2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC) Orlando, FL, USA, December 12-15, 2011

Mission Planning in Unstructured Environments: A Reinforcement Learning Approach

Brandon Basso, Hugh Durrant-Whyte, and J. Karl Hedrick

Abstract— **Mission planning scenarios in robotics typically involve one or more semi-autonomous agents and a human operator. The high-level goal is to find an optimal allocation of agents to tasks. Each individual task or collection of tasks may have subgoals, such as search, localization, or information gathering. Each of these lower-level goals are their own topics, and the high-level goal of task accomplishment can be categorized as a decision problem. The focus of this work is on solving decision problems through reinforcement learning. Tasks are completely parameterized by a minimal set of basis constraints—spatial, temporal, and (agent-task) coupling—which produces a single cost for each agent-task pairing. The cost completely captures the ability of a specific agent to perform a specific task. A novel state representation is presented that mitigates exponential growth of the state space by scaling independently of the spatial dimension. The decision problem is cast as an MDP and an optimal policy is found using Q-learning. Simulation results are presented for a two-agent two-task example, showing convergence of the value function and improved learning over time. To highlight the scalability of the learning algorithm, additional simulations compare the learned policy to a hand-coded greedy policy for varying number of tasks. The learned policy is shown to reliably allocate agents to tasks with minimal parameter tuning and is robust to low-level changes to agent dynamics as well as random agent motion.**

I. INTRODUCTION

A decision problem (DP) involves choosing between several possible outcomes based on an input and objective. As specific instantiation of a DP, task allocation (TA) mission planning problems involve choosing an allocation of work to work-performers so as to accomplish the mission. A popular version of mission planning, know as the vehicle routing problem, involves assigning agents to tasks in manner such that work for all agents at the end of the mission is minimized [1]. Humans routinely solve DPs in many settings, such as air traffic control. However, automated solutions to such complex spatiotemporal planning scenarios can offload tedious and laborious work from human operators, and computed solutions may in fact be more effective than human solutions.

The standard optimization approach to the TA problem involves minimizing one cost function subject to explicitly enumerated constraints on the state and actions. The solution will only be useful to the degree that the objective and constraints reflect reality. Solutions may be difficult to compute, impossible to compute in real-time, and suboptimal but usable solutions may not be available at all times. Additionally, the optimization approach does not present a clear pathway for incorporating high-level guidance, such as human feedback, other than encoding new constraints and resolving.

The goal of this work is to cast a TA-like DP as a Markov Decision Process (MDP) and solve using reinforcement learning (RL). Domain knowledge in RL is encoded as a reward signal, which propagates to a value function and policy through the learning process. The claim is that the RL framework is particularly well-suited for DPs, where complex behaviors can be learned with relatively minimal, high-level guidance provided by the reward signal and system dynamics model.

This work splits the RL approach to solving mission planning problems into three core areas. Section III addresses mission representation, which is the problem of distilling the entire state of a general DP down to a minimal representation. Section IV discusses the definition and modeling of mission planning as an MDP, as well as a specific RL algorithm for determining optimal policies. Section V addresses methods for formally incorporating human feedback directly into the learning algorithm for the purpose of encoding the complex trade-offs that humans constantly make in TA scenarios. Results from learning a policy in a multi-agent, multi-task mission planning scenario are presented in Section VI. Conclusions follow in Section VII.

II. PREVIOUS WORK

DPs in multi-agent systems can be broadly classified as either continuous (ex. consensus, motion planning) or discrete (ie. TA problems such as mission planning and vehicle routing) [2]. The focus in this work is on the TA problem.

A. Task Allocation

The problem of optimally allocating agents to tasks is generally known as an assignment problem [3]. The specific variant for routing mobile agents to tasks is known as the vehicle routing problem [1], and was first proposed in 1959 as a routing problem for trucks carrying goods between a terminal and service stations [4]. Assignment problems were originally addressed in operations research and optimization literature. Such approaches require an objective function and explicit enumeration of constraints on the state and action spaces.

B. Reinforcement Learning

While RL is a widely used technique in robotics [5] and has had particular success in motion control [6], solving TA problems with a learning scheme is not as common. The

This work was undertaken at the Australian Centre for Field Robotics (ACFR) and funded by the Office of Naval Research (ONR) at a part of the C3UV project at the University of California, Berkeley.

authors of [7] use a standard MDP model to represent a world in which an agent learns a policy to bake a cake. However, for real-world TA scenarios, the state space can be intractably large as a result of exponential state-action space scaling. As a result, much work has focused on state space reduction for multiagent planning problems modeled as MDPs [8] [9]. On the modeling front, some research has focused on augmenting MDPs to incorporate temporal effects such as stochastic action durations [10] [11]. In these cases, it is difficult to consider more than the smallest problems, as the state space must now include the time dimension.

C. Human-Robot Interaction

HRI literature relevant for TA includes topics of human trust of autonomous systems, adjustable autonomy, and the importance of explicitly accounting for the human in the loop [12]. The standard TA problem does not inherently support online human assessment or modification of plans. In the RL literature, the authors of [7] address real-time human interaction with a learning agent in their MDP example. Other work focuses on large scale problems in which robots learn a human model for the purpose of determining online when to query a human operator [13] [14]. HRI topics particularly germane to the TA setting—*How many robots are required for a particular mission? How autonomous should they be, assuming autonomy can be adjusted? When should the autonomy level change?*—are open questions and highly application specific [15].

III. Mission Representation

We would like to consider the specific branch of discrete DPs known as TA problems. In a TA mission planning scenario one or more human operators define several abstract goals (tasks) for agents to accomplish. Section III-A describes the scope of missions considered here and Section III-B addresses task representation.

A. Mission Scope

Consider the high-level mission shown in Figure 1. In this scenario, a human operator would like to gather information about an event. Several information streams are available to the operator: direct observation of the event, sensor data from a robotic agent, observations from an informant, and observations relayed through another informant. All data streams are susceptible to error. The human operator must perform the following functions to intelligently allocate agents to tasks:

- *Data Fusion:* The operator must combine several streams of low-level (eg. agent positions) and high-level (eg. priorities) data while taking into account data reliability. Data reliability can mean trust, in the case of information from an informant, or estimated sensor noise in the case of an electronic sensor.
- *Analysis:* Given the data, weighted by its information content or likelihood, the human operator must determine the state of the mission.
- *Action:* Based on the perceived state, the operator must choose an action to achieve the objective, which is information gain in the depicted scenario.

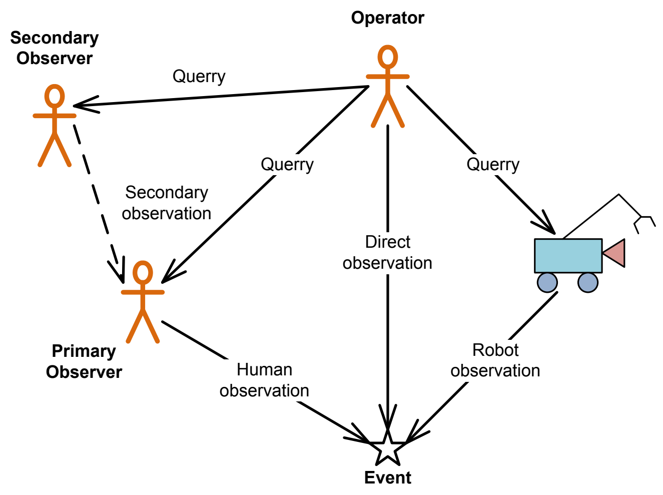


Fig. 1. High-level depiction of a mission planning scenario. One human operator must aggregate data from multiple sources of variable reliability to assess the situation and choose a course of action.

The depicted scenario could be posed as a Distributed Information Gathering (DIG) problem, where the objective is to position sensors so as to maximize the information gain about the environment [16]. However, the DIG problem requires the information content in the world to be represented as a probability density function. While a probabilistic representation of the state has advantages, particularly for data fusion, it limits the scope of missions that can be considered. For example, if the operator wants to consider higher-level objectives than information gain—visiting specific areas for alloted amounts of time, satisfying deadlines, and condition-based tasks—a broader decision problem and correspondingly broad representation must be considered.

Consider the system structure shown in Figure 2. The top loop describes the scenario shown in Figure 1. The fusion center on the left aggregates information into the abstract world state, which contains all pertinent information for the operator. The operator produces a plan, which is executed by the system and may contain both human and robotic agents. The outcome is then fed back through the same sensory pathway, closing the loop. The bottom loop in Figure 2 shows a standard observer structure [17]. The same world state passed to the operator is fed into an operator model. The operator model produces *estimated* plans and a system model produces *estimated* outcomes. These outcomes are compared to actual outcomes to generate an error signal. This abstract error signal encodes the difference between what was predicted by the model and what actually happened, which can be viewed as a measure of model mismatch. That mismatch can then be weighted (“K” block) and fed back into the operator model, closing the loop with the error signal.

The observer perspective presented here has several consequences for the decision making problem.

- The original decision problem reduces to a problem of designing the operator model and system model.
- Assuming an operator model can be realized, the resulting plan estimates are precisely the allocations we seek

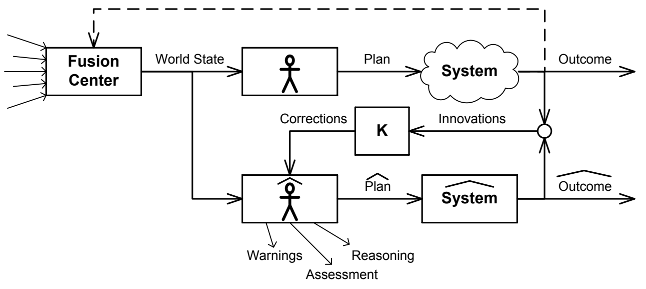


Fig. 2. Observer model for mission planning problems. The top of the model shows the "aggregate-assess-act" flow for a human operator. In parallel, the bottom half of the diagram is an observer, consisting of a human model and system model. By comparing the output of the top and the bottom, an error signal is defined, which can be incorporated back into the model in a learning process.

to compute in the TA problem.

- Assuming a system model for a TA system can be realized, predicted outcomes can be computed and used to update the operator model.
- This perspective lends itself well to a reinforcement learning solution, in which the operator model is a policy that maps world states to plans.
- In the reinforcement learning context, the policy and corresponding value function can be used to produce warnings, state assessments, and other higher-level reasoning about the relative value between different states.

B. Task Representation

A task is defined as an atomic unit of work, completable by one or several agents, which returns a cost when queried by an agent. Tasks considered here can be completely parameterized by three constraint sets: *spatial*, *temporal*, and *coupling*. In TA problems, tasks typically have a spatial element, visiting a waypoint, line segment, or area for example. Temporal constraints can include a time windows and deadlines. Coupling constraints refer to agent-task pairing specifications, such as minimum or maximum numbers of agents per task, and penalties for certain agents performing certain tasks. Other costs, such as task priority, can also be incorporated as task-specific costs.

The cost for a particular task is decomposable into two components, one associated with the cost for a specific agent to perform the task, and one associated with an intrinsic task cost, regardless of agent.

$$J_{mn}(x) = j_{mn}(x) + j_m(x) \quad where\ x \in X \tag{1}$$

Equation 1 defines the cost for agent n to perform task m the task state is completely contained in the abstract world state x. Spatial and temporal constraints, s and t, will only affect the intrinsic task cost, $j_m(x)$, and the coupling constraints, c, will only effect $j_{mn}(x)$. Therefore Equation 1 can be rewritten as Equation 2.

$$J_{mn}(s,t,c) = j_{mn}(c) + j_m(s,t) \quad where\ \{s,t,c\} \in X \tag{2}$$

The semantics of J are such that agents minimize cost: an agent n allocated to task m will drive J_{mn} to zero as spatial, temporal, and coupling constraints are satisfied.

IV. Modeling and Learning

In order to learn an optimal policy, the TA problem must be posed as a RL problem where the operator model in Figure 2 is learned. The model must be able to capture several key features:

- Agents allocating and deallocating from tasks, as well as loitering (neither allocated or deallocated)
- Interrupting and resuming partially completed tasks
- A terminal state (eg. all tasks completed) and possibly intermediate goals (eg. individual task completion)

A. Modeling

A Markov Decision Process (MDP) model is chosen here to represent the reinforcement learning problem. An MDP is defined by the tuple $< S, A, P(.,.), R(.,.,.) >$ [5]. MDPs are beneficial in this setting primarily because TA problems are inherently discrete [2] (ie. agents must be either allocated or unallocated). We adopt the reasonable assumptions that the state is fully observable, state transitions are stochastic, and we can write a reward function that captures the goals of the problem.

Defining the state S presents several challenges. The state certainly must include the condition of the agents (allocated, unallocated) and tasks, as well as task cost. Additionally, the state must capture agent and task location, both of which may vary with time. With three state variables—n agents, m tasks, and c levels of cost discretization—the state space is of dimension $c^{mn}(m+1)^n$ in the most complex scenario. It is possible to further limit the size of the state space with problem-specific assumptions.

The state S is a reduced state in that it contains features of the larger world state, X. There is a $1:1$ mapping between S and X, though many world states X can produce the same reduced state S. The state at each timestep is formally defined as follows in Table I.

TABLE I
State Definition

n	$=$	number of agents
m	$=$	number of tasks
c	$=$	number of cost levels (ie. cost grid resolution)
S	$=$	$\left\{A_1, A_2 ... A_n, J_1^1, J_2^1, ..., J_m^1, J_1^2, ..., J_m^2, ... J_m^n\right\} \Rightarrow$ global state
A	$\in$	$\{0...m\}$ where $A_i = j \Rightarrow$ agent i allocated to task j
J	$\in$	$\{0...c\}$ where $J_j^i = k \Rightarrow$ task j costs k from the perspective of agent i

Several features of this state representation should be noted.

- The state definition is highly generalized: any problem representable as agents, tasks, and costs can be captured here.

- The state space may be reduced by considering subsets; scenarios where each task can only have one allocated agent, for example.
- The representation does not explicitly depend on a spatial dimension, and therefore the size of the state space is invariant with the size of the world.

As a direct consequence of the avoiding space in the state, the proposed representation can be applied to DPs that may not have a clear spatial element, such as scheduling problems.

B. Learning

In order to learn a policy that maps states to actions, we must define define both a transition model and reward function. The specific reward function will be discussed in the Section VI, but a simple first pass is to define the goal state as all the tasks being completed and agents deallocated. Other reward functions can have intermediate goals for completing each task [18].

As a consequence of moving away from a classic "grid-world" spatial representation, a spatial transition model no longer makes sense. Even in the cost-based representation, intuitive state trajectories must still evolve: agents allocated to tasks should tend to stay allocated to tasks, reducing their cost (ie. drive a task towards completion). However, enumerating transition probabilities for even small state spaces would be laborious. Indeed, the purpose of posing the DP as a RL problem was to avoid pre-specifying explicit rules regarding agent-task allocation, but rather to have those behaviors emerge through learning.

The well-studied temporal difference algorithm, Q-learning, is employed here. It does not rely on a transition model, but rather estimates the optimal state action value function, Q^*, by forward simulation [19]. The central Q-learning equation is given below in Equation 3:

$$\begin{aligned} Q(s,a) &\leftarrow (1-\alpha)\, Q(s,a) + \alpha \left[r + \gamma \; max \left\{ Q(s^{'},a) \right\} \right] \\ \epsilon &= (\epsilon_{min} - \epsilon_{max}) \,/\, (T-1)\,(t-1) + \epsilon_{max} \\ \alpha &= \alpha_d / \,(\alpha_d + t) \\ r &= R(s', a, s) \end{aligned} \tag{3}$$

The choice of the learning rate α, exploration probability ϵ, and reward function r is, as is typical, problem specific, and largely tuned. The learning rate α is parameterized here by a positive decay time constant, α_d, and decays exponentially over the time horizon, resulting in more learning initially. The parameter ϵ determines the probability with which the optimal policy is followed. Similar in intuition to the choice of α, the learning agent explores more initially, and follows the policy more in later stages. Early exploration ensures higher state coverage and generally discourages over-learning.

V. Human Robot Interaction (HRI)

Work in HRI by the authors of [7] demonstrated that human input can simply and intuitively be incorporated into DPs. The learned policy not only reflects the optimal decisions in each state, but also incorporates an implicit human model, in that the human intent is directly encoded in the reward function. Reward functions can additionally be modified to include heuristics, and the learned policy can still be guaranteed to be optimal under certain assumptions [18]. This result from the RL community has meaningful implications for HRI. A human guided reward function could be used in mission planning scenarios to give the human operator a well-defined input path for influencing the learned policy. Section VII will discuss possible extensions through HRI techniques to learn policies that reflect the intent of human operators.

VI. Simulation Results

Several simulations have been designed to learn policies for a a variety of configurations, including varying numbers of tasks and agents. In all cases, the learned policy is specific to the particular number of tasks and agents, but general to all locations of tasks and agents. To demonstrate generality, learned policies are tested for a variety of agent-task configurations. Several parameters used during learning are listed in table II.

TABLE II
Q-LEARNING PARAMETERS

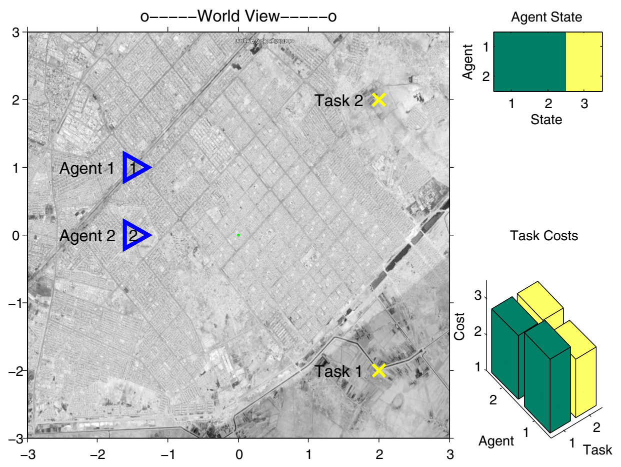
m	number of agents	2
n	number of tasks	2
c	cost grid resolution	5
$[\epsilon_{min},\ \epsilon_{max}]$	ϵ-greedy range	[0.1, 0.5]
α_d	learning rate decay constant	500
γ	discount factor	0.99
T	horizon	20,000

The reward function for the simulations gives a positive reward for reaching the goal state a slightly negative reward everywhere else so as to discourage loitering. A negative reward is also given for dual allocations—two or more agents allocated to the same task. More complex reward functions are possible, but a simple reward function is used here to illustrate learning an intuitive policy given minimal information. The performance of the learning algorithm can be discussed in terms of the Bellman error, defined in Equation 4, and the learning curve, which describes performance of the learned policy as a function of learning time horizon, T.

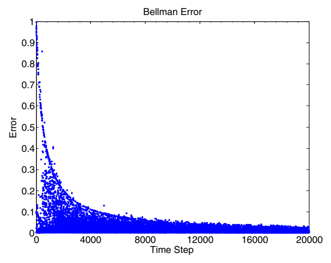
$$e = max \left\{ abs \left[Q^{'}(s,a) - Q(s,a) \right] \right\} \tag{4}$$

The results of an example learning run with parameters given in Table II are shown in Figure 3. In this example, a policy is learned that allocates two agents to two tasks, regardless of the starting positions of the tasks and agents. A purely distance-based cost function is used for simplicity. The Bellman error, shown in Figure 3(b), decays to zero, indicating that Q is approaching a stationary value. The learning curve shows that over the learning time horizon, the number of steps to episode completion decreases. In this example, 2,025 episodes were completed in the 20,000 step learning horizon and the average episode completion time decreased from 13 steps to 8 steps.

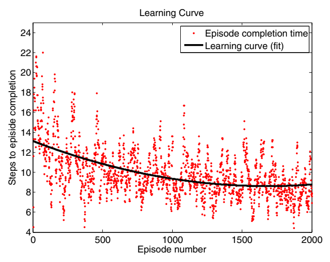
Several advantageous features of the RL approach to solving DPs are apparent in this simple simulation. Since the learned policy maps high-level agent state to high-level actions, it is robust to low level changes, such as changing individual agent dynamics. Agent speed can be varied, for example, and the same policy will guide the agents to complete the tasks. Additionally, although cost in this example is purely a function of distance, adding temporal and coupling constraints as described in Section III-B is simply a matter of encoding those constraints as a cost. For the simple reward function and distance-based cost used here, agents have learned to minimize cost, decoupled from absolute agent/task positions, and regardless of low-level agent dynamics.



(a) Simulation environment screenshot



(b) Bellman error



(c) Learning curve

Fig. 3. Two agent, two task simulation results: (a) The simulation environment shows the locations of the tasks and agents, as well as agent state (both agents are in the "unallocated" state here), and task costs. (b) The Bellman error as given in Equation 4 is shown to decay over time. (c) The learning curve shows that over the learning horizon, the average episode completion time decreases.

The next set of simulations illustrates the scaling and generalization properties of the proposed learning method to variable numbers of tasks and agents. Policies were learned for two-agent and one to four task scenarios. Due to the generality of the learning scheme, a simple scalar parameter (m, number of tasks) was changed, while all other learning parameters were kept constant (given in Table II). Larger state spaces (more tasks) did benefit from a longer learning horizon, T, although increasing T was not necessary to obtain acceptable performance. The learned policies were then simulated each 1000 times for random initial task and agent position. The resulting number of steps to episode completion are compared against the the number of steps to completion for a hand-coded greedy policy in Figure 4. The step size (essentially agent velocity) is part of the agent dynamics simulator and is a fixed parameter for both learned and greedy policies and all configurations. In the unallocated state, agents move randomly with the same step size. Both the learned policy and the greedy policy do not allow dual allocation, and the greedy policy allocates the closest agent to the closest task, breaking ties arbitrarily. Numeric results are summarized in Table III.

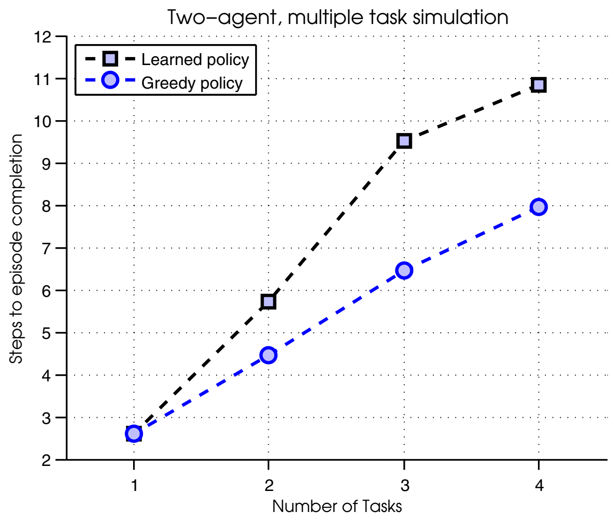


Fig. 4. Results for two-agent, variable task simulations. In each simulation all learning and simulation parameters are held constant, except for those in the table. The number of steps per episodes is averaged over 1000 random simulations for each policy.

TABLE III
MULTI-AGENT MULTI-TASK SIMULATION RESULTS

tasks	agents	# states	# actions	# learned steps	# greedy steps
1	2	36	4	2.6	2.6
2	2	729	9	5.7	4.5
3	2	11664	16	9.5	6.5
4	2	164025	25	10.8	8.0

Figure 4 shows that episode duration increases with load to the system, in the form of increased number of tasks for a constant number of agents. In the one-task scenario, the greedy policy ties the learned policy, indicating that the system learned at a minimum the greedy solution: allocate the closest agent to the closest task. In all other scenarios, the greedy policy outperforms the learned policy. It should be noted that the greedy solution is the optimum solution for one-task and two-task scenarios for two agents, and becomes less optimal as the number of tasks increases. The difference in average episode duration between greedy and learned solutions should decrease for more complex scenarios, as the myopic (greedy) course of action become less likely to be effective. Additionally, over the course of simulating 1000 episodes, several live-lock scenarios did arise for the learned solution, resulting in arbitrarily bad performance (episodes taking longer than a 3σ bound were discarded). Poor performance for the learned solution can be mitigated by longer learning time or more intelligent choice of reward function that takes into account sub-task accomplishment.

Table III shows that the state space grows exponentially with increasing number of tasks, though is still tractable up to two agents and four tasks. Future work will focus on state space size reduction through function approximation and more intelligent choice of states. Through more explicit choice of reward function, longer learning, and more concise state space definition, the learned policy may exceed the myopic greedy solution, particularly for large state spaces.

VII. Conclusion and Future Work

This work takes a reinforcement learning approach to solving decision problems such as those that arise in mission planning. A novel state representation was introduced that is both minimal in the sense that it only represents certain features of the global state, and non-spatial, in that the state space does not scale with the physical space. An MDP was chosen to model the system, and Q-learning was used to learn an optimal policy in the absence of a transition model and with a minimal reward function. The learned policy, though given very little guidance in the form of reward or system model, consistently chooses intuitive courses of action for multi-agent multi-task scenarios. Further, as a direct result of the high-level state representation, the low-level agent dynamics can be perturbed without change to the behavior.

The learned result is benchmarked against a simple greedy policy that allocates each agent to its nearest task. The two policies perform comparably, indicating that a solution close to greedy was learned with minimal guidance. Additionally, the learning scheme was general enough to generate policies for a variety of agent-task configurations without changing any of the learning parameters. Increased performance can be expected through more intelligent choice of reward function and longer learning times for larger state spaces.

Future work will focus on developing a richer MDP model and applying the learned policy to real mission scenarios. Expanding the MDP model to include time varying actions and stochastic action durations as proposed in [11] would allow for temporally constrained tasks, such as tasks with deadlines. Additionally, the current policy is dependent on a specific state space (number of agents and tasks). Future work will focus on parameterizing the state space further so that policies learned for one particular configuration would be applicable to others. Incorporating human feedback in the form of a human-guided reward signal will generate more meaningful policies for real mission planning scenarios and human operators. More realistic task-types, faster learning, and incorporating human feedback in the learning loop will enable applying the learned policies to real-world task allocation systems [20].

References

[1] D. V. P. Toth, *The Vehicle Routing Problem.* SIAM, 2009.

[2] G. M. Mathews, H. Durrant-Whyte, and M. Prokopenko, "Decentralised decision making in heterogeneous teams using anonymous optimisation," *Robot. Auton. Syst.*, vol. 57, no. 3, pp. 310–320, 2009.

[3] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[4] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, September 2005.

[6] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotic Research*, vol. 29, 2010.

[7] A. L. Thomaz and C. Breazeal, "Socially guided machine learning: Designing an algorithm to learn from real-time human interaction," in *NIPS 2005 workshop on Robot Learning in Unstructured Environments*, 2005.

[8] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored mdps," 2001.

[9] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored mdps," 2003.

[10] S. Abdallah and V. Lesser, "Modeling task allocation using a decision theoretic model," in *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems.* New York, NY, USA: ACM, 2005, pp. 719–726.

[11] J. A. Boyan and M. L. Littman, "Exact solutions to time-dependent mdps," in *in Advances in Neural Information Processing Systems.* MIT Press, 2000, pp. 1026–1032.

[12] T. Kaupp, *Human-Robot Collaboration – A Probabilistic Approach.* VDM Publishing, 2009.

[13] T. Kaupp and A. Makarenko, "Measuring human-robot team effectiveness to determine an appropriate autonomy level," in *ICRA*, 2008, pp. 2146–2151.

[14] T. Kaupp, B. Douillard, F. Ramos, A. Makarenko, and B. Upcroft, "Shared environment representation for a human-robot team performing information fusion," *J. Field Robot.*, vol. 24, no. 11-12, pp. 911–942, 2007.

[15] J. W. Crandall and M. L. Cummings, "Developing performance metrics for the supervisory control of multiple robots," in *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction.* New York, NY, USA: ACM, 2007, pp. 33–40.

[16] A. A. Makarenko, "A decentralized architecture for active sensor networks," Doctoral Thesis, The University of Sydney, 2004.

[17] T. Kailath, A. Sayed, and B. Hassibi, *Linear Estimation.* Prentice Hall, 2000, ch. 9: The Kalman Filter.

[18] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 278–287.

[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning).* The MIT Press, March 1998.

[20] B. Basso, "A component-based system architecture for mobile robotics," Masters Thesis, The University of California, Berkeley, 2009.