2013) ranked first in the centralized track and the PSM planner (Komenda 2016) ranked first in the distributed track. The problem domain for that competition was limited to scenarios with multiple agents that are not self-interested and are working together to synthesize a joint plan that solves a common goal. All agents wish thereby for the goal to be reached at the end of the task execution and all share a common utility function. The work covered in this article, by contrast, considers self-interested agents, each with a specific and private goal. Moreover, the utility functions and action costs are not assumed to be common among the different agents. The agents' collaboration behaviors produced in this article are a result of the agents leveraging each other's actions and

avoiding competing for constrained resources, where neither is detrimental to the individual agents' own interests.

Multi-robot symbolic planning Action languages, including PDDL, have been used for symbolic planning for a team of robots Filippidis et al. (2012), Alur et al. (2013), Knepper et al. (2013), Buehler and Pagnucco (2014), Wong and Kress-Gazit (2015, 2016). However, they either do not model possible runtime conflicts (assuming that plans computed can be successfully executed to the end without any interruptions (Knepper et al. 2013; Buehler and Pagnucco 2014) or aim at resolving conflicts *locally* at runtime (Filippidis et al. 2012; Alur et al. 2013; Wong and Kress-Gazit 2015, 2016). 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 (Wong and Kress-Gazit 2015). Such conflict-resolving actions can be very expensive in practice, yielding locally optimal solutions. Very recently, a time-bounded multi-robot planning approach was developed for a team of logistics robots (Schäpers et al. 2018). Like this article, they considered action durations via temporal planning. Unlike this article, they did not consider the uncertainty in action durations, and their planning system models a limited time horizon (of 3 min) to reduce the search space.

While IIDP can be applied to multi-agent planning problems, it is particularly useful to multi-robot systems, where a team of physical agents frequently have to avoid conflicts (due to limited resources) and build action synergies when possible. In comparison to these existing methods, our proposed IIDP algorithms compute high-quality, joint plans, while successfully avoiding computing plans in a joint space. We also model noisy action durations, as one of the factors that affect runtime conflicts, and avoid such conflicts (in probability) at planning time.

Multi-robot task allocation (MRTA) MRTA aims at allocating a set of tasks to a set of robots. MRTA includes a family of problems that can be classified based on the number of tasks a robot can execute at a time (single-task versus multi-task), the number of robots each task requires at most (single-robot vs. multi-robot) and if planning is needed (instantaneous versus time-extended) (Korsah et al. 2013). In this article, the robot team starts with each robot having a fixed task with no opportunity to transfer tasks to other robots, assuming the availability of a single-task, single-robot, time-extended MRTA algorithm. More details on the assumptions made in this article are available in Sect. 3. In practice, it should be noted that MRTA algorithms by themselves cannot compute symbolic plans.

Multi-robot scheduling A multi-robot 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 (Brucker 2007;

Zhang and Parker 2013; Coltin and Veloso 2014). Recent work on multi-robot scheduling further considers temporal uncertainty (in a multi-robot navigation task) (Brooks et al. 2015). However, in scheduling algorithms, generally a predefined and fixed set of tasks are given and need to be scheduled whereas in planning algorithms, actions are generated to achieve a final goal. Hence scheduling algorithms cannot be used for generating action sequences in complex domains that require reasoning about actions.

Multi-robot 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 (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 multi-robot systems using a single (PO)MDP (Khandelwal et al. 2015), multiple (PO)MDPs (Zhang et al. 2013), and DEC-POMDPs (Amato et al. 2015). 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. Therefore, symbolic planning, in comparison to (PO)MDP-based, is more suitable to problems where there are many potential goals and human-interpretable plans are required.

Multi-robot motion planning Existing work has investigated the problem of multi-robot concurrent task assignment and motion (trajectory) planning (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. Although such motion planning algorithms are complimentary to our multi-robot 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). Similar drawbacks exist when considering the task of multi-robot path planning which has received much attention lately (Sharon et al. 2015, 2013; Hönig et al. 2016). Moreover, such planners usually consider the joint-agent state/action space which limits their scalability. A prominent example is presented in Alonso-Mora et al. (2018) where an LTL encoding of motion Raman et al. (2013) is used as a centralized MAP system.

Distributed constraint optimization (DCOP) The DCOP framework (Modi et al. 2003) has been extensively used for

modeling and solving distributed multi-agent coordination problems, ranging from virtual agents (Matsui et al. 2008), meeting scheduling (Hoang et al. 2016), emergency vehicles (Ferreira et al. 2009), and sensor networks (Lesser et al. 2012) to mobile robot teams (Jain et al. 2009; Yedidsion and Zivan 2016). In this body of work, the papers that applied DCOP to real robots only considered one action at each iteration; to the best of our knowledge no previous work has incorporated DCOP with planning so that agents reason about a sequence of actions at every iteration.

In this article, we develop the IIDP framework and algorithms that compute a joint plan for a team of robots, without computing plans in a joint search space, significantly improving the planning efficiency compared to central planning techniques and baseline algorithms such as independent or sequential planning.

3 Definition of MRPCS problems

In this section we detail the assumptions taken to represent the Multi-Robot planning with conflicts and synergies (MRPCS) problem and formally define it.

Distributed system: We assume the set of robots are distributed, in the sense that each has its own goal, knowledge representation, and the ability to plan a sequence of actions to achieve that goal. We assume that each robot can work on at most one task at a time and each task requires only one robot, which corresponds to “single-robot”, “single-task” multi-robot planning problems (Gerkey and Matarić 2004). The tasks are not transferable among robots, and there is no priority to one robot’s task over another’s. Agents’ tasks are loosely coupled, such that one agent’s action can affect the cost of another’s, but there is no action that strictly requires the collaboration of two or more agents.

Communication: We assume that agents are able to communicate with each other reliably, anytime, anywhere about resources required by their current plans (e.g., which corridors are needed and when), which can be realized by directly communicating about the agents’ plans.

Resource sharing and action synergies: Collaborations among robots are realized via resource sharing and constructing action synergies. We assume the agents share the same environment and constrained resources, e.g., corridors that allow one robot at a time. Agents are self-interested and are unwilling to sacrifice their utility to improve social welfare. However, they are indifferent to taking actions that help other agents without introducing extra costs in their own goal achievement, such as sharing door openings or leading a platoon.

Uncertainty: We assume that action outcomes are deterministic, but action durations and costs are uncertain. For instance, two robots running into each other in a narrow corridor (that probabilistically allows only one robot at a time) can still achieve their original navigation goals, but resolving this conflict causes both robots significant extra costs (in the form of extra time to complete navigation actions). This setting motivates the robots to plan to avoid such conflicts.

Homogeneity: We assume that the robots are homogeneous, i.e., all robots share the same set of actions, and action outcomes (deterministically or probabilistically) are the same while being executed on different robots.

Costs: The agents estimate expected costs of their own plans according to the current plans of other agents. Each agent can model its own action costs and interaction costs (i.e. changes in action costs caused by conflicts and synergies). For example, the time for a robot to execute a navigation action at normal speed is an individual action cost, and the time it is blocked by another robot is an interaction cost. Interaction costs are asymmetric, i.e., a difference in cost for one agent in an interaction is not necessarily the same difference in cost for its counterpart. We model collaboration failures of constrained resources as finite costs (i.e., soft constraints).

Following the above assumptions, an MRPCS problem can be specified in the form of $\langle \mathcal{N}, \mathcal{D}, \mathcal{A}, \mathcal{S}, \mathcal{G}, \mathcal{F} \rangle$:

- \mathcal{N} is a set of robots. $|\mathcal{N}| = N$
- \mathcal{D} is a description of the environment, including objects, their properties, and their relations.
- \mathcal{A} is a set of robot actions, each $a \in \mathcal{A}$ being described by its preconditions, effects, and cost $C_A(a)$.
- \mathcal{S} is a set of states in which each is the current state for a robot: $s_i \in \mathcal{S}$ is the state of the i th robot and $|\mathcal{S}| = N$.
- \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{T} is a set of interactions in which each is a set of actions $\{a_0, a_1, \dots\}$ that lead to conflicts or synergies when executed at the same time. The cost of an interaction $t \in \mathcal{T}$ is $C_T(t)$.

The 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 current 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 Sect. 5.1).

The overall goal of solving an MRPCS problem is to maximize overall system utilities under the assumption that agents are self-interested, i.e., to find the user equilibrium with the best social welfare. The means of efficiently solving an MRPCS problem is by providing the agents with an algorithm that will determine when and what they communicate with each other, and when and how they plan and re-plan. The solution is a set of plans $P = \{p_0, p_1, \dots, p_{N-1}\}$, one for each robot, where the objective is to minimize $C(P)$, the total costs of actions and the expected costs of interactions:

$$C(P) = \sum_{i=0}^{N-1} \sum_{a \in p_i} C_A(a) + \sum_{t \in \mathcal{T} \cap P} C_T(t) \cdot Pr(t, P) \quad (1)$$

where $\mathcal{T} \cap P$ is the set of interactions that can be enabled by actions in P , and $Pr(t, P)$ is the probability that t occurs if P is executed (with temporal uncertainties).

4 The IIDP framework and two IIDP algorithms

Optimally solving multi-robot planning problems requires modeling spaces of both joint states and joint actions. The exponentially increasing number of joint actions and possible inter-dependencies of concurrent actions make optimal multi-robot planning challenging. The complexity of multi-robot planning is analyzed in Wagner and Choset (2015). When we further consider noisy action costs, such as the temporal uncertainty in action durations, the MRPCS problem 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 Multi-Robot Planning with Conflicts and Synergies (MRPCS) problems that cannot be solved using existing methods.

It is observed that single-robot symbolic task planning methods typically abstract away lower-level planning such as continuous motion planning, and focuses on finding a sequence of higher-level actions required to achieve a goal task. Similar to that abstraction process, our Iterative Inter-Dependent Planning (IIDP) framework for MRPCS problems abstracts away the process of single-robot planning and focuses on creating a mechanism that enables the agents to efficiently reason about conditional plans. We further define *the inter-dependent cost function* $\mathcal{C}(p, P^M)$ to be a function that estimates the sum of individual action costs of a plan p , and the interaction costs with a set of other robots M and their current plans P^M . The implementation of $\mathcal{C}(p, P^M)$ requires the modeling of action preconditions, effects, and

costs, which is highly domain dependent and hence its development is independent of IIDP.

4.1 The general structure of an IIDP algorithm

Algorithm 1 IIDP: The general framework for MRPCS problems

Input: A state s_i , and goal g_i for robot r_i .

Output: Plan p_i

```

1: Generate initial plan  $p_i$ , where  $s_i \xrightarrow{p_i} g_i$ 
2: while stop_condition()==false do
3:   Send  $p_i$  to other robots in the system
4:    $M = \mathcal{M}(r_i)$ , the set of robots to consider
5:   Receive  $P^M$ , plans from robots in  $M$ 
6:    $p_i = \text{compute\_plan}(P^M)$ 
7: end while
8: return  $p_i$ 

```

The general structure of the IIDP algorithms is depicted in Algorithm 1,

It defines an iteration-based negotiation, where in each iteration, each agent computes a plan, sends it to the other agents and receives theirs, then decides whether to switch to an alternative plan. The input of IIDP includes robot r_i 's initial and goal states (s_i and g_i). The $s \xrightarrow{p} g$ symbol represents that plan p leads to state transitions from initial state s to goal g . In the end of the program, plan p_i is returned. The framework assumes that all the robots concurrently run the same algorithm. $\mathcal{M}(r_i)$ is a function that returns a subset of robots that r_i should consider. The implementation of \mathcal{M} is orthogonal to the IIDP framework and is purposely left flexible to allow domain-dependent optimization of planning time. For example, to apply the IIDP framework to a building-wide team of robots, it is reasonable to assume interactions can only occur among spatially close robots, and make \mathcal{M} return the robots that are currently on the same floor as r_i . The notation P^M represents the set of plans of robots in the subset M .

4.2 The BEST ALTERNATIVE algorithm

Algorithm 2 shows the BEST ALTERNATIVE algorithm.

In this algorithm, the agents reason about which one has the best alternative plan and the one with the best alternative switches to its alternative plan². The difference between the inter-dependent cost of the current plan and the inter-dependent cost of the best alternative plan for robot r_i is: $\delta_i = \mathcal{C}(p_i, P^M) - \mathcal{C}(\hat{p}_i, P^M)$.

The stopping condition for this algorithm is realized when no agent has a better alternative plan. Since reaching this con-

² Ties can be broken in favor of the agent with the longest action sequence or by any other metric.

Algorithm 2 The BEST ALTERNATIVE algorithm

Input: A state s_i , and goal g_i for robot r_i , and max number of iterations Θ

Input: Max number of iterations Θ

Output: Plan p_i

```

1: Generate initial plan  $p_i = \arg \min_p (\mathcal{C}(p, \{\}))$ , where  $s_i \xrightarrow{p} g_i$ 
2: for each  $k \in \{0, 1, \dots, \Theta\}$  do
3:   Send  $p_i$  to other robots in the system
4:   Get the set of robots to consider  $M = \mathcal{M}(r_i)$ , and receive their plans  $P^M$ 
5:   Generate best alternative plan  $\hat{p}_i = \arg \min_p (\mathcal{C}(p, P^M))$ , where  $s_i \xrightarrow{p} g_i$ 
6:   Compute the cost difference  $\delta_i = \mathcal{C}(p_i, P^M) - \mathcal{C}(\hat{p}_i, P^M)$  between the current plan and the best alternative plan
7:   Send  $\delta_i$  to other robots and receive updated plans  $P^M$ 
8:   if  $\delta_i > \max\{\delta_{PM}\}$  then
9:     Change current plan to  $\hat{p}_i$ 
10:  else
11:    if  $\delta_i == \max\{\delta_{PM}\}$  &  $r_i$  wins tie breaking rule then
12:      Change current plan to  $\hat{p}_i$ 
13:    end if
14:  end if
15:  if  $\forall j \in \mathcal{N} \setminus r_i, \delta_j \leq 0$  then
16:    break
17:  end if
18: end for
19: return  $p_i$ 

```

dition cannot be guaranteed, we added a limit on the number of iterations to ensure termination.

It should be noted that in Line 5, the operation of arg min requires a symbolic task planner for computing a sequence of actions while minimizing the overall plan cost. The notion of distributed optimization through using individual best alternative was appropriated from the maximum gain message (MGM) algorithm (Maheswaran et al. 2004) and its adaptation to mobile agent teams (MGM_MST) with dynamically changing constraints (Zivan et al. 2015).

4.3 The INCREASING DEPENDENCY algorithm

Algorithm 3 shows the INCREASING DEPENDENCY algorithm.

Informally, it iteratively computes and saves the conditional “optimal” plan for each robot given other robots’ current plans. In each iteration, the cost of conflict penalty increases (from zero in the first iteration to its full cost in the last iteration) and the utility of enabling a synergy increases (from 0 to full cost reduction).

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.

Algorithm 3 The INCREASING DEPENDENCY algorithm**Input:** A state s_i , and goal g_i for robot r_i .**Input:** Θ : number of rounds of “negotiations”, $\Theta \geq 0$ **Input:** $\mathcal{N}^<_j$ A predefined (arbitrary) order over \mathcal{N} **Output:** Plan p_i

- 1: Generate initial plan $p_i = \arg \min_p (\mathcal{C}(p))$, where $s_i \xrightarrow{p} g_i$
- 2: Send p_i to other robots
- 3: Get the set of robots to consider, $M = \mathcal{M}(r_i)$
- 4: Receive the updated plans of $M^<_i$, which is the subset of agents in M who precede i according to $\mathcal{N}^<_j$
- 5: $\alpha = 0$
- 6: $p_i = \arg \min_p (\mathcal{C}(p, P^{M^<_i}, \alpha))$, where $s_i \xrightarrow{p} g_i$
- 7: **for each** $k \in \{1, \dots, \Theta\}$ **do**
- 8: Send p_i to other robots
- 9: Get the set of robots to consider $M = \mathcal{M}(r_i)$, and receive their plans P^M
- 10: $\alpha = k/\Theta$
- 11: $p_i = \arg \min_p (\mathcal{C}(p, P^M, \alpha))$, where $s_i \xrightarrow{p} g_i$
- 12: **end for**
- 13: **return** p_i

The value of Θ influences the performance of INCREASING DEPENDENCY in the following way. In lines 7–12, we enter a for-loop that has $\Theta + 1$ iterations, where α is a *negotiation depth* parameter that incrementally grows by $1/\Theta$ in each iteration (Line 10). 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 high costs); when $\alpha = 1$, it considers its teammates as important as itself (costs are not discounted). Inter-dependent planning is conducted in Line 11, where we compute the optimal conditional plan while minimizing the conditional plan cost given the current plans of its $M^<_i \subseteq M$ dependent teammates that precede it in the planning order $\mathcal{N}^<_j$, which is a predefined (arbitrary) order over \mathcal{N} .

The difference between the INCREASING DEPENDENCY algorithm and the BEST ALTERNATIVE algorithm is that in the BEST ALTERNATIVE algorithm the agents calculate alternative plans according to the full costs of plan dependencies. On the other hand, in the INCREASING DEPENDENCY algorithm, agents explore the range of solutions between no dependency and full dependency to detect better solutions that lie in between these two extremes.

4.4 Illustrative examples

We make the following assumptions in the following examples:

1. Robots conflict at a node if they arrive at the same time step (even when they are moving in the same direction).
2. Costs of conflicts are cumulative, i.e. the cost of two conflicts is higher than the cost of one conflict. This allows

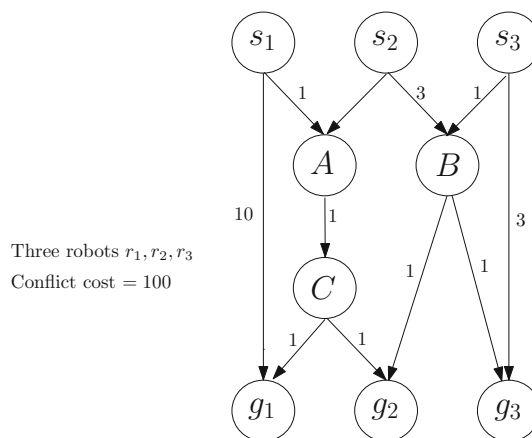


Fig. 4 Example for the different behavior of the two algorithms. In this example INCREASING DEPENDENCY outperforms BEST ALTERNATIVE. By using the BEST ALTERNATIVE algorithm, agents r_1 and r_2 conflict and the total cost is 105, while using the INCREASING DEPENDENCY algorithm, results in no conflict and a solution cost of 10

the algorithm to minimize the probability of conflicts, if not possible to avoid them completely.

3. Robot i starts at s_i and plans to go to g_i .

An illustrative example is shown in Fig. 4 to reflect the difference between the algorithms’ performance. In this example INCREASING DEPENDENCY outperforms BEST ALTERNATIVE. Consider three robots (r_1 , r_2 , and r_3) with their respective start and goal states ($s_i \rightarrow g_i$). Nodes A , B and C represent constrained states which cause conflicts if two agents reach them at the same time step. The edges represent actions that transition from one state to another while the numbers next to each edge represent the cost of taking that action. Conflict costs are modeled by a very large number $\gg 100$. In the BEST ALTERNATIVE algorithm, the agents would start by selecting their best independent plans:

$$\begin{aligned} r_1 : s_1 &\rightarrow A \rightarrow C \rightarrow g_1 \\ r_2 : s_2 &\rightarrow A \rightarrow C \rightarrow g_2 \\ r_3 : s_3 &\rightarrow B \rightarrow g_3 \end{aligned}$$

This set of plans creates two conflicts for robots r_1 and r_2 in nodes A and C . Robot r_1 ’s best alternative is to take an expensive action (100) directly to its goal g_1 . Robot r_2 ’s best alternative is to go through node B and still have one conflict with robot r_3 . In this case only robot r_1 will switch and the resulting solution cost is 105.

In the INCREASING DEPENDENCY algorithm, the initial plans would be the same but as conflict costs increase to 2, agent r_2 will switch to the plan that goes through node B , followed by agent r_3 switching to a direct action to g_3 resulting in no conflict and a solution cost of 10, which is also the optimal solution in this case.

Figure 5 provides an example where BEST ALTERNATIVE outperforms INCREASING DEPENDENCY. Consider two robots

Two robots r_1, r_2
 Conflict at node $A = 100$ each,
 Order for ID: $r_1 \rightarrow r_2$
 $\Theta = 20$
 Unless specified otherwise edge costs are 0

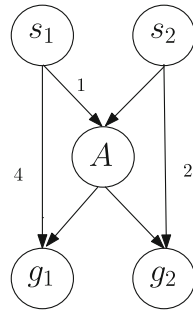


Fig. 5 Example for the different behavior of the two algorithms. In this example BEST ALTERNATIVE outperforms INCREASING DEPENDENCY

(r_1, r_2) with their respective start and end states ($s_i \rightarrow g_i$). Node A represents a constrained state which causes a conflict if two agents reach it at the same time step. The edges represent actions that transition from one state to another while the numbers next to each edge represent the cost of taking that action. Conflict costs are modeled by a cost of 100 for each robot. In the BEST ALTERNATIVE algorithm, the agents would start by selecting their best independent plans:

$$\begin{aligned} r_1 &: s_1 \rightarrow A \rightarrow g_1 \\ r_2 &: s_2 \rightarrow A \rightarrow g_2 \end{aligned}$$

This set of plans creates a conflict for robots r_1 and r_2 in node A . Robot r_1 's best alternative is to take $s_1 \rightarrow g_1$ directly to its goal with a cost of 4 resulting in $\delta_1 = 101 - 4 = 97$. Robot r_2 's best alternative is to go directly to its goal with a cost of 2 resulting in $\delta_2 = 100 - 2 = 98$. In this case only robot r_2 will switch and the resulting solution cost is $1+2=3$, which is the optimal solution.

In the INCREASING DEPENDENCY algorithm, the initial plans would be the same but in the first round of negotiation, as the conflict costs increase to 5, agent r_1 will switch to the plan that goes directly to g_1 incurring a cost of 4. Agent r_2 which is the second in the planning order would not switch its plan since now it has no conflict, resulting in a solution cost of 4, which is not the optimal solution in this case.

4.5 Baseline algorithms

A very crude baseline to compare inter-dependent planning algorithms against is simply having the agents greedily compute individual plans without considering those of other agents (which is the initial stage of the IIDP algorithm's planning process).

A slightly more advanced baseline would be a single sequence of inter-dependent planning where each agent plans according to the plans of the agents in front of it in a set order. We call this algorithm SINGLE ORDER.

In fact, we can see both of these baseline algorithms as variations of the INCREASING DEPENDENCY algorithm. When $\Theta = 0$, the robots compute plans independently, because we define $\alpha = 0$ which results in the INDEPENDENT PLANNING algorithm. When $\Theta = 1$, there is only one round of negotiation which results in the SINGLE ORDER configuration. The performance of SINGLE ORDER 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 get the intended INCREASING DEPENDENCY algorithm.

Finding the best order requires going over all possible $N!$ orderings. As another baseline, we propose the BEST ORDER algorithm which provides the solution achieved by the best possible sequential planning order.

Algorithm 4 The BEST ORDER algorithm

Input: A state s_i , and goal g_i for robot r_i .

Input: $\mathcal{N}^<$, A set of all possible $N!$ orderings of the elements in \mathcal{N}

Output: Plan p_i

```

1: for each  $j \in \{0, 1, \dots, N!\}$  do
2:    $\mathcal{N}^<_j$  = the  $j$ -th ordering in  $\mathcal{N}^<$ 
3:   if  $i == 0$  then
4:     Send  $p_i$  to agent  $i + 1$ , According to  $\mathcal{N}^<_j$ 
5:   else if  $i \neq N$  then
6:     Receive  $p_i$  from agent  $i - 1$ 
7:      $p_i = \arg \min_p (\mathcal{C}(p, \mathcal{D}, p_{i-1}))$ , where  $s_i \xrightarrow{p} g_i$ 
8:     Send  $p_i$  to agent  $i + 1$ 
9:   else
10:    Receive  $p_i$  from agent  $i - 1$ 
11:     $p_i = \arg \min_p (\mathcal{C}(p, \mathcal{D}, p_{i-1}))$ , where  $s_i \xrightarrow{p} g_i$ 
12:  end if
13: end for
14: return  $p_i$ 

```

We present examples as evidence that neither the BEST ORDER nor INCREASING DEPENDENCY algorithms always find the optimal plan.

Figure 6 shows an example where INCREASING DEPENDENCY outperforms BEST ORDER (Left) and an example of BEST ORDER outperforms INCREASING DEPENDENCY (Right).³

In the left diagram of Fig. 6, the individual plans can possibly conflict at nodes A and D . INCREASING DEPENDENCY successfully avoids both conflicts by suggesting plan ($s_1 \rightarrow A \rightarrow C \rightarrow g_1, s_2 \rightarrow B \rightarrow D \rightarrow g_2$), producing the optimal solution. BEST ORDER will try two orderings: $1 \rightarrow 2$ and $2 \rightarrow 1$. SINGLE ORDER with planning order

³ Note that in this example BEST ALTERNATIVE performs the same as INCREASING DEPENDENCY and so we only refer to INCREASING DEPENDENCY.

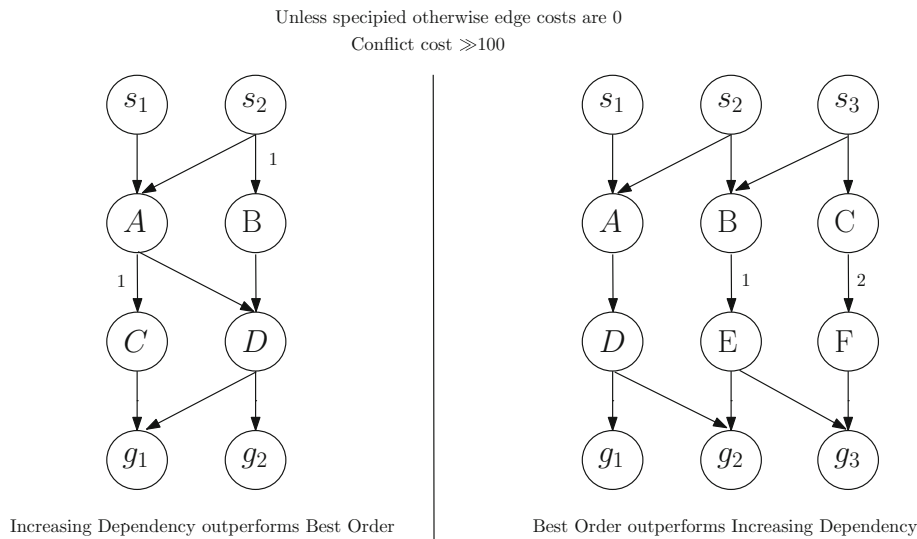


Fig. 6 Two examples with avoiding conflicts. The left diagram shows a case where INCREASING DEPENDENCY outperforms BEST ORDER. The right diagram shows a three-robot example where BEST ORDER finds the optimal plan and INCREASING DEPENDENCY fails. Each node in the

$1 \rightarrow 2$ generates the plan $(s_1 \rightarrow A \rightarrow D \rightarrow g_1, s_2 \rightarrow B \rightarrow D \rightarrow g_2)$, which results in a conflict at D . Similarly, SINGLE ORDER with the planning order $2 \rightarrow 1$ generates a plan that causes a conflict at A . In this example, INCREASING DEPENDENCY outperforms BEST ORDER.

The right diagram of Fig. 6 shows a three-robot example where BEST ORDER outperforms INCREASING DEPENDENCY. BEST ORDER is able to find the optimal plan (with the ordering $1 \rightarrow 2 \rightarrow 3$): $(s_1 \rightarrow A \rightarrow D \rightarrow g_1, s_2 \rightarrow B \rightarrow E \rightarrow g_2, s_3 \rightarrow C \rightarrow F \rightarrow g_3)$. INCREASING DEPENDENCY produces a locally optimal solution $(s_1 \rightarrow A \rightarrow D \rightarrow g_1, s_2 \rightarrow A \rightarrow D \rightarrow g_2, s_3 \rightarrow B \rightarrow E \rightarrow g_3)$. r_1 and r_2 cannot switch to conflict-free plans due to the plan of r_3 , while r_3 has no incentive to change its plan.

4.6 Complexity analysis

The complexity of the different algorithms can be analyzed in two aspects, computation complexity and communication complexity. The presented analysis is a function of four variables: N which is the number of robots, C which denotes the complexity of a single inter-dependent planning operation (C in Line 11), SR which is the complexity of a send/receive operation in which a single robot sends out its plan to the rest of the robots and receives their plans, and Θ which is the negotiation depth.

1. The BEST ALTERNATIVE algorithm has $O(\Theta \cdot (C + N \cdot SR))$ complexity in the worst case since the algorithm runs for Θ times at most, and at each iteration all the robots compute their plans simultaneously (C). The

graph represents a state of a single agent. Each edge is an action that can be taken from the connected node. There are no self edges (the agent can never stay in its current state). No more than one agent can be in the same state

send/receive operation is multiplied by N to account for systems where the communication is sequential.

2. The INCREASING DEPENDENCY algorithm requires that each robot performs one inter-dependent planning operation, and one send/receive operation at each negotiation depth. The inter inter-dependent planning for each robot is done in a sequential manner (one robot at a time) summing up to a total complexity of $O(\Theta \cdot N \cdot (C + SR))$.
3. The BEST ORDER algorithm has $O(N! \cdot N \cdot (C + SR))$ complexity because for every possible order ($N!$), every robot, by order (N), performs a single inter-dependent planning operation (C), and a send/receive operation (SR). The BEST ORDER algorithm is therefore not applicable to large instances and will only be discussed in relation to small scale examples.

Note that none of the above variants of the IIDP framework produce a contingency plan. Uncertainty in action duration might lead to a plan's failure during runtime. In such cases, the IIDP framework is reactivated in order to compute a valid plan. In theory, a plan might fail at every time step resulting in a new call to the IIDP planner. In such cases, the complexity analysis provided above should be multiplied by a factor of T , where T is the planning horizon.

5 An instantiation of IIDP

In this section, we instantiate IIDP using a multi-robot navigation task. Symbolic task planning techniques are needed because robots need to reason about between-room accessi-

bilities 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 Khandelwal et al. (2014), Zhang et al. (2015). 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 Lee et al. (2013) 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 article are not restricted to specific action languages or symbolic planners. We adapt existing formulations of BC-based, single-robot navigation tasks Khandelwal et al. (2014), Zhang et al. (2015) for multi-robot settings. For instance, we use the following rules to define the ownership between rooms and doors:

hasdoor($l1, d1$). *hasdoor*($l1, d2$). *hasdoor*($l2, d2$). ...
default \neg *hasdoor*(L, 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*(L) to represent door D is open, the robot is facing door D , the robot is beside door D , and the robot is in room L . Robot identities are not included in the representation of a robot's location, *loc*(L), 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 multi-robot systems. As an example of how actions are modeled, we arbitrarily select action *cross*(D) and present its definition as below. Crossing door D changes the robot's location from L_1 to L_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*(L_2) **if** *loc*(L_1), *acc*(L_1, D, L_2).
nonexe *cross*(D) **if** *loc*(L), \neg *hasdoor*(L, 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 (Gebser et al. 2014).

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 article, 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 article.
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 .⁴
4. Non-navigation actions do not introduce extra uncertainty at run time, and navigation actions cannot be delayed more than once.

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:

$$\hat{f}(k, \lambda) = \lambda^k e^{-\lambda} / k! \quad (2)$$

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

⁴ For example, such delays can be caused by forcing the robot to stop and say "excuse me" as is done by CoBots Veloso et al. (2015).

Proposition 1 If X and Y are two independent discrete random variables with a Poisson distribution: $X \sim \text{Poisson}(\lambda_1)$ and $Y \sim \text{Poisson}(\lambda_2)$, then their sum $Z = X + Y$ follows another Poisson: $Z \sim \text{Poisson}(\lambda_1 + \lambda_2)$ (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 $\hat{f}(k', \lambda')$. Following Assumption 1, parameter λ' is a function of traveled distance d :

$$\lambda'(d) = \lambda \cdot t(d) = \lambda \cdot d/v \quad (3)$$

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

$$t = t^{act} + t^{del} = d/v + k' \cdot \delta \quad (4)$$

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 Eqs. 3 and 4, we can see the overall navigation time t follows a *shifted* Poisson distribution with PDF:

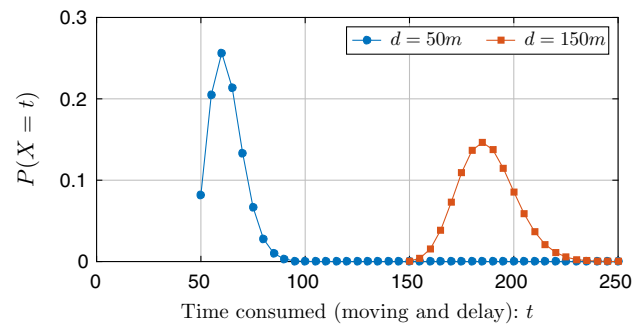
$$f(t, \lambda'(d)) = (\lambda'(d))^{\frac{t-d/v}{\delta}} \cdot e^{-\lambda'(d)} / \left(\frac{t-d/v}{\delta}\right)! \quad (5)$$

Figure 7a 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 Fig. 7b suggest that a shifted Poisson distribution can well represent noisy durations of navigation actions (with parameters properly set).

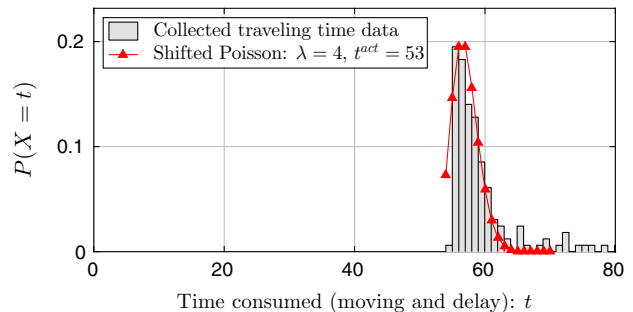
We further remove the parameter of λ' and substitute d/v with t^{act} , and use $\text{Dist}(t^{act}, \lambda')$ to represent a distribution over possible lengths of completion time. Modeling noisy action durations in this way paves the way to further investigating how uncertainty is accumulated over plans that include a sequence of actions. For instance, $t^{act} = 50$ and $\lambda' = 2.5$ correspond to the (blue) circle-mark curve in Fig. 7a. Since (we assume) non-navigation actions do not introduce extra uncertainty at running time, Equation 4 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 $\langle a_0, a_1, \dots \rangle$ 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).

$$p : \langle (a_0, t_0^{act}, \lambda'_0), (a_1, t_1^{act}, \lambda'_1), \dots \rangle$$

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



(a) Noisy durations of navigation actions in simulation.



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

Fig. 7 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

$$\text{Dist} \left(\sum_{k=0}^{K-1} t_k^{act}, \sum_{k=0}^{K-1} \lambda'_k \right)$$

Therefore, $\text{Dist}(t^{act}, \lambda')$ 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.

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

In a two-robot system that includes robots r and r' , p and p' are the 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 conflicts 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 navigation action a over time (parameter λ omitted from PDFs), while

the overlapping in space is handled by the symbolic task planner:

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

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 .⁵

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).

$$t^{wait}(a, a') = \int_0^{\infty} \int_{t_2}^{\infty} (t_1 - t_2) f_1^c(t_1) f_2^s(t_2) dt_1 dt_2 \quad (7)$$

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:

$$Pr^{fail}(a, a') = \int_0^{\infty} \int_{t_1}^{\infty} f_1^c(t_1) f_2^s(t_2) dt_2 dt_1 \quad (8)$$

where f_1^c is the time of robot n 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 5 presents our algorithm for computing interdependent 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.

⁵ In implementation, the integrals are replaced by summation operations, because action completions only happen at specific time instances (e.g., Fig. 7).

Algorithm 5 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 : conflict 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:        $C' \leftarrow C' + \rho \cdot t^{wait}(a, a') \cdot (1 - Pr^{fail}(a, a'))$ 
7:        $C' \leftarrow C' + \omega \cdot (1 - \alpha \cdot (1 - Pr^{fail}(a, a')))$ 
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  $C'$ 

```

6 Experiments

We have implemented IIDP in two domains:

1. An abstract domain that enables testing the algorithms’ performance on large scale instances of agents, state, action, and interaction spaces.
2. An instantiated multi-robot navigation domain where we test both in simulation and on real robots. In this domain we integrate temporal uncertainty into the model. Simulation experiments in this domain were conducted using a realistic multi-robot simulation environment (GAZEBO Koenig and Howard 2004) and a much faster simulator that does not have an interface for visualization or a physics engine for simulating conflict consequences. Noisy action durations in the abstract simulator are sampled from predefined distributions.

Experiments were conducted to investigate how different values of Θ (SINGLE ORDER vs. INCREASING DEPENDENCY) 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 IIDP in systems that include varying numbers of robots.

6.1 Abstract domain simulations

In this section we test the IIDP framework and algorithms on an abstract domain. The software simulator we developed to represent MRPCS problem was designed to enable running multiple experiments with varying numbers of agents, and different sizes of state, action, and interaction spaces. The simulator represents the planning domain of each agent in a graph with states as nodes and actions as edges. An MRPCS problem can be generated by randomly selecting state

Fig. 8 Total cost for different numbers of agents. INCREASING DEPENDENCY and BEST ALTERNATIVE overlap

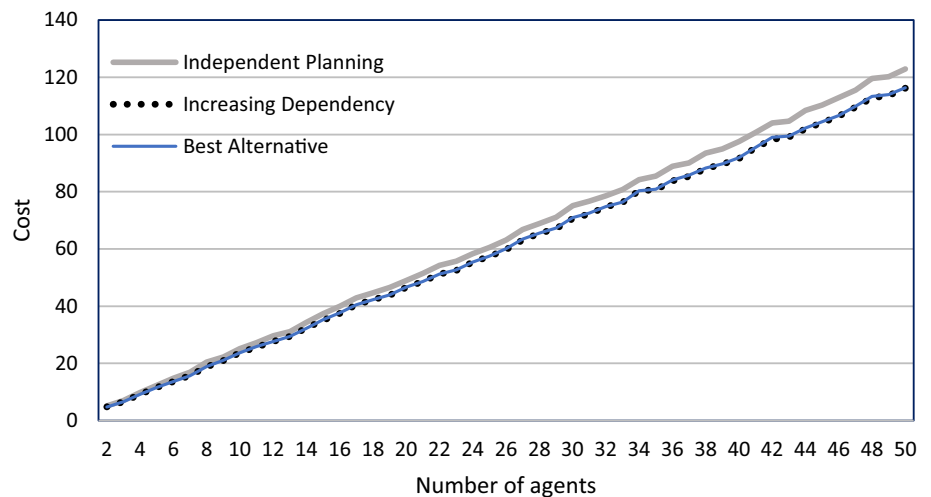
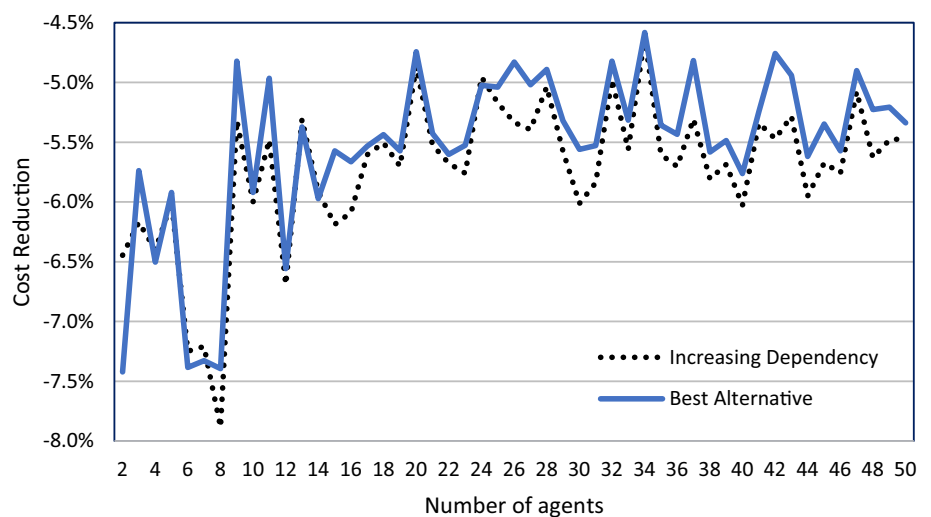


Fig. 9 Percentage of cost reduction from the INDEPENDENT PLANNING algorithm cost



transitions and agent interactions.⁶ We use the problem generator to compare the performances of the IIDP algorithms with respect to the number of agents.

The parameter setting for our experiments was chosen as follows: the number of agents $N = 2 \dots 50$, the number of states in each agent's planning domain $S = 10$, the number of actions $A = 4 \cdot S$, the total number of possible interactions $|\mathcal{T}| = N \cdot 100$. The average sequence of actions per agent from start to end goal is 5. The INCREASING DEPENDENCY parameter $\Theta = 2 \cdot A$ as well as the BEST ALTERNATIVE parameter $\Theta = 2 \cdot A$. The cost of every action is 1, the cost of every conflict is an additional 1, and the benefit of a synergy is a reduction of 1 in the cost of the action. All costs are per action per agent.

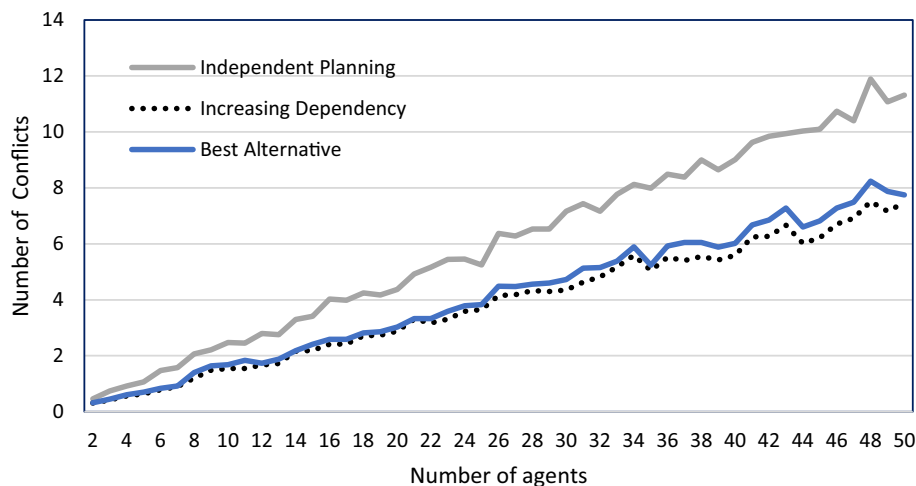
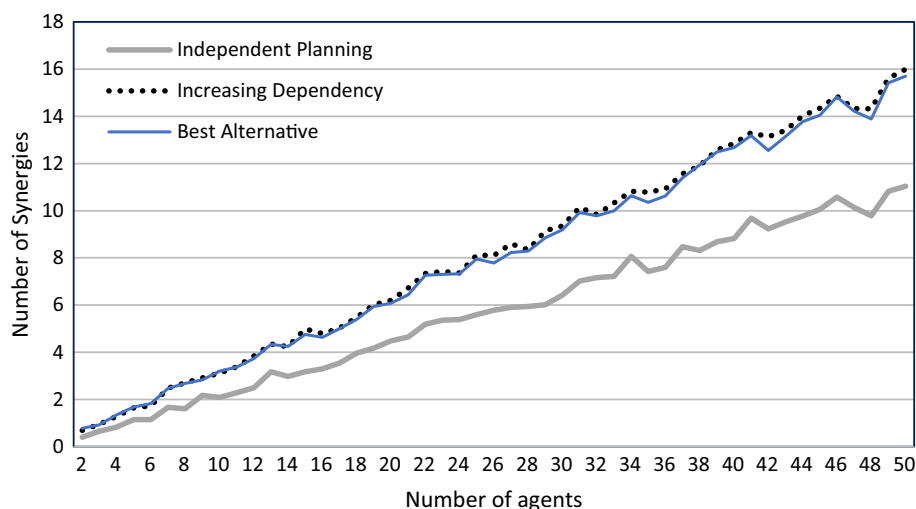
The algorithms that were tested are: INCREASING DEPENDENCY (ID), BEST ALTERNATIVE (BA), and INDEPENDENT PLANNING (IP). For every number of agents we generated

100 random problems where random elements were the start states and end goals of the agents, the mapping of actions to states, and the pairs of actions that had conflicts or synergies. In total there were 4900 different abstract domains generated and all three algorithms ran on all domains. The results of every run were recorded and averaged to compare the algorithms' performance as displayed in Figs. 8–11.

Figure 8 displays the total cost incurred by all the agents as they perform their plans. Each point in the graph is an average over 100 experiments. Both ID and BA outperform IP, with 5.7% and 5.5% average reduction respectively. These differences are statistically significant with $p\text{-value} < 0.05$, using a paired T-Test. The difference between ID and BA however are not statistically significant but we do observe that ID outperforms BA in 42 out of 49 scenarios with regards to average cost. To demonstrate that, we present Fig. 9 that offers a closer look at the difference in cost reduction each algorithm has in comparison with the IP baseline.

Figure 10 shows the average number of conflicts that the agents face. Here, ID and BA also outperform IP significantly

⁶ The code for our simulated environment is publicly available at https://github.com/YuqianJiang/multi_agent_planning.

Fig. 10 Number of conflicts for different numbers of agents**Fig. 11** Number of synergies for different numbers of agents

with 3.8 and 4.1 conflicts on average respectively compared to 6 conflicts for IP. Despite the fact that the difference in the average number of conflicts between ID and BA is not statistically significant, we do observe that ID outperforms BA in all 49 scenarios with regards to average number of conflicts.

Figure 11 shows the average number of synergies that the agents face. Again, ID and BA outperform IP significantly with 8.2 and 8 synergies on average respectively compared to 5.7 synergies for IP. ID outperforms BA in 42 out of 49 scenarios with regards to average number of synergies.

Our conclusion from this empirical evaluation is that the IIDP algorithms are scalable and invariant to the number of agents or the size of the domain. Both proposed algorithms outperform a independent planning significantly, and IP has a small advantage over BA. To test robustness of these results, we experimented with 3 times the number of interactions and found that the results were qualitatively similar. We therefore continue our analysis of a practical domain with the ID algorithm.

6.2 Multi-robot navigation domain

In this section, we evaluate the IIDP algorithms in the multi-robot navigation domain.

Gazebo simulation (Independent Planning versus Increasing Dependency) Fig. 12 (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.

Experiments in GAZEBO were conducted to visualize and validate the whole process of multi-robot plan generation and execution, and to compare INCREASING DEPENDENCY to

Fig. 12 GAZEBO simulation environment (and a picture of a human walker blocking a robot)

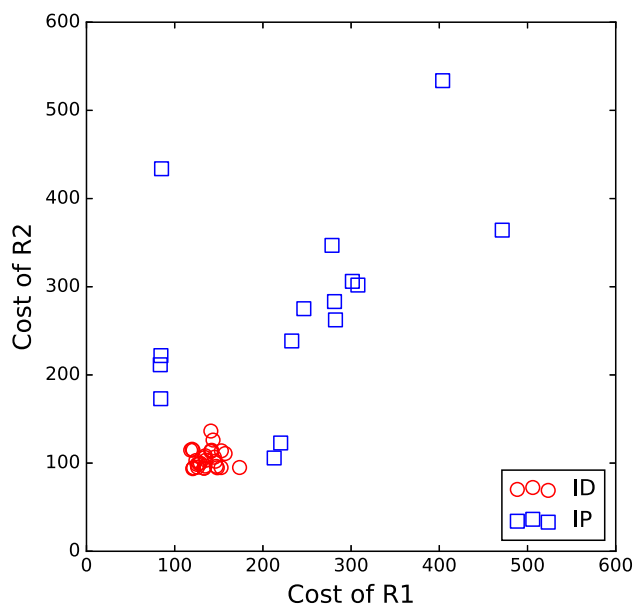
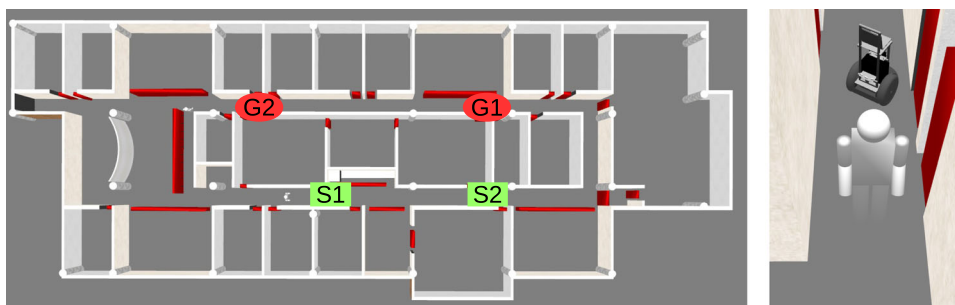


Fig. 13 Costs of robots $R1$ and $R2$ in 45 trials collected using GAZEBO simulation environment

Table 1 Average cost consumed (time) and standard deviation from GAZEBO simulation experiments (reported in Fig. 13)

	Increasing dependency	Independent planning
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

a baseline that computes plans for the robots independently INDEPENDENT PLANNING. The results in the form of execution costs of two robots are shown in Fig. 13. We can see a cluster of red circles in the bottom-left, which indicates the INCREASING DEPENDENCY 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, INCREASING DEPENDENCY ($M = 1$, $\Theta = 2$) significantly reduces the average completion time from more than 250 s to about 120 s.

Abstract simulation (Single Order vs. Increasing Dependency) In order to run a lot more trials, we use an abstract simulator: navigation actions' noisy durations are sampled from a shifted Poisson distribution (Eq. 5); conflict cost is 40; $waitforopen(D)$ failure cost is 12; and conflicts are possible only if both robots are taking navigation actions.⁷ Figure 14 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 3) affects the overall cost. Figure 15 reports the results: $\Theta = 0$ means no collaborations between robots (baseline); $\Theta = 1$ and $\Theta = 2$ correspond to SINGLE ORDER and INCREASING DEPENDENCY 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 INCREASING DEPENDENCY 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 conflicts or share doors. Comparing the triangle and square curves in both subfigures, we find that INCREASING DEPENDENCY enables more action synergies than SINGLE ORDER (via sharing door-opening actions), even when the overall cost reduction introduced by such synergies is small.

Analysis of temporal 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 16 reports the results of these experiments. When $R1$ has a head start of -1 , the overall cost of our INCREASING

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.

Fig. 14 Abstract simulation environment, where action durations are generated by sampling from pre-specified distributions. Doors are marked by the letter *d* and corridors by *cor*

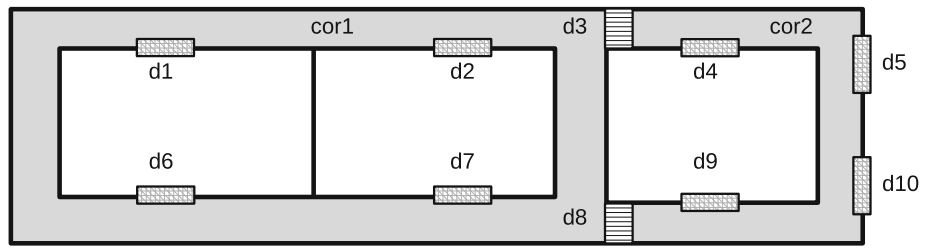


Fig. 15 Planning for a two-robot system (evaluating Θ): *R1* and *R2* need to move from *d2* to *d9* and from *d7* to *d4* respectively

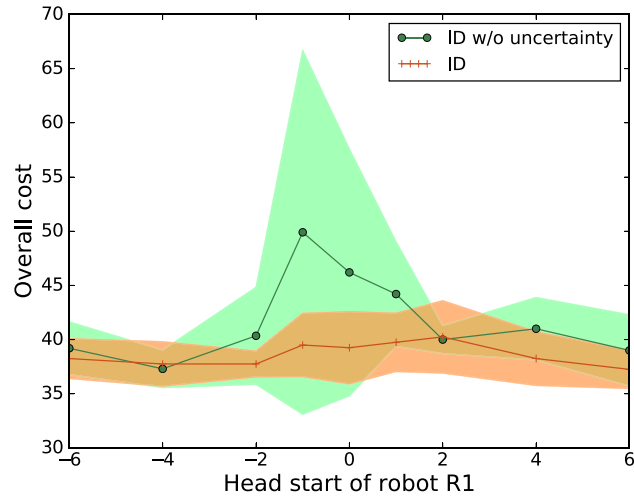
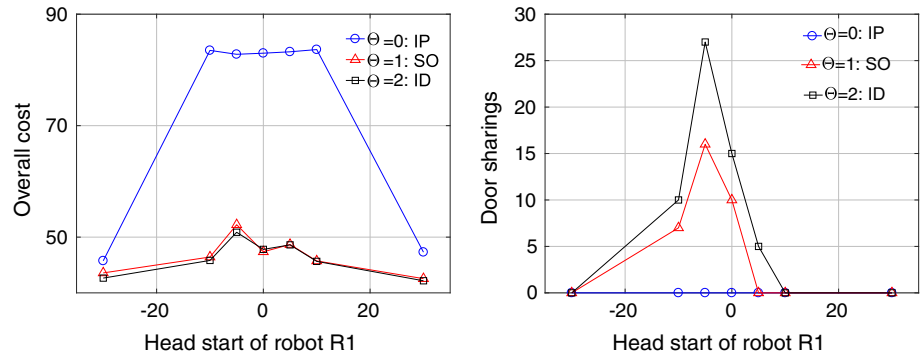


Fig. 16 Planning for a two-robot system (evaluating the need of modeling temporal uncertainty): *R1* and *R2* need to move from *d3* (*cor1* side) to *d4* and from *d7* to *d5* respectively

DEPENDENCY 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 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. INCREASING DEPENDENCY 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.

Evaluation of the effect of Θ and M In a team that includes more than two robots, IIDP algorithms have the option to consider only a *subset* of its teammates in conditional planning (specified by M in Algorithm 3). This set of experiments was conducted in a team of three robots to evaluate how the size of M affects the performance of IIDP. Accordingly, we adjust the value of Θ (SINGLE ORDER vs. INCREASING DEPENDENCY) and the value of $|M|$. In this experiment, $|M| = 1$ means M contains only the teammate right before r_i in the arbitrary order $\mathcal{N}^{< j}$. Robots *R2* and *R3* have different head starts before *R1*'s plan execution. The subfigures of Fig. 17 report the results of nine different head start combinations. In each subfigure, the x-axis corresponds to one of the four INCREASING DEPENDENCY configurations, and the y-axis corresponds to the overall cost. We do not see significant differences over the four INCREASING DEPENDENCY 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 conflicts 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 INCREASING DEPENDENCY configurations. The reduction of average cost by considering plans of all other robots is significant,

Fig. 17 Planning for a three-robot system (overall cost under four configurations of INCREASING DEPENDENCY): $R1$, $R2$ and $R3$ need to move from $d1$ to $d9$, from $d6$ to $d4$, and from $d10$ to $d8$ (*cor1* side) respectively

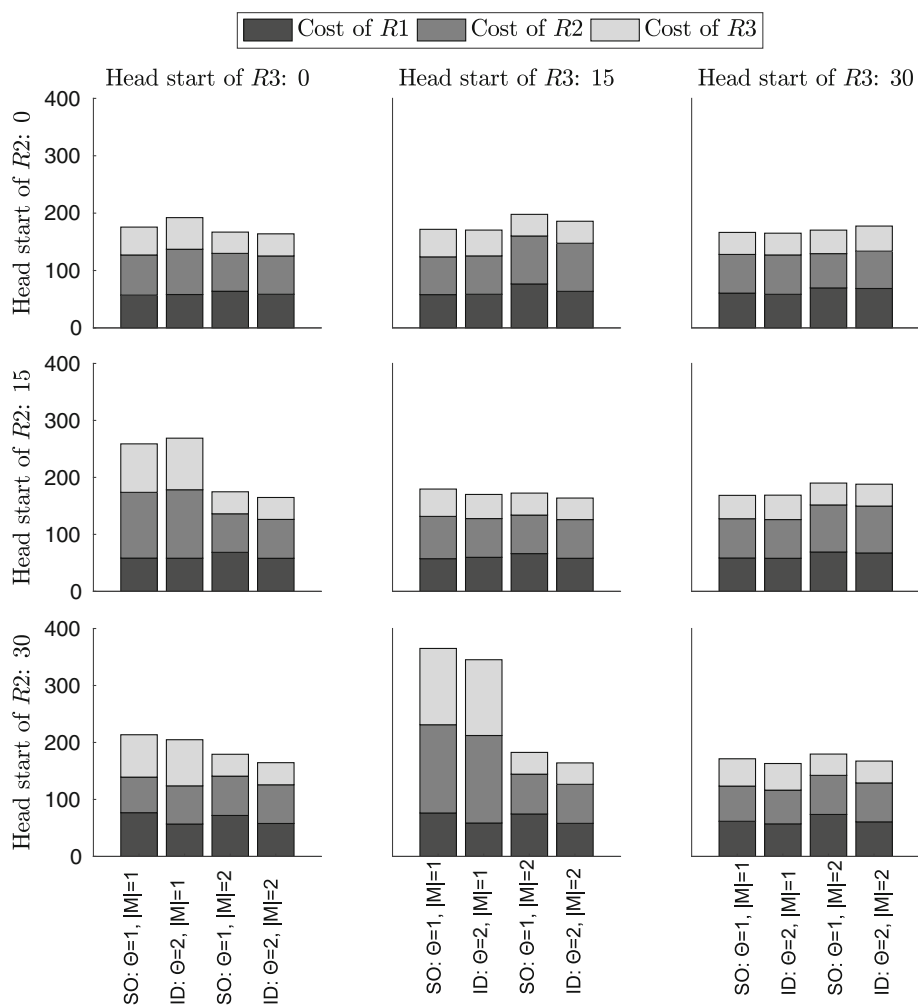


Table 2 Mean and standard deviation values of the four configurations in Fig. 17. Given $|M| = 2$, the average overall cost using INCREASING DEPENDENCY is significantly different from that of SINGLE ORDER (v -value=0.0128)

	Number of teammates considered in conditional planning	
	$ M = 1$	$ M = 2$
SO : $\Theta = 1$	207.81 (66.24)	179.28 (9.81)
ID : $\Theta = 2$	205.35 (62.25)	171.03 (9.99)

regardless of Θ 's value: p -value=0.03 when $\Theta = 1$, and p -value=0.02 when $\Theta = 2$. Given all other robots are considered in conditional planning, INCREASING DEPENDENCY performs significantly better than SINGLE ORDER (bold font). However, when only one other robot is considered, we do not see a significant difference between INCREASING DEPENDENCY and SINGLE ORDER (the left two columns). Therefore, INCREASING DEPENDENCY 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 INCREASING DEPENDENCY 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.

Figure 18 shows a picture of $R2$ (robot on the right) waiting to follow $R1$ (robot on the left) through door $D1$, using INCREASING DEPENDENCY.

A video of this trial is available at: <https://youtu.be/ADbH3sppLHQ>.

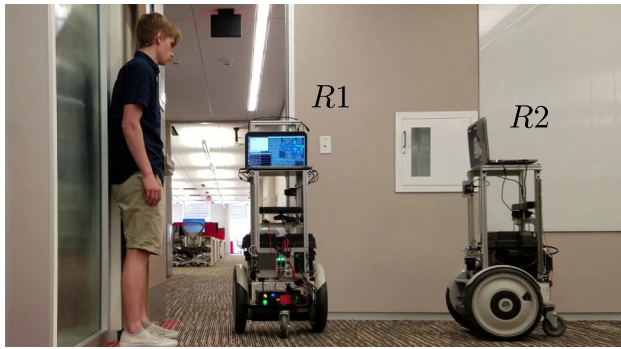


Fig. 18 Using INCREASING DEPENDENCY, two robots construct action synergy by sharing a door-opening action: robot R1 asks for help from a human for opening the door and is executing the *gothrough* action, while robot R2 is waiting to follow R1 through the door

7 Conclusions and future work

We introduce the multi-robot planning with conflicts and synergies (MRPCS) problem, and develop a novel, iterative inter-dependent planning (IIDP) framework and algorithms. We propose two algorithms for IIDP, INCREASING DEPENDENCY and BEST ALTERNATIVE. We test the algorithms in an abstract domain which enables scaling to a large number of robots. We also instantiate IIDP on a multi-robot navigation problem with temporal uncertainty, where we introduce a *shifted Poisson* distribution to represent robot navigation delays and present a novel algorithm for computing conditional plan cost. In experiments, we observe that IIDP algorithms bring significant improvements against baselines where robots do not coordinate their plans, or coordinate but do not model temporal uncertainty, and INCREASING DEPENDENCY enables more collaborative synergies and less conflicts, compared to INDEPENDENT PLANNING and SINGLE ORDER. Finally, we implement IIDP on real robots and present a demo where the robots share plans to realize a synergy.

There are a number of ways to make improvements in this line of research. For instance, Sect. 3 includes a list of assumptions, which might not hold in some domains. A relatively strong assumption made in this article is that the robots can reliably communicate among the team members with zero cost. Future work can look into scenarios where robot communications are unreliable, delayed, or expensive. The single-robot, single-task assumption, as described in Sect. 3, can be removed in future work, e.g., to consider tasks that require two robots to move a box that is too heavy for individuals. The evaluation of our IIDP framework in the real world was primarily using a team of mobile robots conducting navigation tasks. There are a number of other multi-robot tasks that can be used for evaluations, such as human guidance, dual-arm manipulation, and object delivery. The developed IIDP framework

was evaluated only using a homogeneous team of robots. After implementing and evaluating the IIDP framework with homogeneous teams, natural follow-up work is to apply the developed IIDP framework to heterogeneous teams or human-robot teams. The communication cost of human-robot teams will likely be much higher than multi-robot systems. However, given the increasing presence of collaborative robots in everyday environments, the potential impact of applying IIDP to human-robot collaboration domains will be significant.

Acknowledgements This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-1637736, IIS-1651089, IIS-1724157), the Office of Naval Research (N00014-18-2243), Future of Life Institute (RFP2-000), DARPA, Intel, Raytheon, 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.

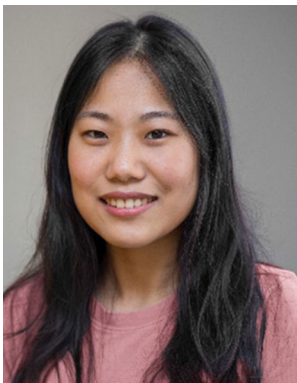
References

- Alonso-Mora, J., DeCastro, J. A., Raman, V., Rus, D., & Kress-Gazit, H. (2018). Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles. *Autonomous Robots*, 42(4), 801–824.
- Alur, R., Moarref, S., & Topcu, U. (2013). Counter-strategy guided refinement of $gr(1)$ temporal logic specifications. In *Formal Methods in Computer-Aided Design (FMCAD)*, IEEE, 2013, (pp. 26–33).
- Amato, C., Konidaris, G., Cruz, G., Maynor, C. A., How, J. P., & Kaelbling, L. P. (2015). Planning for decentralized control of multiple robots under uncertainty. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, (pp. 1241–1248).
- Boutillier, C., & Brafman, R. I. (2001). Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14, 105–136.
- Brafman, R. I., & Domshlak, C. (2008). From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS*, 28–35.
- Brenner, M. (2003). A multiagent planning language. In *Proceedings of the Workshop on PDDL, ICAPS*, Vol. 3, (pp. 33–38).
- Brooks, J., Reed, E., Gruver, A., & Boerkoel Jr., J. C. (2015). Robustness in probabilistic temporal planning. In *National Conference on Artificial Intelligence (AAAI)*.
- Brucker, P. (2007). *Scheduling algorithms*. Berlin: Springer.
- Buehler, J., & Pagnucco, M. (2014). A framework for task planning in heterogeneous multi robot systems based on robot capabilities. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- Coltin, B., & Veloso, M. (2014). Scheduling for transfers in pickup and delivery problems with very large neighborhood search. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- Crosby, M., Rovatsos, M., & Petrick, R. P. (2013). Automated agent decomposition for classical planning. In *ICAPS*, 46–54.
- Dresner, K., & Stone, P. (2008). A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31, 591–656.
- Fentanes, J. P., Lacerda, B., Krajnik, T., Hawes, N., & Hanheide, M. (2015). Now or later? predicting and maximising success of nav-

- igation actions from long-term experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, (pp. 1112–1117).
- Ferreira, P. R., dos Santos, F., Bazzan, A. L. C., Epstein, D., & Waskow, S. J. (2009). Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies. *Autonomous Agents and Multi-Agent Systems*, 20, 421–443.
- Fikes, R. E., & Nilsson, N. J. (1972). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3), 189–208.
- Filippidis, I., Dimarogonas, D. V., & Kyriakopoulos, K. J. (2012). Decentralized multi-agent control from local ltl specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, (pp. 6235–6240).
- Gebser, M., Kaminski, R., Kaufmann, B., & Schaub, T. (2014). Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694.
- Gelfond, M., & Lifschitz, V. (1998). Action languages. *Electronic Transactions on Artificial Intelligence*, 3, 195–210.
- Gerkey, B. P., & Mataric, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research (IJRR)*, 23(9), 939–954.
- Ghallab, M., Knoblock, C., Wilkins, D., Barrett, A., Christianson, D., Friedman, M., Kwok, C., Golden, K., Penberthy, S., Smith, D. E., Ying, S., Weld, D. (1998). Pddl-the planning domain definition language. 501–510.
- Guo, X., & Hernández-Lerma, O. (2009). *Continuous-time Markov decision processes*. Berlin: Springer.
- Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, 26, 191–246.
- Hoang, K. D., Fioretto, F., Hou, P., Yokoo, M., Yeoh, W., Zivan, R. (2016). Proactive dynamic distributed constraint optimization. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*. International Foundation for Autonomous Agents and Multiagent Systems, (pp. 597–605).
- Hoffmann, J., & Nebel, B. (2001). The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14, 253–302.
- Hönig, W., Kumar, T. S., Cohen, L., Ma, H., Xu, H., Ayanian, N., & Koenig, S. (2016). Multi-agent path finding with kinematic constraints. In *ICAPS*, 477–485.
- Jain, M., Taylor, M. E., Yokoo, M., & Tambe, M. (2009). DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*.
- Khandelwal, P., Barrett, S., & Stone, P. (2015). Leading the way: An efficient multi-robot guidance system. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, (pp. 1625–1633).
- Khandelwal, P., Yang, F., Leonetti, M., Lifschitz, V., & Stone, P. (2014). Planning in Action Language BC while Learning Action Costs for Mobile Robots. In *International Conference on Automated Planning and Scheduling (ICAPS)*.
- Khandelwal, P., Zhang, S., Sinapov, J., Leonetti, M., Thomason, J., Yang, F., Gori, I., Svetlik, M., Khante, P., & Lifschitz, V. et al. (2017) Bwibots: A platform for bridging the gap between ai and human-robot interaction research. *The International Journal of Robotics Research*.
- Knepper, R. A., Layton, T., Romanishin, J., & Rus, D. (2013). Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, (pp. 855–862).
- Knill, O. (1994). *Probability and stochastic processes with applications*. Overseas Press.
- Koenig, N., & Howard, A. (2004). 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*, Vol. 3, pp. 2149–2154.
- Komenda, A., et al. (2016). Privacy-concerned multiagent planning. *Knowledge and Information Systems*, 48(3), 581–618.
- Korsah, G. A., Stentz, A., & Dias, M. B. (2013). A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12), 1495–1512.
- Kovács, D. L. (2012). A multi-agent extension of pddl3. 1.
- Lee, J., Lifschitz, V., & Yang, F. (2013). Action language bc: Preliminary report. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pp. 983–989. AAAI Press.
- Lesser, V., Ortiz, C. L., Jr., & Tambe, M. (2012). *Distributed sensor networks: A multiagent perspective* (Vol. 9). Berlin: Springer.
- Ma, H., & Koenig, S. (May 2016). Optimal target assignment and path finding for teams of agents. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.
- Maheswaran, R. T., Pearce, J. P., & Tambe, M. (2004). Distributed algorithms for dcop: A graphical-game-based approach. In *Parallel and Distributed Computing Systems (PDCS)*, (pp. 432–439).
- Matsui, T., Matsuo, H., Silaghi, M., Hirayama, K., & Yokoo, M. (2008). Resource constrained distributed constraint optimization with virtual variables. In *AAAI*, (pp. 120–125).
- Mausam, & Weld, D. S. (2008). Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research*, 31, 33–82.
- Modi, P. J., Shen, W.-M., Tambe, M., & Yokoo, M. (2003). An asynchronous complete method for distributed constraint optimization. *AAMAS*, 3, 161–168.
- Nilsson, N. J. (1984). *Shakey the robot*. Technical report, SRI INTERNATIONAL MENLO PARK CA.
- Oglietti, M., Cesta, A. (2004). Cstrips: Towards explicit concurrent planning. In *Proceedings of the 3rd Italian WS on Plan. and Sched., 9th National Symposium of Association Italiana per l'Int. Artif*, (pp. 1–13).
- Raman, V., Piterman, N., & Kress-Gazit, H. (2013). Provably correct continuous control for high-level robot behaviors with actions of arbitrary execution durations. In *Robotics and Automation (ICRA), 2013 IEEE International Conference*, (pp. 4075–4081).
- Schäpers, B., Niemueller, T., Lakemeyer, G., Gebser, M., & Schaub, T. (2018). Asp-based time-bounded planning for logistics robots. In *ICAPS*, (pp. 509–517).
- Sharon, G., Stern, R., Felner, A., & Sturtevant, N. R. (2015). Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219, 40–66.
- Sharon, G., Stern, R., Goldenberg, M., & Felner, A. (2013). The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195, 470–495.
- Smith, D. E., & Weld, D. S. (1999). Temporal planning with mutual exclusion reasoning. *IJCAI*, 99, 326–337.
- Tsugawa, S., & Kato, S. (2010). Energy its: another application of vehicular communications. *IEEE Communications Magazine*, 48(11), 120–126.
- Turpin, M., Michael, N., & Kumar, V. (2014). Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1), 98–112.
- Veloso, M. M., Biswas, J., Coltin, B., & Rosenthal, S. (2015). CoBots: Robust symbiotic autonomous mobile service robots. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*.
- Wagner, G., & Choset, H. (2015). Subdimensional expansion for multi-robot path planning. *Artificial Intelligence*, 219, 1–24.
- Wong, K. W., & Kress-Gazit, H. (2015). 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*, (pp. 339–345).

- Wong, K. W., & Kress-Gazit, H. (2016). Need-based coordination for decentralized high-level robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Yedidsion, H., & Zivan, R. (2016). Applying `dcop_mst` to a team of mobile robots with directional sensing abilities. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1357–1358. International Foundation for Autonomous Agents and Multiagent Systems.
- Younes, H. L., & Simmons, R. G. (2004). Solving generalized semi-markov decision processes using continuous phase-type distributions. In *The AAAI Conference on Artificial Intelligence*.
- Zhang, S., Jiang, Y., Sharon, G., & Stone, P. (2017). Multirobot symbolic planning under temporal uncertainty. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 501–510. International Foundation for Autonomous Agents and Multiagent Systems.
- Zhang, S., Sridharan, M., & Washington, C. (2013). Active visual planning for mobile robot teams using hierarchical POMDPs. *IEEE Transactions on Robotics*, 29(4), 975–985.
- Zhang, S., Yang, F., Khandelwal, P., & Stone, P. (September 2015). 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.
- Zhang, Y., Parker, L. E. (2013). Multi-robot task scheduling. In *IEEE International Conference on Robotics and Automation (ICRA)*, (pp. 2992–2998).
- Zivan, R., Yedidsion, H., Okamoto, S., Grinton, R., & Sycara, K. P. (2015). Distributed constraint optimization for teams of mobile sensing agents. *Journal of Autonomous Agents and Multi-Agent Systems*, 29, 495–536.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Yuqian Jiang is a Ph.D. student in the Computer Science Department at the University of Texas at Austin. Her main research interests are robotics and AI, with specific interests in planning and learning techniques that support service robot architectures. Yuqian received B.S. degrees in Computer Science and Mathematics from UT Austin in 2017. Her undergraduate honors thesis first compares the performance of two task planners of different formalisms in a robot navigation

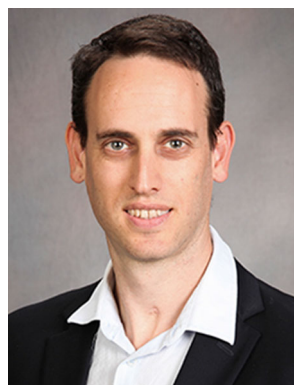
domain, and then presents an algorithm to efficiently plan for a team of mobile robots while minimizing total expected costs.



Harel Yedidsion is currently a postdoctoral researcher in the Computer Science Department at the University of Texas at Austin. His main research interest is how to design intelligent multiagent systems such as teams of robots that can efficiently cooperate to solve tasks in a dynamic environment. For this end he develops both central and distributed algorithms for multiagent coordination. His work also touches on Robot Perception, Manipulation, Navigation, Human Robot Interaction, Natural Language Processing, Planning, Game Theory and Reinforcement Learning.



Shiqi Zhang is an Assistant Professor at the Department of Computer Science, The State University of New York (SUNY) at Binghamton. From 2016 to 2018, he was an Assistant Professor with Cleveland State University. From 2014 to 2016, he was a Postdoctoral Fellow at the University of Texas at Austin. He received his Ph.D. in Computer Science (2013) from Texas Tech University. Before that, he received his Master's (2008) and BS (2006) from Harbin Institute of Technology in China. He was a visiting student at Tsinghua University in 2007 and 2008, and interned at Microsoft Research Asia in 2012. Dr. Zhang's research lies in the intersection of artificial intelligence and robotics. He is particularly interested in developing algorithms that integrate computational modalities of planning, reasoning, and learning for service robots that work in human-inhabited, everyday environments, such as homes, hospitals and offices.



Guni Sharon is an assistant professor in the department of computer science and engineering (CSE) at Texas A&M University. He received his doctoral, master's and bachelor's degrees in information systems engineering from Ben-Gurion University. Sharon's current work focuses on developing and applying artificial intelligence techniques for optimizing transportation networks. Prior to joining Texas A&M, he was a postdoc in the computer science department at University of Texas at Austin. He is the recipient of the Outstanding Paper Award from the Association for the Advancement of Artificial Intelligence (AAAI).



Peter Stone is the David Bruton, Jr. Centennial Professor and Associate Chair of Computer Science, as well as Chair of the Robotics Portfolio Program, at the University of Texas at Austin. Professor Stone's research interests in Artificial Intelligence include machine learning (especially reinforcement learning), multiagent systems, robotics, and e-commerce. Professor Stone received his Ph.D in Computer Science in 1998 from Carnegie Mellon University. From 1999 to

2002 he was a Senior Technical Staff Member in the Artificial Intelligence Principles Research Department at AT&T Labs - Research. He is an Alfred P. Sloan Research Fellow, Guggenheim Fellow, AAAI Fellow, IEEE Fellow, AAAS Fellow, Fulbright Scholar, and 2004 ONR Young Investigator. In 2007 he received the prestigious IJCAI Computers and Thought Award, given biannually to the top AI researcher under the age of 35, and in 2016 he was awarded the ACM/SIGAI Autonomous Agents Research Award. Professor Stone co-founded Cogitai, Inc., a startup company focused on continual learning, in 2015, and currently serves as President and COO.