Apprenticeship Scheduling: Learning to Schedule from Human Experts

Matthew Gombolay
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, Massachusetts 02139
gombolay@csail.mit.edu

Reed Jensen, Jessica Stigile, & Sung-Hyun Son
Lincoln Laboratory
Massachusetts Institute of Technology
244 Wood Street
Lexington, MA 02420
{rjensen,jessica.stigile,sson}@ll.mit.edu

Julie Shah
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, Massachusetts 02139
julie_a_shah@csail.mit.edu

Abstract

Coordinating agents to complete a set of tasks with intercoupled temporal and resource constraints is computationally challenging, yet human domain experts can solve these difficult scheduling problems using paradigms learned through years of apprenticeship. A process for manually codifying this domain knowledge within a computational framework is necessary to scale beyond the one-expert, one-trainee apprenticeship model. However, a human domain expert often has difficulty describing their decision-making process, causing the codification of this knowledge to become laborious. We propose a new approach for capturing domain-expert heuristics through a pairwise ranking formulation. Our approach is model-free and does not require enumerating or iterating through a large state-space. We empirically demonstrate that this approach accurately learns multi-faceted heuristics on both a synthetic data set incorporating job-shop scheduling and vehicle routing problems and a real-world data set consisting of demonstrations of experts solving a variant of the weapon-to-target assignment problem. Our approach is able to learn scheduling policies of superior quality to those generated, on average, by human experts conducting an anti-ship missile defense task.

Introduction

Optimization and scheduling of resources is a costly, challenging problem that affects almost every aspect of our lives. In healthcare, patients with non-urgent needs who experience prolonged wait times have higher rates of treatment noncompliance and missed appointments (Kehle et al. 2011; Pizer and Prentice 2011). In military engagements, the weapon-to-target assignment problem requires warfighters to deploy minimal resources to mitigate as many threats as possible while maximizing the duration of survival (Lee, Su, and Lee 2003). The problem of optimal task allocation and sequencing with upper- and lowerbound temporal constraints (i.e., deadlines and wait constraints) is NP-Hard (Bertsimas and Weismantel 2005), and real-world scheduling problems quickly become computationally intractable. However, human domain experts are able to learn from experience to develop strategies, heuristics and rules-of-thumb to effectively respond to these problems. The challenge we pose is to autonomously learn the strategies employed by these domain experts. This knowledge can be applied and disseminated more efficiently with such a model than with a single-expert, single-apprentice model.

Researchers have realized important progress toward capturing domain-expert knowledge from demonstration (Berry et al. 2011; Abbeel and Ng 2004; Konidaris, Osentoski, and Thomas 2011; Zheng, Liu, and Ni 2015; Odom and Natarajan 2015; Vogel et al. 2012; Ziebart et al. 2008). For example, in one recent work (Berry et al. 2011) an AI scheduling assistant, called PTIME, learns how users prefer to schedule events. PTIME can then propose scheduling changes when new events occur by solving an integer program. Two limitations to this work exist: PTIME requires users to explicitly rank their preferences over scheduling options to initialize the system, and PTIME uses a complete solver, which must consider an exponential number of options in the worst case.

Research aimed at capturing domain knowledge solely based on user demonstration led to the development of Inverse Reinforcement Learning (IRL) (Abbeel and Ng 2004; Konidaris, Osentoski, and Thomas 2011; Zheng, Liu, and Ni 2015; Odom and Natarajan 2015; Vogel et al. 2012; Ziebart et al. 2008). IRL serves the dual purpose of learning an unknown reward function for a given problem and learning a policy to optimize that reward function. However, there are two primary drawbacks to IRL for scheduling problems: computational tractability and the need for an environment model.

In the classical apprenticeship learning algorithm developed by Abbeel and Ng in 2004, one must solve a Markov Decision Process (MDP) repeatedly until a convergence criteria is satisfied. However, enumerating a large state-space, such as one found in large-scale scheduling problems involving hundreds of tasks and tens of agents, can quickly become computationally intractable due to memory limitations. Approximate dynamic programming approaches exist which essentially reformulate the problem as regression (Konidaris, Osentoski, and Thomas 2011; Mnih et al. 2015), yet the amount of data required to regress over a large state space remains challenging, and MDP-based scheduling solutions exist only for simple problems (Wu et al. 2011; Wang and Usher 2005; Zhang and Dietterich 1995).

IRL also requires a model of the environment for training.

At its most basic, reinforcement learning uses a Markovian transition matrix that describes the probability of transitioning from an initial state to a subsequent state when taking a given action. For circumstances in which the environment dynamics are unknown or difficult to model within the constraints of a transition, researchers have developed Q-Learning and its variants, which have had much recent success (Mnih et al. 2015). However, these approaches require the ability to practice, or explore the state-space by querying a black-box emulator to solicit information about how taking a given action in a specific state changes that state.

Another effort has been to directly learn a function that maps states to actions (Chernova and Veloso 2007; Terrell and Mutlu 2012; Huang and Mutlu 2014). For example, Ramanujam and Balakrishnan trained a discrete-choice model using real data from air traffic controllers and showed how the model can accurately predict the correct runway configuration for an airport (Ramanujam and Balakrishnan 2011). Sammut et al. (Sammut et al. 1992) applied a decision tree model for an autopilot to learn to control an aircraft from expert demonstration. Action-driven learning techniques offer much promise for learning policies from expert demonstrators, but they have not been applied to complex scheduling problems. In order for these methods to succeed, one must model the scheduling problem in a way that allows for efficient computation of a scheduling policy.

In this paper, we propose a technique, which we call "apprenticeship scheduling," to capture this domain knowledge in the form of a scheduling policy. Our objective is to learn scheduling policies through expert demonstration and validate that schedules produced by the policies are of comparable quality to those generated by human or synthetic experts. Our approach efficiently utilizes domain-expert demonstrations without the need to train within an environment emulator. Rather than explicitly modeling a reward function and relying on dynamic programming or constraint solvers, which become computationally intractable for large-scale problems of interest, our objective is to use action-driven learning to extract the strategies of domain experts to efficiently schedule tasks.

The key to our approach is using pairwise comparisons between the actions taken (e.g., schedule agent a to complete task τ_i at time t) and the set of actions not taken (e.g., unscheduled tasks at time t) to learn relevant model parameters and scheduling policies demonstrated by the training examples. We validate our approach using both a synthetic data set of solutions for a variety of scheduling problems, and a real-world data set of demonstrations from human experts solving a variant of the weapon-to-target assignment problem.

Preliminaries

We aim to empirically demonstrate the generalizability of our learning approach through application to a variety of problem types. Korsah et al. provide a comprehensive taxonomy for classes of scheduling problems, which vary with formulation of constraints, variables, and objective or utility function (Korsah, Stentz, and Dias 2013). Within this taxonomy, there are four classes addressing interrelated utilities and constraints: No Dependencies (ND), In-Schedule Dependencies (ID), Cross-Schedule Dependencies (XD), and Complex Dependencies (CD). The ND problem class consists of independent tasks that must be assigned to agents, where the utility of one assignment does not affect the utility of other possible assignments. This class of problems is addressed, for example, in (Liu and Shell 2013). The ID class consists of problems in which the assignment of an agent to a task may affect the utility of other tasks assigned to that agent. This class is addressed, for example, in (Brunet, Choi, and How 2008) and (Nunes and Gini 2015). For XD class problems, making any assignment of an agent to a task can affect the utility of any other agent performing any other task, as addressed in (Gombolay, Wilcox, and Shah 2013). The CD class incorporates conditional constraints, where assigning an agent to a task affects the manner in which that task and other tasks can be performed. For example, in a rescue scenario, the travel routes available to fire trucks depend on the roads the bulldozers are assigned to clear (Jones, Dias, and Stentz 2011).

The Korsah et al. taxonomy also delineates between tasks that require one agent to perform, i.e. "single-agent tasks" (SA), and tasks requiring multiple agents, i.e. "multi-agent tasks" (MA). Agents that perform one task at a time are "single-task agents" (ST), and agents capable of performing multiple tasks at the same time are "multi-task agents" (MT). Lastly, the taxonomy distinguishes between instantaneous assignment (IA), in which all task and schedule commitments are made at the same time, versus time-extended assignment (TA), in which current and future commitments are planned.

In this work, we demonstrate our approach for two families of scheduling problems that span these classes. The first problem is the "Vehicle Routing Problem with Time Windows, Temporal Dependencies, and Resource Constraints (VRPTW-TDR)," which is an **XD [ST-SA-TA]** class problem. Depending on parameter selection, this family of problems encompasses the traveling salesman, job-shop scheduling, multi-vehicle routing, and multi-robot task allocation problems, among others. We consider agents to perform tasks sequentially (ST) and each task to require one agent (SA), with commitments made over time (TA). We also assume agents are heterogeneous in that they perform tasks at different rates. An agent that is incapable of performing a task is specified with a null completion rate. The objective is to minimize the makespan or other time-based performance measure. Agents and tasks have defined starting locations, and task locations are static. Each agent travels with a constant speed between task locations, and agents may only perform tasks when at the corresponding task location.

The second problem is the "Weapon-To-Target Assignment Problem (WTA)," which involves selecting weapons (e.g., a missile) to fire at targets (e.g., a stationary military compound). The canonical formulation as described in (Lee, Su, and Lee 2003) is in the ND class. However, we consider a more complex, **CD [MT-MA-TA]** class variant of the problem for anti-ship missile defense (ASMD). Here, one must determine how to deploy a set of soft kill weapons, also known as decoys, to distract an enemy's anti-ship missile from impacting one's own ship. These decoys are the agents, and the neutralization of missiles are the tasks. The effectiveness E_i^a of deploying a decoy a against target τ_i at a given location $\vec{x_a} = [x, y, \theta]$ and time t is dependent on the time history of all other decoy deployments h. Decoys can distract many missiles (MT), and many decoys can be used to distract the same missile along various points of its trajectory. Task allocation and scheduling commitments are made over time (TA). The key challenge of this problem is that the time history of how decoys have been deployed thus far affects the future effectiveness of decoys, where they should be deployed, and when they should be deployed. Agents and tasks have defined starting locations. Each task (i.e., missile) is modeled as a dynamical system with a homing function $F_\tau(h, t)$ that guides the missile towards its target and is a function of the current time and the time history h of previous decoy deployments. Decoys travel with a constant speed to their target locations x_g from the ship that deploys them.

Model for Apprenticeship Learning

In this section we present a framework for learning, via expert demontration, a scheduling policy that correctly determines which task to schedule as a function of task state.

Many approaches to learning such models are based on Markov models, such as reinforcement learning or inverse reinforcement learning (Busoniu, Babuska, and De Schutter 2008; Barto and Mahadevan 2003; Konidaris and Barto 2007; Puterman 2014). These models, however, do not capture the temporal dependencies between states and are computationally intractable for large problem sizes. To determine which tasks to schedule at which times, we draw inspiration from the domain of web page ranking (Page et al. 1999), or predicting the most relevant web page in response to a search query. One important component of page ranking is capturing how pages relate to one another as a graph with nodes (i.e., web pages) and directed arcs (i.e., links between those pages) (Page et al. 1999). This connectivity is a suitable analogy for the complex temporal dependencies (i.e., precedence, wait, and deadline constraints) relating tasks in a scheduling problem.

Recent approaches to page ranking have focused on pairwise and listwise models, which have been shown to have advantages over pointwise models (Valizadegan et al. 2009). In listwise ranking, the goal is to generate a ranked list of web pages directly (Cao et al. 2007; Valizadegan et al. 2009; Volkovs and Zemel 2009), while a pairwise approach determines ranking based on pairwise comparisons between individual pages (Jin, Valizadegan, and Li 2008; Pahikkala et al. 2007). We chose the pairwise formulation to model the problem of predicting the best task to schedule at time t.

The pairwise model has key advantages over the listwise approach. First, classification algorithms (e.g., support vector machines) can be directly applied (Cao et al. 2007). Second, a pairwise approach is non-parametric, in that the cardinality of the input vector is not dependent upon the number of tasks (or actions) that can be performed in any instance. Third, training examples of pairwise comparisons in the data can be readily solicited. From a given observation in which a task was scheduled, we only know which task was most important – not the relative importance between all tasks. Thus, we create training examples based on pairwise comparisons between the scheduled and unscheduled tasks. A pairwise approach is most natural because we lack a context to determine the relative rank between two unscheduled tasks.

Consider a set of tasks, $\tau_i \in \tau$, each of which has a set of real-valued features, γ_{τ_i}. Each scheduling-relevant feature $\gamma_{\tau_i}^j$ may represent, for example, the deadline, the earliest time the task is available, the duration of the task, which resource r is required by this task, etc. Next, consider a set of m observations, $O = \{O_1, O_2, \ldots, O_m\}$. Observation O_m consists of a feature vector $\{\gamma_{\tau_1}, \gamma_{\tau_2}, \ldots, \gamma_{\tau_n}\}$ describing the state of each task, the task scheduled by the expert demonstrator (including a null task, $\tau_\emptyset$, if no task was scheduled) and the time at which an action was taken. The goal is to then learn a policy that correctly determines which task to schedule as a function of task state.

We deconstruct the problem into two steps: Step 1): For each agent/resource pair, determine the candidate next task to schedule. Step 2): For each task, determine whether to schedule the task from the current state. In order to learn to correctly assign the next task to schedule, we transform each observation O_m into a new set of observations by performing pairwise comparisons between the scheduled task τ_i and the set of tasks that were not scheduled (Equations 1-2). Equation 1 creates a positive example for each observation in which a task τ_i was scheduled. This example consists of the input feature vector, $\phi^m_{\langle\tau_i,\tau_x\rangle}$, and a positive label, $y^m_{\langle\tau_i,\tau_x\rangle} = 1$. Each element of the input feature vector $\phi^m_{\langle\tau_i,\tau_x\rangle}$ is computed as the difference between the corresponding values in the feature vectors γ_{τ_i} and γ_{τ_x}, describing scheduled tasks τ_i and unscheduled task τ_x. Equation 2 creates a set of negative examples with $y^m_{\langle\tau_x,\tau_i\rangle} = 0$. For the input vector, we take the difference of the feature values between unscheduled task τ_x and scheduled task τ_i.

This feature set is then augmented to capture additional contextual information important for scheduling, which may not be captured in examples consisting solely of differences between features of tasks. For example, one's scheduling policy may change based on the progress towards completion of the tasks, i.e. based on proportion of tasks completed so far. To provide this high-level information, we include ξ_τ, the set of contextual, high-level features describing the set of tasks for observation O_m, in (Equations 1-2). Prior work has shown that domain experts are adept at describing the features (both high-level, contextual and task-specific) used in their decision-making, yet, it is more difficult for experts to describe how they reason about these features (Cheng, Wei, and Tseng 2006; Raghavan, Madani, and Jones 2006).

We can use these observations to train a classifier $f_{priority}(\tau_i, \tau_x) \in \{0, 1\}$ to predict whether it is better to schedule task τ_i as the next task rather than τ_x. Given this pairwise classifier, we can determine which single task τ_i is the highest priority task τ_i^* according to Equation 3 by determining which task is most often higher priority in comparison to the other tasks in $\boldsymbol{\tau}$.

Next, we must learn to predict whether τ_i^* should be scheduled or the agent should remain idle. We train a second classifier, $f_{act}(\tau_i) \in \{0, 1\}$, which predicts whether or not τ_i should be scheduled. In our observations set, O, we only have examples in which a task was scheduled and those in which no task was scheduled. To train this classifier, we construct a new set of examples according to Equation 4 where positive labels are assigned to examples from O_m in which a task was scheduled and negative labels to examples in O_m in which no task was scheduled.

Finally, we construct a scheduling algorithm to act as an apprentice scheduler (Figure 1). Lines 1- 2 iterate over each agent at each time step. In Line 3, the highest priority task τ_i^* is determined for a particular agent. In Lines 4- 5, τ_i^* is scheduled *iff* $f_{act}(\tau_i^*)$ predicts that τ_i^* should be scheduled at the current time.

A benefit of the pairwise ranking formulation is that one can apply any one of a number of standard machine learning classification techniques to learn $f_{priority}(\tau_i, \tau_x)$ and $f_{act}(\tau_i)$. In our experimental evaluation, we compare the performance of a decision tree, support vector machine and other common classification techniques.

Algorithm 1 Pseudocode for an Apprentice Scheduler

```
ApprenticeScheduler($\boldsymbol{\tau}$,$\boldsymbol{A}$,$\boldsymbol{TC}$,$\boldsymbol{\tau_R}$)
1: for $t = 0$ to $T$ do
2:     for all agents $a \in \boldsymbol{A}$ do
3:         $\tau_i^* \leftarrow \underset{\tau_i \in \boldsymbol{\tau}}{\text{argmax}} \sum_{\tau_x \in \boldsymbol{\tau}} f_{priority}(\tau_i, \tau_x)$
4:         if $f_{act}(\tau_i^*) == 1$ then
5:             Schedule $\tau_i^*$
6:         end if
7:     end for
8: end for
```

$$\begin{aligned} {}^{rank}\theta^m_{\langle \tau_i, \tau_x \rangle} &:= \left[\xi_{\boldsymbol{\tau}}, \gamma_{\tau_i} - \gamma_{\tau_x} \right], \\ y^m_{\langle \tau_i, \tau_x \rangle} &= 1, \\ \forall \tau_x \in \boldsymbol{\tau} \backslash \tau_i, \forall O_m &\in \boldsymbol{O} | \tau_i \text{ scheduled in } O_m \end{aligned} \tag{1}$$

$$\begin{aligned} {}^{rank}\theta^m_{\langle \tau_x, \tau_i \rangle} &:= \left[\xi_{\boldsymbol{\tau}}, \gamma_{\tau_x} - \gamma_{\tau_i} \right], \\ y^m_{\langle \tau_x, \tau_i \rangle} &= 0, \\ \forall \tau_x \in \boldsymbol{\tau} \backslash \tau_i, \forall O_m &\in \boldsymbol{O} | \tau_i \text{ scheduled in } O_m \end{aligned} \tag{2}$$

$$\widehat{\tau_i^*} = \underset{\tau_i \in \boldsymbol{\tau}}{\text{argmax}} \sum_{\tau_x \in \boldsymbol{\tau}} f_{priority}\left(\tau_i, \tau_x\right) \tag{3}$$

$$\begin{aligned} {}^{act}\phi^m_{\tau_i} &:= \left[\xi_{\boldsymbol{\tau}}, \gamma_{\tau_i} \right], \\ y^m_{\tau_i} &= \begin{cases} 1 : \tau_i \text{ scheduled in } O_m \wedge \\ \qquad \tau_i \text{ scheduled in } O_{m+1} \\ 0 : \tau_\emptyset \text{ scheduled in } O_m \end{cases} \end{aligned} \tag{4}$$

Data Sets

Next, we validate that schedules produced by our learned policies are of comparable quality to those generated by human or synthetic experts.

Synthetic Data Set

First we generated a synthetic dataset in which schedules were produced through application of context-dependent scheduling heuristics. Our objective was to show that our technique learns both the heuristics and policy for their correct application. We constructed a synthetic data set based on the Vehicle Routing Problem with Time Windows, Temporal Dependencies, and Resources. Problems involved two heterogeneous agents and 20 partially ordered tasks located within a 20 x 20 grid.

We constructed a mock heuristic to serve as our source for synthetic-expert demonstrations, as shown in Figure 2. Our heuristics were based on our prior work in scheduling (Tan et al. 2001; Gombolay, Wilcox, and Shah 2013) and prior work addressing the vehicle routing problem with time windows (Solomon 1987). In Lines 1-6, the algorithm collects all alive and enabled tasks $\tau_i \in \boldsymbol{AE}$ as defined by (Muscettola, Morris, and Tsamardinos 1998). Consider a pair of tasks τ_i and τ_j, with start and finish times s_i, f_i and s_j, f_j, respectively, such that there is a wait constraint requiring τ_i to start at least $W_{\langle \tau_j, \tau_i \rangle}$ units of time after τ_j. A task τ_i is alive and enabled if $t \geq f_j + W_{\tau_j, \tau_i}$ for all such τ_j and $W_{\langle \tau_j, \tau_i \rangle}$ in $\boldsymbol{\tau}$.

Algorithm 2 Pseudocode for the Mock Heuristic

```
MockHeuristic($\boldsymbol{\tau}$,$\boldsymbol{A}$,$\boldsymbol{TC}$,$\boldsymbol{\tau_R}$)
1: $\boldsymbol{\tau_{AE}} \leftarrow$ initialize alive and enabled task set
2: for all $\tau_i \in \boldsymbol{\tau}$ do
3:     if all wait constraints for $\tau_i$ have been satisfied then
4:         $\boldsymbol{\tau_{AE}} \leftarrow \boldsymbol{\tau_{AE}} \cup \tau_i$
5:     end if
6: end for
7: for all agents $a \in \boldsymbol{A}$ do
8:     if Speed $\leq 1\frac{m}{s}$ //Vehicle Routing Problem then
9:         $\vec{l}_x \leftarrow$ location of $\tau_x$
10:        $\vec{l}_a \leftarrow$ location of agent $a$
11:        $\theta_{xa} \leftarrow \frac{acos\left(\vec{l}_x^T \vec{l}_a\right)}{\|\vec{l}_x\| \|\vec{l}_a\|}$
12:        $\tau_i^* \leftarrow \underset{\tau_x \in \boldsymbol{\tau_{AE}}}{\text{argmin}} \Big( \|\vec{l}_x - \vec{l}_a\|$
13:                $+ \alpha_1 \theta_{xa} + \alpha_2 \|\vec{l}_x - \vec{l}_a\| \theta_{xa} \Big)$
14:    else if $\sum_{\tau_i} \sum_{\tau_x} 1_{R_{\tau_i} = R_{\tau_x}} \geq$ c //Resource Contention Mode then
15:        $\tau_i^* \leftarrow \underset{\tau_x \in \boldsymbol{\tau_{AE}}}{\text{argmax}} \left( \left( \sum_{\tau_i} \sum_{\tau_x} 1_{R_{\tau_i} = R_{\tau_x}} \right) - \alpha_3 d_{\tau_x} \right)$
16:    else
17:        $\tau_i^* \leftarrow \underset{\tau_x \in \boldsymbol{\tau_{AE}}}{\text{argmin}}\, d_{\tau_x}$
18:    end if
19:    if $R_{\tau_i^*}$ is unoccupied at time $t$ then
20:        if agent $a$ could travel to reach $\tau_i^*$ by time $t$ then
21:            Schedule $\tau_i^*$
22:        end if
23:    end if
24: end for
```

Next, the heuristic iterates over each agent and task to find the highest-priority task τ_i^* to schedule for each agent.

In Lines 7-18, the algorithm determines which heuristic is most appropriate to apply. If agent speed is sufficiently slow, travel time will become the major bottleneck. If the agents are fast but one or more resources are heavily utilized, use of these resources can become the bottleneck. Otherwise, task durations and associated wait constraints are generally most important.

In Line 8, the algorithm identifies travel distance as the most important bottleneck and switches to a heuristic well-suited for vehicle routing that minimizes a weighted, linear combination of features (Gambardella, Éric Taillard, and Agazzi 1999; Solomon 1987) comprised of the distance and angle relative to the origin between agent a and τ_x and an indicator term for whether τ_x must be executed to satisfy a wait constraint for another task τ_a. This rule is based on prior work on the vehicle routing problem (Gambardella, Éric Taillard, and Agazzi 1999; Solomon 1987) and on a heuristic proposed to mitigate resource-contention in multi-robot, multi-resource problems (Gombolay, Wilcox, and Shah 2013). In Line 14, the algorithm determines that there may be a resource bottleneck and tries to alleviate it by switching to a resource-contention mode and applying a heuristic that returns the task $\tau_i^* \in \boldsymbol{\tau}_{AE}$ that maximizes a weighted, linear combination of the commonality of the task's required resource less its deadline. If neither travel distance nor resource contention are perceived as the major bottlenecks, the algorithm switches to applying an Earliest Deadline First rule (Line 16), which performs well across many scheduling domains (Chen et al. 2014; Gombolay et al. 2013). If the resource required for τ_i^*, $R_{\tau_i^*}$, is idle (Line 19), and the agent is able to reach the task by time t (Line 20), then the heuristic schedules task τ_i^* at time t (Line 21). We note that an agent is able to reach task τ_i^* if $t \geq f_j + k\left(x_i - x_j\right) / \|x_i - x_j\|$ for all $\tau_j \in \boldsymbol{\tau}$ that the agent has already completed, where k is the agent's speed.

With the heuristic shown in Algorithm 2, we generated a set of training data incorporating $30,000$ task sets – $10,000$ for each type of bottleneck identified by the heuristic. A spectrum of problems (i.e. traveling salesman, job-shop scheduling, multi-vehicle routing) was represented, as task locations, agent travel speeds and task completion rates were varied. These task sets provided $533,737$ observations. In 96% of the observations, the mock heuristic idled (i.e., chose not to schedule a task), and in 4% of the observations, the mock heuristic scheduled an agent to complete a task.

Real-World Data Set

We collected a real-world data set consisting of human demonstrators of various skill levels solving the ASMD weapon-to-target assignment problem. We utilized a virtual gaming environment requiring players to manage a set of heterogeneous decoys to defeat raids of heterogeneous enemy anti-ship missiles. We modeled a scenario with five types of decoys and ten types of threats. The threats were randomly generated for each played scenario, thereby promoting the development of strategies that were robust to a distribution of threat scenarios. Each decoy had a specified effectiveness against each threat type. Players attempted to deploy a set of decoys of the correct decoy types, at the right location, and at the right times in order to distract incoming missiles. Threats were launched over time, meaning an effective deployment at time t could become counterproductive at a future point in time as new enemy missiles were launched. Games were scored as follows: 10,000 points were received each time a threat was neutralized and 2 points for each second each threat spent homing in on a decoy. 5,000 points were subtracted for each threat impact and 1 points for each second each threat spent homing in on one's own ship. Lastly, 25-1,000 points were subtracted for each deploy of a decoy, depending on the type.

The collected data set consisted of 311 games played from 35 human players across 45 threat configurations or "scenarios". We sub-selected sixteen threat configurations such that each configuration had at least one human demonstration that mitigated all enemy missiles. For these sixteen threat configurations, there were 162 total games played by 27 unique human demonstrators. Players consisted of technical fellows and associates as well as contractors at MIT Lincoln Laboratory, and their expertise varied from "generally knowledgeable about the ASMD problem" to "domain experts" with professional experience or training in anti-ship missile defense.

Empirical Evaluation

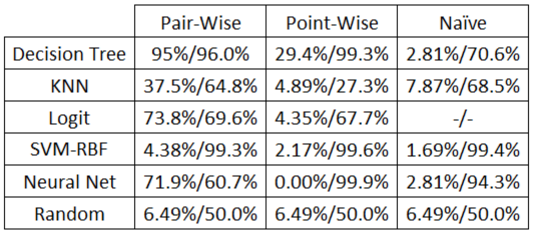
In this section, we evaluate our prototype for apprenticeship scheduling on the synthetic and real-world data sets.

Synthetic Data Set

We trained our model using a decision tree, KNN classifier, logistic regression (logit) model, support vector machine with a radial basis function kernel (SVM-RBF) and a neural network to learn $f_{priority}(.,.)$ and $f_{act}(.)$. We randomly sampled 85% of the data for training and 15% for testing. We defined the features as follows: The high-level feature-vector of the task set, ξ_τ, is comprised of the agents' speed and the degree of resource contention $\sum_{\tau_i}\sum_{\tau_x} 1_{R_{\tau_i}=R_{\tau_x}}$. The task-specific feature vector γ_{τ_i} is comprised of the task's deadline, a binary indicator for whether or not the task's precedence constraints have been satisfied, the number of tasks sharing this task's resource, a binary indicator for whether or not this task's resource is available, the travel time remaining to reach the task, the distance agent a would travel to reach τ_i and the angular difference between the vector describing the location of agent a and the vector describing the position of τ_i relative to agent a.

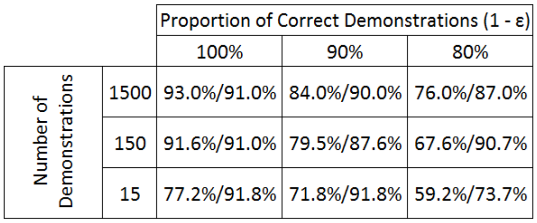
We compared the performance of our pairwise approach with a point-wise approach and a naïve approach. In the point-wise approach, training examples for selecting the highest priority task were of the form ${}^{rank}\phi_{\tau_i}^m := [\xi_{\boldsymbol{\tau}}, \gamma_{\tau_i}]$; the label $\gamma_{\tau_i}^m$ was equal to 1 if task τ_i was scheduled in observation m, and was 0 otherwise. In the naïve approach, examples were comprised of an input vector that concatenates the high-level features of task set and the task-specific features of the form ${}^{rank}\phi^m := [\xi_{\boldsymbol{\tau}}, \gamma_{\tau_1}, \gamma_{\tau_2}, \ldots, \gamma_{\tau_n}]$; labels y^m are equal to the index of the task τ_i scheduled in observation m.

Table 1 depicts the sensitivity (i.e., true positive rate) and specificity (i.e., true negative rate) of the model. We found



	Pair-Wise	Point-Wise	Naïve
Decision Tree	95%/96.0%	29.4%/99.3%	2.81%/70.6%
KNN	37.5%/64.8%	4.89%/27.3%	7.87%/68.5%
Logit	73.8%/69.6%	4.35%/67.7%	-/-
SVM-RBF	4.38%/99.3%	2.17%/99.6%	1.69%/99.4%
Neural Net	71.9%/60.7%	0.00%/99.9%	2.81%/94.3%
Random	6.49%/50.0%	6.49%/50.0%	6.49%/50.0%

Table 1: Sensitivity/specificity for machine learning techniques using the pair-wise, point-wise, and naïve approaches.



		Proportion of Correct Demonstrations (1 - ε)		
		100%	90%	80%
Number of Demonstrations	1500	93.0%/91.0%	84.0%/90.0%	76.0%/87.0%
	150	91.6%/91.0%	79.5%/87.6%	67.6%/90.7%
	15	77.2%/91.8%	71.8%/91.8%	59.2%/73.7%

Table 2: Sensitivity/specificity for a pair-wise decision tree varying the number and proportion of correct demonstrations.

that a pairwise model outperformed the point-wise and naïve approaches. Within the pairwise model, a decision tree provided the best performance: The trained decision tree was able to identify the correct task and when to schedule that task 95% of the time, and was able to accurately predict when no task should be scheduled 96% of the time.

We sought to more fully understand the performance of a decision tree trained with a pairwise model as a function of the number and quality of training examples, as shown in Table 2. We trained decision trees with our pairwise model with 15, 150, and 1,500 demonstrations. The sensitivity and specificity reported in Table 2 for 15 and 150 demonstrations are the average sensitivity and specificity of ten models trained via random sub-sampling without replacement. We also varied the quality of the training examples, assuming the demonstrator was operating under an ϵ-greedy approach with a probability of $(1 - \epsilon)$ of selecting the correct task to schedule and selecting another task from a uniform distribution otherwise. This assumption is conservative; a demonstrator making an error would be more likely to pick the second- or third-best task than selecting a task at random.

Training a model based on pairwise comparison between the scheduled task and unscheduled tasks effectively produced policies of comparable quality to those generated by the synthetic expert. The decision tree model performed well due to modal nature of the multi-faceted scheduling heuristic. We note that this dataset was composed of scheduling strategies with mixed discrete-continuous functional components, and in future work, performance can be further improved by combining decision trees with logistic regression. This hybrid learning approach has seen success in machine learning classification tasks (Landwehr, Hall, and Frank 2005), and can be readily applied to this apprenticeship scheduling framework.

Real-World Data Set

We trained and tested a decision tree on our pairwise scheduling model via leave-one-out cross-validation using the sixteen real demonstrations in which a player mitigated all enemy missiles. Each demonstration came from a unique threat scenario. Features for each decoy/missile pair (or null decoy deployment from inaction) included indicators for whether the decoy had been placed such that the missile was successfully distracted by that decoy, whether the missile would be lured into hitting the ship by the decoy placement, or whether the missile would be unaffected by the placement. Across all sixteen scenarios, the average and standard deviation of players' scores was 74,728 ± 26,824. With merely 15 examples of expert human demonstrations, our apprenticeship scheduling model was able to achieve an average score of 87,540 with standard deviation of 16,842.

We performed statistical analysis to evaluate our hypothesis that the scores produced by the learned policy were statistically significantly better than the average scores achieved by the human demonstrators. The null hypothesis stated that the number of scenarios in which the apprenticeship scheduling model achieved superior performance was less than or equal to the number of scenarios in which the average score of the human demonstrators was superior to the apprenticeship scheduler. We set the significance level at $\alpha = 0.05$. Application of a binomial test[1] rejects the null hypothesis, meaning that the learned scheduling policy performed better than the human demonstrators on statistically significantly more scenarios (12 versus 4 scenarios), with $p = 0.011$. This promising result was achieved with a relatively small training set and indicates that the learned policy can form the basis for a training tool to improve the average player's score.

Conclusions

We propose a technique for apprenticeship scheduling that relies on a pairwise comparison of scheduled and unscheduled tasks to learn a model for task prioritization. We validate our apprenticeship scheduling algorithm on both a synthetic data set covering a variety of scheduling problems with lower- and upperbound temporal constraints, resource constraints and travel distance considerations, as well as a real-world data set where human demonstrators solved a variant of the weapon-to-target assignment problem. Our approach is able to learn scheduling policies of superior quality to those generated, on average, by human experts conducting an anti-ship missile defense task.

[1] The probability of rejecting the null hypothesis is $1 - \sum_{k=0}^{a} \binom{a}{a+b} \rho^a (1 - \rho)^b$ with $\rho = 1/2$, and where a and b are the number of scenarios the apprenticeship scheduling model performed better than the average human score. and vice versa, respectively.

Acknowledgements

This work was supported by the National Science Foundation Graduate Research Fellowship Program under grant number 2388357.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*.

Barto, A. G., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems* 13(1-2):41–77.

Berry, P. M.; Gervasio, M.; Peintner, B.; and Yorke-Smith, N. 2011. Ptime: Personalized assistance for calendaring. *ACM Trans. Intell. Syst. Technol.* 2(4):40:1–40:22.

Brunet, L.; Choi, H.-L.; and How, J. P. 2008. Consensus-based auction approaches for decentralized task assignment. In *Proc. AIAA GNC*.

Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. SMC Part C* 38(2):156–172.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proc. ICML*, 129–136. ACM.

Cheng, T.-H.; Wei, C.-P.; and Tseng, V. S. 2006. Feature selection for medical data mining: Comparisons of expert judgment and automatic approaches. In *Proc. CBMS*, 165–170.

Chernova, S., and Veloso, M. 2007. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proc. AAMAS*, 233:1–233:8. ACM.

Gambardella, L. M.; Éric Taillard; and Agazzi, G. 1999. MACS-VRPTW: A multiple colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*, 63–76. McGraw-Hill.

Gombolay, M.; Wilcox, R.; and Shah, J. 2013. Fast scheduling of multi-robot teams with temporospatial constrints. In *Proc. RSS*.

Huang, C.-M., and Mutlu, B. 2014. Learning-based modeling of multimodal behaviors for humanlike robots. In *Proc. HRI*, 57–64.

Jin, R.; Valizadegan, H.; and Li, H. 2008. Ranking refinement and its application to information retrieval. In *Proc. Conference on WWW*, 397–406. ACM.

Jones, E.; Dias, M.; and Stentz, A. 2011. Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous Robots* 30(1):41–56.

Kehle, S. M.; Greer, N.; Rutks, I.; and Wilt, T. 2011. Interventions to improve veterans? access to care: A systematic review of the literature. *Journal of General Internal Medicine* 26(2):689–696.

Konidaris, G., and Barto, A. 2007. Building portable options: Skill transfer in reinforcement learning. In *Proc. IJCAI*, 895–900.

Konidaris, G.; Osentoski, S.; and Thomas, P. 2011. Value function approximation in reinforcement learning using the fourier basis. In *Proc. AAAI*, 380–385.

Korsah, G. A.; Stentz, A.; and Dias, M. B. 2013. A comprehensive taxonomy for multi-robot task allocation. *IJRR* 32(12):1495–1512.

Landwehr, N.; Hall, M.; and Frank, E. 2005. Logistic model trees. *Machine Learning* 59(1-2):161–205.

Lee, Z.-J.; Su, S.-F.; and Lee, C.-Y. 2003. Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *IEEE Trans. SMC Part B* 33(1):113–121.

Liu, L., and Shell, D. A. 2013. Optiml market-based multi-robot task allocation via strategic pricing. In *Proc. RSS*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proc. KR&R*.

Nunes, E., and Gini, M. 2015. Multi-robot auctions for allocation of tasks with temporal constraints. In *Proc. AAAI*, 2110–2116.

Odom, P., and Natarajan, S. 2015. Active advice seeking for inverse reinforcement learning. In *Proc. AAAI*, 4186–4187.

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. SIDL-WP-1999-0120.

Pahikkala, T.; Tsivtsivadze, E.; Airola, A.; Boberg, J.; and Salakoski, T. 2007. Learning to rank with pairwise regularized least-squares. In *SIGIR Workshop on Learning to Rank for Information Retrieval*, 27–33.

Pizer, S. D., and Prentice, J. C. 2011. What are the consequences of waiting for health care in the veteran population? *Journal of General Internal Medicine* 26(2):676–682.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Raghavan, H.; Madani, O.; and Jones, R. 2006. Active learning with feedback on features and instances. *Journal of Machine Learning Research* 7:1655–1686.

Ramanujam, V., and Balakrishnan, H. 2011. Estimation of maximum-likelihood discrete-choice models of the runway configuration selection process. In *Proc. ACC*, 2160–2167.

Sammut, C.; Hurst, S.; Kedzier, D.; and Michie, D. 1992. Learning to fly. In *Proc. ICML*, 385–393.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35(2):254–265.

Tan, K.; Lee, L.; Zhu, Q.; and Ou, K. 2001. Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering* 15(3):281 – 295.

Terrell, A., and Mutlu, B. 2012. A regression-based approach to modeling addressee backchannels. In *Proc. Special Interest Group on Discourse and Dialogue*, 280–289.

Valizadegan, H.; Jin, R.; Zhang, R.; and Mao, J. 2009. Learning to rank by optimizing NDCG measure. In *NIPS*, 1883–1891.

Vogel, A.; Ramach, D.; Gupta, R.; and Raux, A. 2012. Improving hybrid vehicle fuel efficiency using inverse reinforcement learning. In *Proc. AAAI*, 384–390.

Volkovs, M. N., and Zemel, R. S. 2009. Boltzrank: Learning to maximize expected ranking gain. In *Proc. ICML*, 1089–1096.

Wang, Y.-C., and Usher, J. M. 2005. Application of reinforcement learning for agent-based production scheduling. *Eng. Appl. Artif. Intell.* 18(1):73–82.

Wu, J.; Xu, X.; Zhang, P.; and Liu, C. 2011. A novel multi-agent reinforcement learning approach for job scheduling in grid computing. *Future Generation Computer Systems* 27(5):430 – 439.

Zhang, W., and Dietterich, T. G. 1995. A reinforcement learning approach to job-shop scheduling. In *Proc. IJCAI*, 1114–1120. Morgan Kaufmann.

Zheng, J.; Liu, S.; and Ni, L. 2015. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *Proc. AAAI*, 2198–2205.

Ziebart, B. D.; Maas, A.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, 1433–1438.