Multi-Robot Human Guidance: Human Experiments and Multiple Concurrent Requests

Piyush Khandelwal[†,‡]
[†]Cogitai Inc.
683 S Glenhurst Dr
Anaheim, CA 92808, USA
piyushk@{cogitai.com,cs.utexas.edu}

Peter Stone[‡]
[‡]Department of Computer Science
University of Texas at Austin
Austin, TX 78712, USA
pstone@cs.utexas.edu

ABSTRACT

In the multi-robot human guidance problem, a centralized controller makes use of multiple robots to provide navigational assistance to a human in order to reach a goal location. Previous work used Markov Decision Processes (MDPs) to construct a formalization for this problem [13], and evaluated this framework in an abstract setting only, i.e. without experiments using high-fidelity simulators or real humans. Additionally, it was unable to handle multiple concurrent requests and did not consider buildings with multiple floors. The main contribution of this paper is the introduction of an extended MDP framework for the multi-robot human guidance problem, and its application using a realistic 3D simulation environment and a real multi-robot system. The MDP formulation presented in this paper includes support for planning for multiple guidance requests concurrently as well as requests that require a human to traverse multiple floors. We evaluate this system using real humans controlling simulated avatars, and provide a video demonstration of the system implemented on real robots.

Keywords

Multi-Robot Planning; MDP; MCTS; UCT

1. INTRODUCTION

Recent progress in the development of service robots is making it increasingly possible to deploy multiple robots in a building to perform work such as maintenance, delivery, and building security. Given that many robots may be ubiquitously present in the environment performing such routine activities, it is becoming realistic to envision a scenario in which a human approaches a robot and asks for help on a new task. Such new tasks may require coordination between multiple robots, and may have to be done in parallel and without neglecting the robots' background/routine tasks. An automated system ought to help the human complete this new task, while ensuring that robots are minimally deviated from their duties. In this paper, we study the problem where this new task is a human asking for navigational assistance, and how the automated system can use multiple robots to efficiently guide the human to his goal.

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

This problem, termed the multi-robot human guidance problem, has been formulated as a Markov Decision Process(MDP) [13]. Using this MDP, an automated system can allow the human to follow a robot all the way from the start location to his goal. Alternatively, the system may choose to use one robot to instruct the human to walk ahead in a particular direction, while a second robot can prepare to provide subsequent navigational assistance further ahead. If such handoffs can be correctly executed, there are two main advantages. First, a human's ability to navigate crowded building environments may be far superior to a robot's, allowing the human to reach his goal faster if he can complete part of the route himself. Second, robots are deviated for less time from their background duties to help the human.

MDPs are well suited to handle such a formulation, as they can capture the uncertainty in the human's motion patterns in case the human misinterprets the instructions conveyed by a robot. The first contribution of this paper is an extended MDP framework that goes beyond previous work by incorporating multiple concurrent guidance requests and requests requiring traversal across multiple floors.

Additionally, such a problem cannot be preplanned for, as the possible combinations of robot, start, and goal locations can be intractably high for even a small building. At the same time, immediate action needs to be taken in order to allow the human to reach his goal quickly. For this reason, we use real-time probabilistic planning techniques to solve the multi-robot coordination problem. In this paper, we demonstrate that planning approaches based on Monte-Carlo Tree Search (MCTS) can be used to approximately solve the guidance MDP in real-time. Whereas past work has been evaluated in comparison to naive baselines, for the purpose of more meaningful evaluation, we introduce a new heuristic baseline adapted from the Pickup and Delivery Problem with Transfers (PDP-T) [7] to handoff the human from one robot to the next.

Since MDP's inevitably abstract away many details about the real world, it is necessary to implement this framework on a real system to ensure that human-robot coordination can be effectively solved in practice. In previous work, the multi-robot guidance system was neither implemented on real robots nor a high-fidelity simulation environment [13]. The third contribution of this paper is the instantiation of the MDP framework on a real multi-robot system and a realistic 3D simulation. We perform a user study in simulation, where real humans control simulated avatars, to demonstrate that planning in the abstract MDP domain does not differ substantially from a more realistic implementation.

2. RELATED WORK

Multi-robot coordination via decentralized Partially Observable MDPs (POMDPs) has been studied by auctioning different roles, under the assumption that each robot can be assigned a well-defined role or behavior [6]. MCTS has also been used to distribute a set of tasks among multiple robots [11]. In the multi-robot human guidance problem, there is no explicit demarcation of subtasks such that they can be easily distributed among robots.

Recent research has investigated how multiple real robots can coordinate to perform tasks such as delivery and box-pushing in small controlled environments using decentralized POMDPs with macro actions [1, 2]. In this prior work, actions are designed such that robots are controlled at a coarse granularity using the POMDP, which is similar to how we design actions in Section 3. In our work, centralized communication is possible, the state is assumed to be fully observable, and our approaches to solving the MDP are designed towards producing good solutions in real-time rather than the optimal solution. Consequently, the scale of problems solvable by our system is considerably larger.

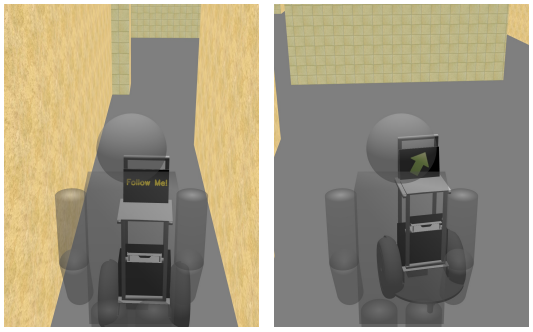
Both the Pickup and Delivery Problem with Transfers (PDP-T) [7] and the Dial-A-Ride Problem (DARP) [10] define a system of agents performing geographically distributed tasks, similar to the problem studied in this paper. While both these problems are typically defined at a larger scale than those dealt by our system, these problems assume that action outcomes are deterministic. In contrast, the MDP formulation described in this paper is more suitable for the stochasticity that arises from human-robot interaction.

While multi-robot guidance has not often been studied in previous literature, many systems have used single robots to guide people. For instance, robots have been used to lead tours [25] and provide navigational assistance to the elderly and visually impaired [18, 19]. Other mediums apart from robots have also been used to guide people in buildings. For instance, stationary booths and personal handheld devices have also been used to provide navigational instructions to humans [4, 5]. Our work differs in motivation from these previous works. We assume a system of robots is already in place performing routine duties, and guidance is an independent task that needs to be completed in parallel.

3. PROBLEM FORMULATION

An MDP can be expressed as $M = \langle S, A, P, R \rangle$ where S represents the environment's state space, $A(s)$ is the set of actions the system can take at state $s \in S$, P is the transition function that gives the transition probability of reaching a particular next state from a given state-action pair ($P : S \times A \times S \to \mathbb{R}$ such that $\sum_{s'} P(s, a, s') = 1$), and R is the reward received given a transition ($R : S \times A \times S \to \mathbb{R}$) [24]. A task inside this MDP is episodic with a start state where one or multiple people have just requested navigational assistance, and a set of termination states in which all persons have arrived at their goal locations. A task solution is represented as a policy $\pi(s) : S \to A_s$ that provides the action the system should take at state s.

When formulating an MDP, among the key representational decisions are how to represent the state and action spaces, which in turn determine the transition function. In the multi-robot human guidance problem, the main source of uncertainty is the human's motion. We thus construct the state and action spaces such that this motion can be captured as the stochastic transition function among states that can be influenced by robot actions. To this end, the MDP state in the multi-robot human guidance problem needs to keep track of locations of all the persons and robots in the environment, as well as the background tasks each robot needs to perform. Actions in this MDP allow the system to proactively move robots to help humans, as well as lead or direct humans to particular locations (see Figure 1). The reward function captures the two objectives of the multi-robot human guidance problem: (i) completing guidance requests quickly, and (ii) reducing diversions from background tasks.



(a) LEAD action (b) DIRECT action

Figure 1: The system directing a real human controlling a simulated avatar, as explained in Section 6. In Fig. 1a the system is using a robot to lead a person from one graph node to an adjacent one. Fig. 1b demonstrates how the system can direct the human to walk ahead by himself in a particular direction.

In a building environment, human and robot locations can be expressed as $\langle x, y, f \rangle$, where $x, y \in \mathbb{R}$ represent real world coordinates on floor f. We constrain possible locations by expressing them using a topological graph $g = \langle N_g, E_g \rangle$. The set of nodes N_g represent key locations in the environment where humans are likely to make navigational choices, and the set of edges E_g connect neighboring nodes. Figure 2a illustrates the graph for a simple environment. Given g, physical locations can be projected on edge e_{uv} from node u to node v to produce a location tuple $l_{uvp} = \langle u, v, p \rangle$ where $p \in [0, 1]$ represents how far along edge e_{uv} the projection lies. If $p = 0$, then the object projects exactly at node u, and if $p = 1$ then the object projects exactly at node v.

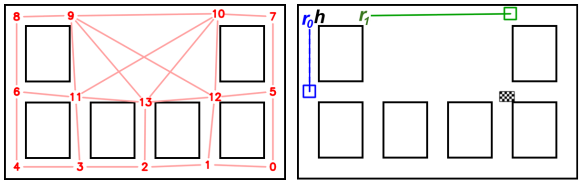
Next, we build on the MDP described in previous work [13] and formalize the multi-robot human guidance problem with support for multiple concurrent requests and multiple floors.

3.1 State Representation

The MDP state uses a factored representation. A state s can be expressed using many different sets of factors:

$$s = s_{independent} \times s_{guidance_0} \times \cdots \times s_{guidance_n}.$$

Here, $s_{independent}$ tracks various factors of the domain that are independent from each individual guidance request, such as information about each robot's location, the background task each robot needs to perform, and whether a robot is currently assisting a human instead of performing its background task. $s_{guidance_i}$ captures information specific to ith guidance request being concurrently addressed, and n represents the number of guidance requests



(a) Topological repr. (b) MDP State repr.

Figure 2: Topological representation for a single-floored environment and an example of a single-request MDP state. The human's location is marked by a black h, and robots are marked as r_0 and r_1. The human's goal (node 12) is marked by a checkered flag, and the squares mark the location of background task for both robots (nodes 6 and 10).

Formally, $s_{independent}$ comprises the following information for every robot r, labeled as continuous (C) or discrete (D):

- l_u (D), l_v (D), and l_p (C) represent robot r's topological graph location, and can take the following values:

$$l_u \in N_g, l_v \in N_g, \text{ and } l_p \in [0,1].$$

- $\tau_d \in N_g$ (D) is the location of the background task τ that the robot needs to perform.

- τ_T (C) is the total time the robot needs to spend at τ_d to complete background task τ.

- τ_t (C) is the time the robot has already spent at τ_d. It can take any value less than τ_T, and is initialized to 0 whenever a new background task τ is assigned to robot r once the previous one is completed.

- h (D) represents if the robot has been diverted to help a guidance request, and can take the following values:

$$h \in \{\text{NONE}\} \cup N_g,$$

  where NONE represents that the robot has not been assigned and $h \in N_g$ represents that the robot should move to h to help with a guidance request.

- *preemptible* (D) represents whether the robot can be reassigned. A robot in the midst of leading a person from graph node u to adjacent graph node v cannot be reassigned until it reaches graph node v.

$s_{guidance_i}$ comprises request-specific environment factors:

- $loc_v \in N_g$ (D), $loc_u \in N_g$ (D), and $loc_p \in [0,1]$ (C) represent a person's graph location (for request i), where loc_v is always the person's current location and loc_u is the person's previous location.

- *a.type* (D) represents whether any assistance was provided to the human at his current location loc_v by a colocated robot. Assistance can only be provided when the person is exactly at loc_v, i.e. $loc_p = 1$. Furthermore, *a.type* can take values in {NONE, LEAD, DIRECT}. LEAD specifies that the person should follow the colocated robot, and DIRECT specifies that the person should walk ahead by himself in the direction of an adjacent graph node.

- *a.loc* (D) is a location adjacent to the human's current location *loc* that the system has advised the human to move to, and can take values in:

$$\begin{cases} \text{NONE} & a.type = \text{NONE} \\ v : v \in adjacentNodes(loc_v) & a.type \neq \text{NONE} \end{cases}$$

  A special case for $\langle a.type, a.loc \rangle$ is $\langle \text{LEAD}, loc_v \rangle$, which specifies that both the colocated robot and the human should wait at location loc_v.

- *goal* represents the human's goal location.

3.2 Action Representation

The actions in this MDP are defined at a high level of control, and it is assumed that robots have control mechanisms for navigating to a location in the environment. Actions are taken whenever any human completes a transition to loc_v, and consequently the action duration is not constant and the MDP formulation is semi-Markov [9]. At any given state, there are many robots that the system needs to control concurrently. For instance, at the state represented by Figure 2b, the system can execute an action where robot r_0 directs the human to node 9, and robot r_1 is assigned to stay at node 9 to wait for the human. Once the human reaches node 9, the system can take a second action where robot r_1 leads the human to his destination at node 12.

As expressed, these actions require coordinating multiple robots, and can contain one of the following elements:

- *AssignRobot*(r, v) $(v \in N_g)$ – Deviate robot r from its current background task and start navigation to any node v to assist in a guidance request at a future time.

- *ReleaseRobot*(r) – Release a previously assigned robot r, and allow it to return to its background task.

- *DirectHuman*(i, r, v) – Use robot r to direct a colocated human (represented by request i) to node v adjacent to the human's (and robot's) location $guidance_i.loc_v$. This action is depicted in Figure 1b.

- *LeadHuman*(i, r, v) – Use robot r to lead a colocated human (represented by request i) to adjacent node v (depicted in Figure 1a). In the special case when $v = guidance_i.loc_v$, robot r asks the human to wait at the current location for a fixed amount of time. The robot can use this time to work on a background task at the current location.

For the two actions presented in the previous paragraph, the first action equates to executing [*DirectHuman*(0,0,9), *AssignRobot*(1,9)] when the human is at node 8, and [*LeadHuman*(1,0,12)] once the human reaches node 9. In the unlikely situation that the human chooses to move to node 6 instead of node 9 despite being directed to do so, r_1 does not need to stay at node 9 indefinitely. The system can execute *ReleaseRobot*(1) as one of the constituents of the next action to release robot r_1 back to its background task, and execute a different policy to help the human reach the goal.

It is important to note that not all combinations of action elements constitute a valid action. For instance, the system cannot execute [*LeadHuman*(0,0,9), *AssignRobot*(0,6)] at the first state, as robot r_0 cannot move to nodes 6 and 9 at the same time. A single robot can either be (re)assigned a new location to aid a guidance request, lead a colocated

human to a particular location, or be released from a previous assignment. Similarly, a robot can either direct or lead a colocated human, but not both. And finally, action elements are expressed in a canonical order such that no two sequences of elements can lead to the same action.

One of the main challenges of the action representation presented above is that the number of possible actions at a given state is extremely high. For instance, at the state represented by Figure 2b, the number of possible actions is 720.[1] To reduce this branching, we use operator decomposition [22] to split various concurrent elements of an action into sequential actions. Decomposition decouples different actions and allows simulation-based search (see Section 4.3) to quickly prune useless action elements without considering all possible combinations that include that element.

3.3 Transition Function

Each constituent element of the action space induces some deterministic changes in the domain, which are outlined in Table 1. Once the deterministic changes from a given action sequence have been executed, the system waits for time to pass. During this transition, changes to the state occur from three sources (apart from those mentioned in Table 1):

1. Robots that complete background tasks are assigned new ones. The queue of background tasks is known apriori, resulting in a deterministic state change.

2. Robots and humans move around in the environment. This motion is inherently non-deterministic since navigation speeds can be variable, and robots and humans may take longer if they need to navigate around obstacles.

   We ignore this non-determinism and assume deterministic motion on the part of robots and humans, as it constrains the set of continuous variables in the state representation to a discrete set of values. Consequently, simulation-based planners converge faster as the same states get repeated more often since this non-deterministic branching is ignored. In this model, humans move at an average speed of $\bar{v_h}$, and robots move at an average speed of $\bar{v_r}$. We discuss the drawbacks of ignoring this non-determinism while considering more realistic implementations in Section 6.

3. It is also possible that humans may misinterpret directions from robots. For instance, for the situation depicted in Figure 1b, should the robot giving advice be misoriented from its estimate of its own location, it may direct the human to the path on the left instead of the intended path on the right. The transition model needs to account for such non-determinism.

   In this model, we assume that when a robot leads a human to an adjacent location (Figure 1a), or waits with the human at the current location, the transition is always deterministic. If a robot directs the human, or no robot is present at a human's location, the model assumes the human transitions to an adjacent graph node non-deterministically, with the most likely outcome being the direction intended by the robot, or the human's previous direction of motion, respectively (formal details in [13]). If a robot directs the human to a different floor, the robot displays the message "Please proceed to floor X" to the human instead of an arrow, and this transition is assumed to be deterministic.

From this description of the transition function, it is evident that we have made many design approximations in the construction of this MDP. In Section 6, we demonstrate that despite hand-coding the transition function, performance metrics gathered using a multi-robot guidance system implemented in a high fidelity simulation environment (with real humans controlling simulated avatars) follow similar trends to those observed when using these approximations.

3.4 Reward

The reward function needs to balance the time taken to complete guidance tasks against the time for which robots are deviated from their background task. The overall reward function can be expressed as a linear combination of individual rewards as follows:

$$R_{ss'} = -\sum_{i=0}^{n} u_i \Delta t - \sum_{r} U^r_{ss'},$$

where n is the number of ongoing guidance requests, Δt is time elapsed during the transition from state s to state s', u_i is the utility of performing guidance task i.

The second term in the above expression captures the cost of deviating robots from their background tasks, and it can be computed using the deterministic motion model of robots as follows. The utility loss for a robot r diverted from its background task τ (with average utility $\bar{\tau}_u$) is dependent on the amount of time the robot is delayed in reaching task location τ_d:

$$U^r_{ss'} = \bar{\tau}_u(\mathit{timeDest}(r_{s'}, r_s.\tau_d) + \Delta t - \mathit{timeDest}(r_s, r_s.\tau_d)),$$

where $\mathit{timeDest}(r_s, r_s.\tau_d)$ is the time robot r needs to reach τ_d from state s in which robot r is located at $r_s.\langle l_u, l_v, l_p\rangle$, and can be computed using average robot speed $\bar{v}_r$.

Now that we have described all the elements of the MDP, in the next section we describe how this MDP can be approximately solved using various approaches, and present an empirical comparison of these approaches.

4. MDP SOLUTION APPROACHES

In this section, we present three main approaches along with a number of variants to generate a policy for guiding humans given an MDP as formulated in the previous section. The first two approaches are intuitive heuristics for solving the MDP, and only make use of actions in the domain that have deterministic outcomes in order to ensure that humans reach their destination. The third approach, based on MCTS, makes use of simulation-based search to find actions that maximize the cumulative reward estimate.

4.1 Single Robot Approach

In the guidance MDP, it is assumed that for every human requesting assistance, a human approaches a robot performing a background task at a location. One straightforward solution is to use that robot to lead the human all the way to

[1]Either robot can be assigned to one of 14 locations or not assigned (15 choices per robot). At the same time, r_0 can direct the human to nodes 6 or 9, or not direct the human ($15^2 \times 3$ choices). Alternatively, r_0 can lead the human to nodes 6 or 9, or wait at node 8, and r_1 is the only robot that can be assigned (15×3 choices). The total number of actions is thus $15^2 \times 3 + 15 \times 3 = 720$.

Action	Next State $s' = s$ except:
AssignRobot(r,v)	$s'.robot_r.h = u$
ReleaseRobot(r)	$s'.robot_r.h = \text{NONE}$
DirectHuman(i,r,v)	$s'.guidance_i.a.\langle type, loc\rangle = \langle \text{DIRECT}, v\rangle$
LeadHuman(i,r,v)	$s'.guidance_i.a.\langle type, loc\rangle = \langle \text{LEAD}, v\rangle$, $s'.robot_r.h = v$, $s'.robot_r.preemptible = \textbf{false}$.
$\varnothing$ (passage of time)	$s'.guidance_i.a.\langle type, loc\rangle = \langle \text{NONE}, \text{NONE}\rangle$, $\forall r : s'.robot_r.\langle h, preemptible\rangle = \begin{cases} \langle \text{NONE}, \textbf{true}\rangle & s.robot_r.preemptible = \textbf{false} \\ \langle s.robot_r.h, \textbf{true}\rangle & otherwise \end{cases}$, *and other deterministic and non-deterministic changes described in Section 3.3.*

Table 1: Transition functions for each action in the MDP. $\varnothing$ represents the action when time moves ahead in the domain, after all individual action elements in the sequence have been executed.

his goal. This heuristic can be implemented by taking a sequence of *LeadHuman* actions to the adjacent node along the shortest path to the human's goal. Once the human reaches the goal location, the robot returns back to the start location and continues its original background task. We term this approach the *SingleRobot* approach.

Sometimes, completing the robot's background task at the start location and then leading the human all the way to his goal can generate a higher reward than directly following the SingleRobot approach, for instance when the robot's next background task is in the same direction as the goal. This approach can be achieved by executing *LeadHuman* (with the current location as the argument) repeatedly at a request's start location until the background task is completed, after which the SingleRobot approach is followed. We term this approach as SingleRobot with Wait, or *SingleRobotW*.

4.2 PDP-T Approach

Previous work has looked at the Pickup and Delivery Problem with Transfers (PDP-T) [7] where a set of packages need to be delivered by multiple robots given specific delivery windows and robot capacity constraints. The problem is solved in a decentralized manner where each robot uses heuristics to decide on package transfers, and an auction mechanism is used to coordinate between multiple robots. While there are differences between the multi-robot guidance and PDP-T domains, we present a heuristic approach adapted from that work here.

Intuitively, this approach follows the same principle as the SingleRobot approach, except it attempts to exchange the robot leading the human should it yield a higher reward, using one of the following two handoff situations:

1. The robot leading the human and the second robot are at the same location, and the second robot can take over leading the human to the goal.

2. The robot leading the human reaches an elevator, and directs the human to another floor where the second robot is waiting to lead the human.

The pseudocode for the PDP-T approach is outlined in Algorithm 1. Similar to the SingleRobot approach, a robot is leading a human to the goal (Lines 3-5). The system attempts to find a free robot that is performing background tasks whose path to perform these tasks will intersect the human's path to the goal (Lines 8-10). If an intersection exists, the systems finds the policy with the highest cumulative reward that satisfies one of the two handoff situations (Line 11). Such a policy can perform better than leading the human all the way to goal using the original robot, and

Algorithm 1 The PDP-T approach

```
1: for all request i ∈ s.guidance do
2:     if transitionIncomplete(i) then cont. to next i
3:     r ← getColocatedRobotId(i)
4:     reward_best ← rewardForSingleRobotPolicy(i, r)
5:     path_r ← pathToGoalForSingleRobotPolicy(i, r)
6:     r_exchange ← null
7:     for all robot r' ∈ s.robots do
8:         if notFree(r') then cont. to next r'
9:         path_r' ← pathForFutureBackgroundTasks(r')
10:        if noIntersect(path_r, path'_r) then cont. to next r'
11:        reward_r' ← rewardForExchangePolicy(i, r, r')
12:        if reward_r' > reward_best then
13:            r_exchange ← r'
14:            reward_best ← reward_r'
15:    if r_exchange is not null then
16:        planExchange(i, r, r')
17:        setNotFree(r')
```

the system selects the policy among all robots that maximizes this reward (Lines 12-15). The system then plans and executes actions necessary to execute this policy (Lines 15-17).

4.3 MCTS Planning Approach

Using approximate planning algorithms such as MCTS to search through the space of available actions can help a system find a policy that produces a better cumulative reward than heuristic approaches in expectation [16]. Using a simulator of the MDP to generate samples, MCTS executes many planning simulations from the current state to compute estimates of the cumulative reward for taking different actions, while building a tree to guide search. In the multi-robot human guidance problem, the system can make many such planning simulations while humans are in midst of making transitions from one graph node to another.

Furthermore, MCTS can restrict search to more relevant regions of the state space using intelligent action selection schemes such as UCB1 [3], which is used in experiments presented in this paper. Whenever it encounters a state in the search tree it has not previously seen, it can bootstrap exploration using a default policy, such as the SingleRobot approach in the multi-robot human guidance problem. Whenever a simulation reaches a terminal state, or the length of the trajectory in the simulation reaches a prespecified length, the rewards observed along the simulation are backpropagated back to the tree root, informing the action selection process in the tree. In this paper, we use the MaxMCTS(λ) backup strategy [14], which promotes faster

convergence than standard Monte Carlo backups by biasing reward estimates quickly to good yet potentially suboptimal policies.

Unfortunately, action branching in the multi-robot human guidance MDP is still too high for MCTS to converge to policies that outperform the heuristic approaches. We reduce the action branching in the domain by eliminating certain actions using the following domain specific heuristics:

1. As explained in Section 3.2, robots can be assigned to any location $v \in N_g$. However, the path to any location $v \in N_g$ must pass through one of the one of the robot's adjacent locations. With this knowledge, we limit assignable locations to the robot's current or adjacent graph nodes. The system can move a robot to a non-adjacent location using multiple *AssignRobot* actions at subsequent timesteps.

2. The maximum number of robots that can be diverted to help humans in the environment is limited to the number of active guidance requests. For example, if the environment has 10 robots and the system is servicing 2 guidance requests, only 2 robots can lead humans or be assigned to a location at a given time.

3. A robot colocated with a human must provide some assistance to the human, in the form of a *LeadHuman* or *DirectHuman* action.

Using heuristics may remove an action necessary for optimality, but similar to biased backpropagation, it can speed up convergence to a good yet potentially suboptimal policy.

To demonstrate the importance of incorporating stochasticity during planning, we also evaluate MCTS with a planning model where each action with non-deterministic outcome has been determinized to its most likely outcome. MCTS-D trades planning with an inaccurate model for faster convergence of value estimates due to smaller branching within this inaccurate model. Finally, while MCTS planning simulations can be performed while the human is moving around in the environment to select future actions, the system does not have time to perform planning when initially approached by a human to perform a guidance task. Consequently, the system can either make the human *Wait* at the start location to perform planning, or start to *Lead* the human to the goal (as per the SingleRobot policy), and take MCTS planning actions subsequently. These variations in MCTS planning strategy lead to 4 variants that will be evaluated in the following section: MCTS(Wait), MCTS(Lead), MCTS-D(Wait), and MCTS-D(Lead).

5. EVALUATION

We first describe the experimental setup. Experiments were run on 2 different domain sizes (Figure 3). The small domain contains 1 floor and 5 robots, and the large domain contains 2 floors, 10 robots, and 3 elevators. We assume it takes a human 15 seconds to use the elevator to move from one floor to the other. Since real robots need considerable human assistance to enter and exit the elevator, in contrast, we assume a robot takes 30 seconds instead. At the start of each episode, the system needs to solve for either 1 or 2 concurrent requests. An episode terminates if all humans have reached their destination, or 300/600 seconds have elapsed for the small/large domain, respectively.

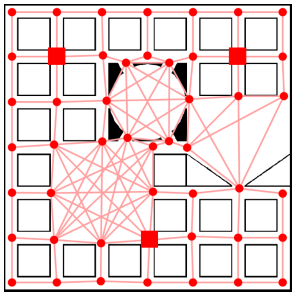


Figure 3: Small domain and graph (50m $\times$ 50m in size). The large domain contains two such floors, connected via 3 elevators marked by red squares.

Domain parameters required for evaluation were set as follows. Humans moved with an average speed ($\bar{v}_h$) of 1m/s. Robots moved with an average speed ($\bar{v}_r$) of 0.5m/s. The background task time across all tasks (τ_T) was set to 10s. The fixed amount of time for which a robot asks a colocated human to wait at the current location (the special case of the LeadHuman action) was set to 10s. The utility of guidance requests (u_i), was always set to 1 for all guidance requests.

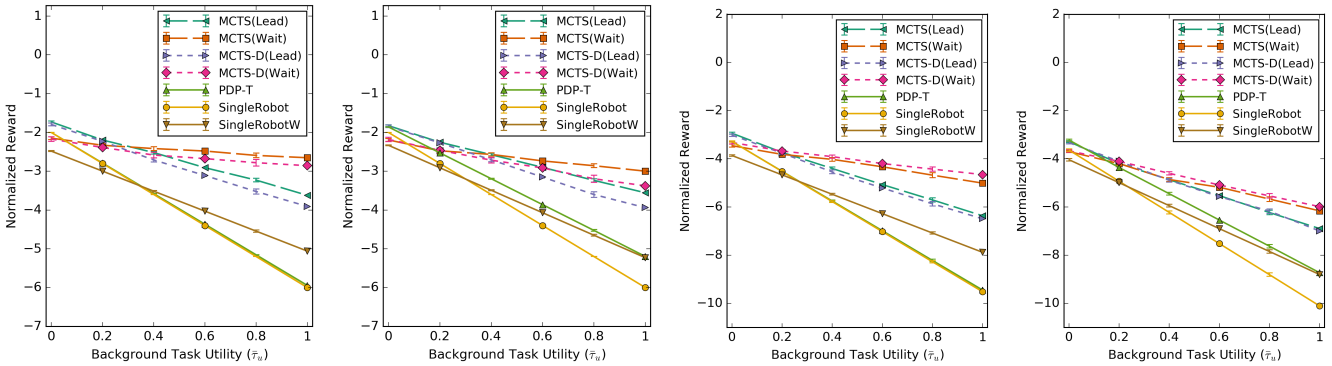
The average background task utility ($\bar{\tau}_u$) was varied to observe performance at different settings of the parameter. A value of 0 specifies that deviating robots from background tasks incurs no reward loss. A value in between 0 and 1 specifies that background tasks carry a weight less than that of guidance tasks. A value of 1 specifies that background and guidance tasks are equally weighted in the reward metric.

Since actions are taken whenever a human completes a transition to a graph node, the time in between successive actions can be computed. This computed time is provided to a single-threaded MCTS implementation for planning, emulating planning time available in a real-world system. MCTS planning trajectories were run up to a horizon of 150/300 seconds in the small/large domain, respectively.

All results have been averaged across 1,000 trials with randomly assigned start and goal locations. Since the distance from start to goal may be different for different trials, we normalize episode time and cumulative reward prior to aggregation. Each result is normalized by the minimal amount of time required for all humans in the domain to walk to their destination, given an average human speed ($\bar{v}_h$) of 1m/s.

Prior to evaluating all approaches, the λ bias parameter for the MaxMCTS(λ) backup strategy was tuned for MCTS variants. Performance was evaluated for $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ at 4 different combinations of domain size, number of concurrent guidance requests, and $\bar{\tau}_u$. The best average performance across these domain settings for MCTS(Wait), MCTS(Lead), MCTS-D(Wait), and MCTS-D(Lead) was achieved at λ equal to 0.4, 0.4, 0, and 0, respectively. These values of λ were then used for each corresponding MCTS variant throughout remaining experiments.

Figure 4 compares the performance of all baselines and MCTS variants under various problem configurations and difficulties. From these results, we can make the following observations. SingleRobot performs better than SingleRobotW only when the background task utility is low, since waiting to complete a low utility background task is a poor idea. PDP-T always performs equivalently or better than SingleRobot (as expected), and performs significantly better when multiple floors are present by directing people from one



(a) 1 Req, Small Domain (b) 1 Req, Large Domain (c) 2 Req, Small Domain (d) 2 Req, Large Domain

Figure 4: Performance of MCTS and Heuristic approaches as the number of concurrent requests, domain size, and background task utility are varied. A higher normalized value indicates better performance. Problem difficulty increases from left to right. Error bars represent standard error on reward performance.

floor to another instead of making a robot traverse through the elevator to lead a human.

Similar to SingleRobot and SingleRobotW, MCTS(Lead) and MCTS-D(Lead) outperform MCTS(Wait) and MCTS-D(Wait) at low background task utilities, respectively. At most domain settings (Figures 4a–4c), MCTS variants outperform corresponding MCTS-D variants, indicating the importance of incorporating the domain's stochasticity during planning. However, at the most difficult domain setting (Figure 4d), MCTS-D variants start performing better. At this difficulty, due to a much larger state-action space, faster convergence of value estimates in MCTS-D due to smaller branching in its planning model outweighs the detrimental effects of using an incorrect model.

In all but one domain configuration, an MCTS approach outperforms the best performing heuristic baseline. At $\bar{\tau}_u = 0$ in Figure 4d, the PDP-T baseline outperforms both MCTS Lead variants, as MCTS is unable to produce a policy that outperforms the elevator transfer approach used by PDP-T.

Overall, these results demonstrate that in most problem settings, MCTS planning can outperform domain-specific heuristic approaches. Furthermore, incorporating stochasticity in planning is necessary for improving performance, especially in smaller domains.

6. IMPLEMENTATIONS

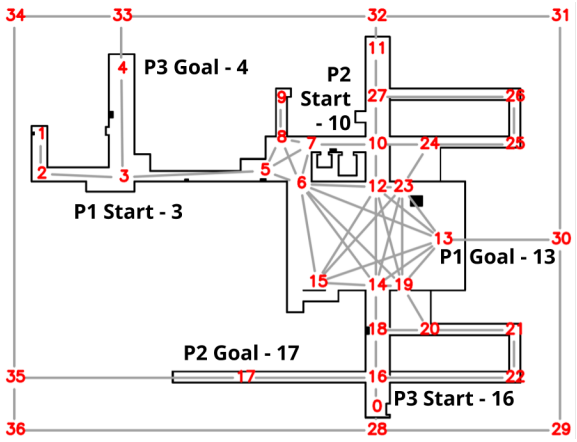
The multi-robot guidance framework was implemented on the BWIBot multi-robot system [15], which is built using ROS [20], GAZEBO 3D simulator [17], and ROCON multi-robot coordination framework [23]. Dijkstra's algorithm is used to find a path to a destination, and the robot moves along this path using the Elastic Bands approach [21]. This approach makes use of active contours [12] to execute local control that balances the straightness of the executed path with distance from obstacles. Consequently, robots do not move in straight lines along graph edges as expressed in the MDP. Monte Carlo Localization [8] is used by the robot to estimate it's own position, and there may be some uncertainty in this estimate. The visualization in Figure 1 has been generated using the 3D simulation implementation.

There are two main issues that need to be resolved in order to implement the multi-robot guidance MDP. The first pertains to the position and motion of robots and humans in the environment. The position of robots and humans need to be projected on the MDP topological representation during initialization, and this projection then needs to be constantly updated as entities move around. The topological representation does not account for the robot's orientation, and the speed of the robot provided to planning based approaches (0.5m/s) is less than its true linear speed (0.75m/s) to account for rotation time. Finally, as explained in Section 3.3, the transition function assumes a deterministic motion model, whereas the motion of entities in the real world is quite non-deterministic. As a result, when an MCTS variant is used, once a state transition occurs, it is necessary to map the real state to the closest one in the planning tree in order to determine the best action estimated via planning. When such a mapping cannot be made, the SingleRobot baseline is executed at the state instead.

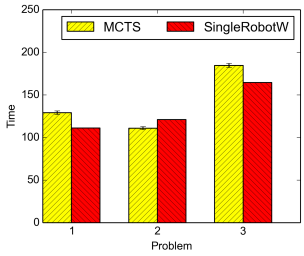
The second issue relates to the fact the model used to make human decisions, explained in Section 3.3, is different from real humans. In fact, it may be the case that real humans differ considerably from one another in behavior. Specifically, real humans tend to gravitate towards any robot in their field of view, which is not captured in the model presented in Section 3.3. A good direction for future work is the investigation of better or learned models for planning.

In this work, we mitigate this issue in part by changing how transitions in the MDP are implemented. Through initial empirical observations, it was noted that when a human transitions to a location and observes a robot approaching the same location, he typically waits for the robot to stop moving under the expectation of receiving instructions. Consequently, in the implementations, a human transition was only registered once any robots close to approaching the same location had completed navigation as well.

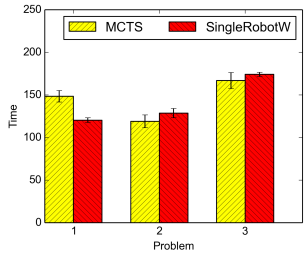
Figure 5a illustrates the map and topological representation (with 3 robots) used to construct a 3D simulation environment (based off a floorplan of a real building). While being reasonably realistic, this simulation environment lacks many artifacts present in the real world. For instance, a human being guided to a coffee shop or office can finish the navigation task on his own based on his own perception of the environment. Since the simulation environment does not contain artifacts to indicate that the human's destination was reached, the simulation implementation was con-



(a) User Study Map (80m × 60m) and Problems



(b) Predicted Times



(c) Actual Times

Figure 5: Figure 5a shows the map for the simulation environment and user study problems, and Figures 5b-5c show the model predicted and empirically determined times for completing each problem.

strained to ensure that a robot was present with the human upon reaching the goal to declare that the goal was reached.

We performed a user study using this simulation environment, with real humans controlling simulated avatars. The user study was limited to simulation due to the prohibitive difficulty of setting up a repeatable experiment in the real world where multiple robots are distributed across a building. While perception and control of the avatar in simulation is not realistic, the software modules for robot navigation, localization and user interface on the robot are the same in both simulation and the real multi-robot system. Thus, this user study can help evaluate the system in a semi-realistic setting where the model used for planning is inaccurate.

The goal of this user study is to establish that performance metrics gathered using a more realistic implementation still follow similar trends to those gathered using the methodology in Section 5, where planning was performed using an accurate model. Videos are available showcasing the system working both in simulation (`https://vid.me/fOvE`) and with real robots (`https://vid.me/Qghh`).

Each of the 22 participants performed 3 guidance problems (indicated in Figure 5a), with 11 participants performing the SingleRobotW baseline, and 11 participants performing the MCTS(Wait) approach. The following domain parameters were used ($\bar{v}_h = 1$m/s, $\bar{v}_r = 0.5$m/s, $\tau_T = 10$s, $u_i = 1$, and $\bar{\tau}_u = 0.5$). Only the SingleRobotW and MCTS(Wait) were evaluated in this section as these approaches perform best among all heuristic and MCTS approaches, respectively, at the selected domain parameters.

Since the MDP reward computation makes use of a model to estimate the utility loss of deviating robots from their background tasks (see Section 3.4), utility loss cannot be directly measured in simulation or the real world. In contrast, the time required by a human to reach his goal can be directly measured. For this reason, we compare completion times predicted by using the model (across 1,000 trials) and those from the user study (across 11 trials) in Figure 5. One instance of the SingleRobotW policy was excluded from results – the participant followed the robot at too far a distance to trigger state transitions in the MDP, and never attempted to move closer to the robot.

For Problems 1 and 2, the times from the actual case study match predicted times. In Problem 3, MCTS performed better than predicted, and we provide one observed discrepancy between prediction and reality. When directed by a robot at node 14 to walk towards node 6, the model predicts the most likely outcome as $[14 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 8 \rightarrow 6 \ldots]$. In reality, even if a human walked into the dead end at node 9, they were more likely to explore the corridor along nodes $5 \leftrightarrow 3$ themselves, which was the desired direction of motion for the goal at node 4. This discrepancy outlines a shortcoming of using a simple hand-coded model for human behavior, and more realistic models are necessary for planning to closely match real world performance.

7. DISCUSSION

This paper demonstrates that real-world multi-robot coordination problems requiring stochastic transitions can be successfully implemented at a scale of 5 to 10 robots, and MCTS can be used to approximately solve such problems in real-time. This paper also demonstrates that straightforward models of human behaviors are useful for solving the multi-robot human guidance MDP, but can also generate idiosyncratic behavior that does not reflect the behavior of real humans (Problem 3 in our user study). Future work should look at not only learning such models, but how to quickly select an appropriate model for a particular human if many such models are available.

We also demonstrated that the multi-robot human guidance problem can support multiple concurrent guidance requests, albeit making the policy search and planning problems more difficult. The relative improvement of MCTS planning over heuristic baselines is considerably less when 2 concurrent requests are present in the large domain (Figure 4d), compared to other problems (Figures 4a–4c). Overall, this paper successfully demonstrates that MCTS planning can be used to solve multi-robot coordination problems in real-time, giving performance much better than specialized heuristic baselines in most domain configurations.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287, IIS-1637736, IIS-1651089), ONR (21C184 - 01), AFOSR (FA9550-14-1-0087), Raytheon, Toyota, AT&T, Lockheed Martin, and Yujin Robot. Peter Stone serves on the Board of Directors of, Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

REFERENCES

[1] C. Amato, G. D. Konidaris, , A. Anders, G. Cruz, J. P. How, and L. P. Kaelbling. Policy search for multi-robot coordination under uncertainty. In *Robotics: Science and Systems Conference (RSS)*, 2015.

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

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 2002.

[4] J. Baus, A. Krüger, and W. Wahlster. A resource-adaptive mobile navigation system. In *International Conference on Intelligent User Interfaces (IUI)*, 2002.

[5] A. Butz, J. Baus, A. Krüger, and M. Lohse. A hybrid indoor navigation system. In *International Conference on Intelligent User Interfaces (IUI)*, 2001.

[6] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero. Decentralized multi-robot cooperation with auctioned pomdps. *International Journal of Robotics Research (IJRR)*, 2013.

[7] B. Coltin and M. Veloso. Online pickup and delivery planning with transfers for mobile robots. In *International Conference on Robotics and Automation (ICRA)*, 2014.

[8] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. *National Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI)*, 1999.

[9] R. A. Howard. *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*. Courier Dover Publications, 2013.

[10] J.-J. Jaw, A. R. Odoni, H. N. Psaraftis, and N. H. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 1986.

[11] B. Kartal, E. Nunes, J. Godoy, and M. Gini. Monte Carlo tree search with branch and bound for multi-robot task allocation. In *IJCAI-16 Workshop on Autonomous Mobile Service Robots*, 2016.

[12] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision (IJCV)*, 1988.

[13] P. Khandelwal, S. Barrett, and P. Stone. Leading the way: An efficient multi-robot guidance system. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015.

[14] P. Khandelwal, E. Liebman, S. Niekum, and P. Stone. On the analysis of complex backup strategies in Monte Carlo tree search. In *International Conference on Machine Learning (ICML)*, 2016.

[15] P. Khandelwal, S. Zhang, J. Sinapov, M. Leonetti, J. Thomason, F. Yang, I. Gori, M. Svetlik, P. Khante, V. Lifschitz, J. Aggarwal, R. Mooney, and P. Stone. BWIBots: A platform for bridging the gap between AI and human-robot interaction research. *International Journal of Robotics Research (IJRR)*, 2017.

[16] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, 2006.

[17] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[18] G. Lacey and K. M. Dawson-Howe. The application of robotics to a mobility aid for the elderly blind. *Robotics and Autonomous Systems*, 1998.

[19] M. Montemerlo, J. Pineau, N. Roy, et al. Experiences with a mobile robotic guide for the elderly. In *Innovative Applications of Artificial Intelligence (IAAI)*, 2002.

[20] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA-09 Workshop on Open Source Software*, 2009.

[21] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *International Conference on Robotics and Automation (ICRA)*, 1993.

[22] T. S. Standley. Finding optimal solutions to cooperative pathfinding problems. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[23] D. Stonier, J. Lee, and H. Kim. Robotics in concert. http://www.robotconcert.org/, 2015.

[24] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Cambridge University Press, 1998.

[25] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, et al. MINERVA: A second-generation museum tour-guide robot. In *International Conference on Robotics and Automation (ICRA)*, 1999.