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
MIT Lincoln Laboratory
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 "single-expert, single-trainee" apprenticeship model. However, human domain experts often have difficulty describing their decision-making processes, 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 multifaceted 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 weapon-to-target assignment problem.

Introduction

Resource scheduling and optimization 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 non-compliance 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 in order to mitigate as many threats as possible while maximizing the duration of survival [Lee *et al.*, 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 made significant progress toward capturing domain-expert knowledge from demonstration [Berry *et al.*, 2011; Abbeel and Ng, 2004; Konidaris *et al.*, 2011; Zheng *et al.*, 2015; Odom and Natarajan, 2015; Vogel *et al.*, 2012; Ziebart *et al.*, 2008]. In one recent work [Berry *et al.*, 2011], an AI scheduling assistant, called PTIME, learned how users prefer to schedule events. PTIME was subsequently able to propose scheduling changes when new events occurred by solving an integer program. Two limitations to this work exist: PTIME requires users to explicitly rank their preferences about scheduling options to initialize the system, and also uses a complete solver that, in the worst-case scenario, must consider an exponential number of options.

Research focused on capturing domain knowledge based solely on user demonstration has led to the development of inverse reinforcement learning (IRL) [Abbeel and Ng, 2004; Konidaris *et al.*, 2011; Zheng *et al.*, 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.

The classical apprenticeship learning algorithm developed by Abbeel and Ng in 2004 requires repeated solving of a Markov decision process (MDP) until a convergence criterion is satisfied. However, enumerating a large state-space, such as that 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 that essentially reformulate the problem as regression [Konidaris *et al.*, 2011; Mnih *et al.*, 2015], but 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. In order to address circumstances in which environmental 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 will change that state.

Another method has been directly learning 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 this 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 great 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 these 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 upon 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 in order to efficiently schedule tasks.

The key to our approach is the use of 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 the 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 [Lee *et al.*, 2003]. The synthetic and real-world problem domains we use to empirically validate our approach represent two of the most challenging classes within the taxonomy established by Korsah et al. [Korsah *et al.*, 2013].

Problem Domain

We aimed to empirically demonstrate the generalizability of our learning approach through application to a variety of problem types. Korsah et al. provided a comprehensive taxonomy for classes of scheduling problems, which vary according to formulation of constraints, variables and objective or utility function [Korsah *et al.*, 2013]. Within this taxonomy, there are four classes addressing interrelated utilities and constraints: No Dependencies (ND (e.g., [Liu and Shell, 2013])), In-Schedule Dependencies (ID) (e.g., [Brunet *et al.*, 2008; Gombolay and Shah, 2014; Nunes and Gini, 2015]), Cross-Schedule Dependencies (XD) (e.g., [Gombolay *et al.*, 2013]) and Complex Dependencies (CD) (e.g.,[Jones *et al.*, 2011]).

The Korsah et al. taxonomy also delineates between tasks requiring one agent, i.e. "single-agent tasks" (SA); and tasks requiring multiple agents, i.e. "multi-agent tasks" (MA). Similarly, 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, and "time-extended assignment" (TA), in which current and future commitments are planned.

In this work, we demonstrate our approach for two of the most difficult classes of scheduling problems defined within this taxonomy: **XD [ST-SA-TA]** and **CD [MT-MA-TA]**. The first problem we consider 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 upon parameter selection, this family of problems encompasses the traveling salesman, job-shop scheduling, multi-vehicle routing and multi-robot task allocation problems, among others. The second problem is a more complex variant of the weapon-to-target assignment problem (WTA) [Lee *et al.*, 2003], which falls under the **CD [MT-MA-TA]** class.

Model for Apprenticeship Learning

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

Many approaches to learning via demonstration, such as reinforcement or inverse reinforcement learning, are based on Markov models [Busoniu *et al.*, 2008; Barto and Mahadevan, 2003; Konidaris and Barto, 2007; Puterman, 2014]. Markov models, however, do not capture the temporal dependencies between states and are computationally intractable for large problem sizes. In order 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 (representing web pages) and directed arcs (representing links between those pages) [Page *et al.*, 1999]. This connectivity is a suitable analogy for the complex temporal dependencies (precedence, wait and deadline constraints) relating tasks within a scheduling problem.

Recent approaches to page ranking have focused on pairwise and listwise models, which each 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 *et al.*, 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 during 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 scheduled and unscheduled tasks. A pairwise approach is more natural because we lack the necessary context to determine the relative rank between two unscheduled tasks.

Consider a set of tasks, $\tau_i \in \tau$, in which each task 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 learn a policy that correctly determines which task to schedule as a function of the task state.

We deconstruct the problem into two steps: 1): For each agent/resource pair, determine the candidate next task to schedule; and 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 unscheduled tasks (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 task 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 task features. For example, a scheduling policy may change based on progress toward task completion; i.e., the 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).

Our technique relies upon the ability of domain experts to articulate an appropriate set of features for the problem. We believe this to be a reasonable limitation. Results from prior work have indicated that domain experts are adept at describing the (high-level, contextual and task-specific) features used in their decision-making; however, it is more difficult for experts to describe how they reason about these features [Cheng *et al.*, 2006; Raghavan *et al.*, 2006]. In future work, we aim to extend our approach to include feature learning rather than relying upon experts to enumerate the important features they reason about in order to construct schedules.

Given these observations O_m and their associated features, we can 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. With 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 has the highest cumulative priority in comparison to the other tasks in $\boldsymbol{\tau}$.

In this work, we train a single classifier $f_{priority}(\tau_i, \tau_x)$ to model the behavior of the set of all agents rather than train one $f_{priority}(\tau_i, \tau_x)$ for each agent. $f_{priority}(\tau_i, \tau_x)$ is a function of all features associated with the agents; as such, agents need not be interchangeable, and different sets of features may be associated with each agent.

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

Finally, we construct a scheduling algorithm to act as an apprentice scheduler (Algorithm 1). This algorithm takes as input the set of tasks $\boldsymbol{\tau}$, agents $\boldsymbol{A}$, temporal constraints (i.e., upper- and lowerbound temporal constraints) relating tasks in the problem $\boldsymbol{TC}$, and the set of task pairs that require the same resources and can therefore not be executed at the same time, $\boldsymbol{\tau_R}$. 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 *if* $f_{act}(\tau_i^*)$ predicts that τ_i^* should be scheduled at the current time. Note that iteration over agents (Line 2) can be performed according to a specified ordering, or one can alternatively learn a more general priority function to select and schedule the best agent/task pair using $f_{priority}\left(\langle\tau_i, a\rangle, \langle\tau_x, a'\rangle\right)$, $f_{act}\left(\langle\tau_i, a\rangle^*\right)$. In the latter case, the features γ_{τ_i} are mapped to agent/task pairs rather than tasks.

One benefit of the pairwise ranking formulation is the ability to apply any 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 compared the performance of a decision tree, support vector machine and other common classification techniques.

Algorithm 1 Pseudocode for an Apprentice Scheduler

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

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

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

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

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

Data Sets

Next, we validate that schedules produced by our learned policies are of comparable quality to those generated by human or synthetic experts. We considered a synthetic data set from the **XD [ST-SA-TA]** class of problems and a real-world data set from the **CD [MT-MA-TA]** class of problems defined by Korsah et al. [Korsah *et al.*, 2013].

Synthetic Data Set

For our first investigation, we generated a synthetic data set of scheduling problems in which agents were assigned to complete a set of tasks. Tasks were related through precedence or wait constraints as well as deadline constraints, which could be absolute (relative to the start of the schedule) or relative to another task's start or finish time. Agents were required to access a set of shared resources to execute each task (e.g., the task's physical location). Agents and tasks had defined starting locations, and task locations were static. Each agent traveled at a constant speed between task locations, and agents were only able to perform tasks when present at the corresponding task location. Task completion times were potentially non-uniform and agent-specific, as would be the case for heterogeneous agents. An agent that was incapable of performing a task was assumed to have an infinite completion time for that task. The objective was to minimize the makespan or other time-based performance measures.

This problem definition spans a range of scheduling problems, including the traveling salesman, job-shop scheduling, multi-vehicle routing and multi-robot task allocation problems, among others. We describe this range as a vehicle routing problem with time windows, temporal dependencies, and resource constraints (VRPTW-TDR), which falls within the **XD [ST-SA-TA]** class in the taxonomy by [Korsah *et al.*, 2013]: agents perform tasks sequentially (ST), each task requires one agent (SA) and commitments are made over time (TA).

To generate our synthetic data set, we developed a mock scheduling expert that applies one of a set of context-dependent rules based on the composition of the given scheduling problem. This behavior was based upon rules presented in prior work addressing these types of problems [Gombolay *et al.*, 2013; Gombolay and Shah, 2014; Solomon, 1987; Tan *et al.*, 2001]. Our objective was to show that our apprenticeship scheduling algorithm learns both context-dependent rules and how to identify the associated context for their correct application.

The mock scheduling expert functions as follows: First, the algorithm collects all alive and enabled tasks $\tau_i \in \boldsymbol{\tau_{AE}}$ as defined by [Muscettola *et al.*, 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 $\boldsymbol{\tau}$.

Next, the heuristic iterates over each agent to identify the highest-priority task, τ_i^*, to schedule for that agent. The algorithm determines which scheduling rule is most appropriate to apply for each agent. If agent speed is sufficiently slow (≤ 1 m/s), travel time will become the major bottleneck. If agents move quickly but utilize one or more resources R heavily ($\sum_{\tau_i} \sum_{\tau_x} 1_{R_{\tau_i} = R_{\tau_x}} \geq$ c for some constant c), use of these resources can become the bottleneck. Otherwise, task durations and associated wait constraints are generally most important.

If the algorithm identifies travel distance as the primary bottleneck, it chooses the next task by applying a priority rule well-suited for vehicle routing that minimizes a weighted, linear combination of features [Gambardella *et al.*, 1999; Solomon, 1987] comprised of the distance and angle relative to the origin between agent a and τ_x. This rule is depicted in Equation 5, where $\vec{l_x}$ is the location of τ_x, $\vec{l_a}$ is the location of agent a, θ_{xa} is the relative angle between the vector from origin to the agent location and the origin to the location of τ_x and α_1 and α_2 are weighting constants.

$$
\tau_i^* \leftarrow \underset{\tau_x \in \boldsymbol{\tau_{AE}}}{\text{argmin}} \left(\|\vec{l_x} - \vec{l_a}\| + \alpha_1 \theta_{xa} + \alpha_2 \|\vec{l_x} - \vec{l_a}\| \theta_{xa} \right) \tag{5}
$$

If the algorithm identifies resource contention as the most important bottleneck, it employs a rule to mitigate resource-contention in multi-robot, multi-resource problems based on prior work in scheduling for multi-robot teams [Gombolay *et al.*, 2013]. Specifically, the algorithm uses Equation 6 to select the high-priority task to schedule next, where d_{τ_x} is the deadline of τ_x and α_3 is a weighting constant.

$$\tau_i^* \leftarrow \underset{\tau_x \in \boldsymbol{\tau_{AE}}}{\operatorname{argmax}} \left(\left(\sum_{\tau_i} \sum_{\tau_x} 1_{R_{\tau_i} = R_{\tau_x}} \right) - \alpha_3 d_{\tau_x} \right) \qquad (6)$$

If the algorithm decides that temporal requirements are the major bottleneck, it employs an Earliest Deadline First rule (Equation 7), which performs well across many scheduling domains [Chen and Askin, 2009; Gombolay *et al.*, 2013; Gombolay and Shah, 2014].

$$\tau_i^* \leftarrow \underset{\tau_x \in \boldsymbol{\tau_{AE}}}{\operatorname{argmin}}\, d_{\tau_x} \qquad (7)$$

After selecting the most important task, τ_i^*, the algorithm determines whether the resource required for τ_i^*, $R_{\tau_i^*}$, is idle and whether the agent is able to travel to the task location by time t. If these constraints are satisfied, the heuristic schedules task τ_i^* at time t. (An agent is able to reach task τ_i^* if $t \geq f_j + k\,(l_i - l_j) / \|l_i - l_j\|$ for all $\tau_j \in \boldsymbol{\tau}$ that the agent has already completed, where k is the agent's speed.)

We constructed the synthetic data set for two homogeneous agents and 20 partially ordered tasks located within a 20 x 20 grid.

Real-World Data Set

We collected a real-world data set consisting of human demonstrators of various skill levels solving the anti-ship missile defense (ASMD) weapon-to-target assignment problem. In this problem, one must determine how to deploy a set of soft kill weapons, or decoys, to prevent enemy anti-ship missiles from impacting one's own ship. These decoys represent the agents, and the neutralization of each missile represents a task. 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 upon the time history of all other decoy deployments h. Decoys are able to distract many missiles (MT), and many decoys can be used to distract the same missile at 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, as well as where and when they should be deployed. Agents and tasks have defined starting locations. Each task (missile) is modeled as a dynamical system with a homing function $F_\tau(h, t)$ that guides the missile toward its target and is a function of the current time and the time history of previous decoy deployments. Decoys travel at a constant speed to their target locations x_g from the ship that deploys them. This ASMD problem falls into the **CD [MT-MA-TA]** class variant.

Data was collected from domain experts playing a serious game, called Strike Group Defender[1] (SGD), for ASMD training. Game scenarios involved five types of decoys and 10 types of threats. The threats were randomly generated for each played scenario, thereby promoting the development of strategies that were robust to a varied distribution of threat scenarios. Each decoy had a specified effectiveness against each threat type. Players attempted to deploy a set of decoys using the correct types at the correct locations and times in order to distract incoming missiles. Threats were launched over time; an effective deployment at time t could become counterproductive in the future 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 were received for each second a threat spent homing in on a decoy. $5,000$ points were deducted for each threat impact and 1 point was deducted for each second a threat spent homing in the players ship. 25-$1,000$ points were subtracted for each decoy deployment, with the deducted point value depending upon the decoy type.

The collected data set consisted of 311 games played by 35 humans across 45 threat configurations, or "levels". From this set, we also separately analyzed 16 threat configurations such that each configuration included at least one human demonstration in which the ship was protected from all enemy missiles. For these 16 threat configurations, there were 162 total games played by 27 unique human demonstrators. The player cohort consisted of technical fellows and associates, as well as contractors at a federally funded research and development center (FFDRC), and their expertise varied from "generally knowledgeable about the ASMD problem" to "domain experts" with professional experience or training in ASMD.

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, was 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} was comprised of the task's deadline, a binary indicator for whether or not the task's precedence constraints had been satisfied, the number of other tasks sharing the given task's resource, a binary indicator for whether or not the given task's resource was available, the travel time remaining to reach the task location, 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 pointwise approach and a naïve approach. In the pointwise approach, training examples for selecting the highest-priority task were of the form ${}^{rank}\phi_{\tau_i}^m := [\xi_\tau, \gamma_{\tau_i}]$. 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 concatenated the high-level features of the task set and the task-specific features of

[1] SGD was developed by Pipeworks Studio in Eugene, Oregon, USA.

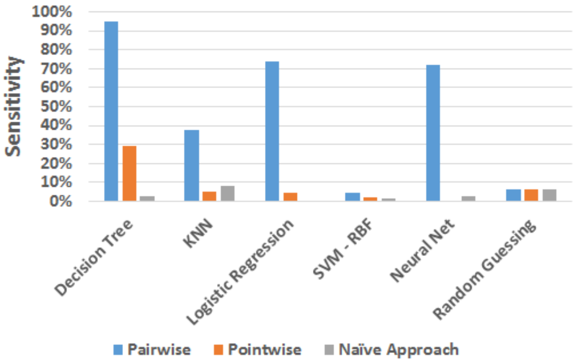

Figure 1: Sensitivity of machine learning techniques using the pairwise, pointwise and naïve approaches.

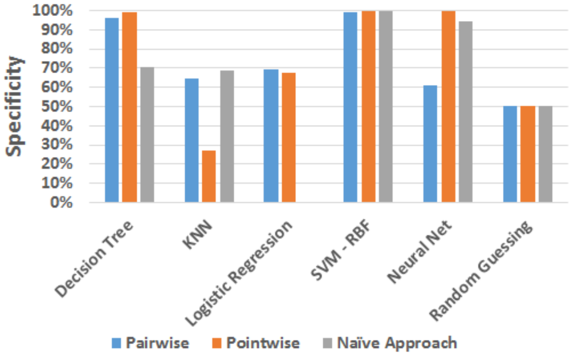

Figure 2: Specificity of machine learning techniques using the pairwise, pointwise and naïve approaches.

the form ${}^{rank}\phi^m := [\xi_{\boldsymbol{\tau}}, \gamma_{\tau_1}, \gamma_{\tau_2}, \ldots, \gamma_{\tau_n}]$; labels y^m were given by the index of the task τ_i scheduled in observation m.

Figures 1-2 depict the sensitivity (true positive rate) and specificity (true negative rate) of the model, respectively. We found that a pairwise model outperformed the pointwise and naïve approaches. Within the pairwise model, a decision tree yielded 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.

In order 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, we trained decision trees with our pairwise model using 15, 150 and 1,500 demonstrations. The sensitivity and specificity depicted in Figures 3-4 for 15 and 150 demonstrations are the mean sensitivity and specificity of 10 models trained via random subsampling without replacement. We also varied the quality of the training examples, assuming the demonstrator was operating under an ϵ-greedy approach with a $(1-\epsilon)$ probability 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 to select 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 the modal nature of the multifaceted scheduling heuristic. Note that this dataset was composed of scheduling strategies with mixed discrete-continuous functional components; performance could potentially be improved upon in future work by combining decision trees with logistic regression. This hybrid learning approach has been successful in prior machine learning classification tasks [Landwehr *et al.*, 2005] and can be readily applied to this apprenticeship scheduling framework. There is also an opportunity to improve performance through hyper-parameter tuning (e.g. to select the minimum number of examples in each leaf of the decision tree). Comprehensive investigation of the relative benefits for a range of learning techniques is left for future work.

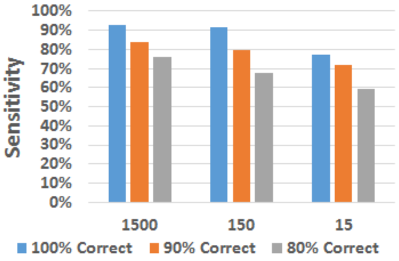

Figure 3: Sensitivity for a pairwise decision tree varying the number and proportion of correct demonstrations.

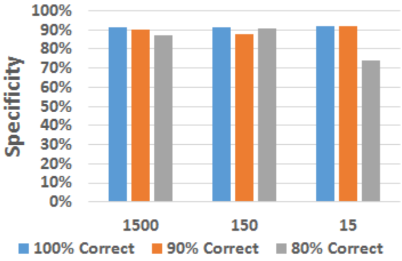

Figure 4: Specificity for a pairwise decision tree varying the number and proportion of correct demonstrations.

Real-World Data Set

We trained and tested a decision tree on our pairwise scheduling model via leave-one-out cross-validation using 16 real demonstrations in which a player successfully protected the ship from all enemy missiles. Each demonstration originated from a unique threat scenario. Features for each decoy/missile pair (or null decoy deployment due to inaction) included indicators for whether a decoy had been placed such that a missile was successfully distracted by that decoy, whether a missile would be lured into hitting the ship due to decoy placement, or whether a missile would be unaffected by decoy placement.

Across all 16 scenarios, the mean player score was $74,728$ ± $26,824$. With only 15 examples of expert human demonstrations, our apprenticeship scheduling model achieved a mean score of $87,540$, with a standard deviation of $16,842$.

We performed statistical analysis to evaluate our hypothesis that the scores produced by the learned policy would be statistically significantly better than the scores achieved by the human demonstrators. The null hypothesis stated that the number of scenarios in which the apprenticeship scheduling model achieved superior performance would be less than or equal to the number of scenarios in which the mean score of the human demonstrators was superior to that of the apprenticeship scheduler. We set the significance level at $\alpha = 0.05$, which means that the risk of identifying a difference between the mean scores earned by the apprenticeship scheduler and the set of human performers when no such difference exists is less than 5%.

Results from a binomial test rejected the null hypothesis, indicating that the learned scheduling policy performed better than the human demonstrators in significantly more scenarios (12 versus 4 scenarios; $p < 0.011$). In other words, we can say with 95% certainty that the apprenticeship scheduler outperformed the average human player for the majority of the presented missile defense scenarios. This promising result was achieved using a relatively small training set, and suggests that learned policy can form the basis for a training tool to improve the average player's score.

Conclusions

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

Acknowledgements

We would like to thank Commander Scott Orosz, USA, Retired, Deputy Director of Electronic Warfare Programs for the US Navy (PMR-51). This work was supported by the National Science Foundation Graduate Research Fellowship Program under grant number 2388357.

References

[Abbeel and Ng, 2004] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

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

[Berry *et al.*, 2011] Pauline M. Berry, Melinda Gervasio, Bart Peintner, and Neil Yorke-Smith. Ptime: Personalized assistance for calendaring. *ACM Trans. Intell. Syst. Technol.*, 2(4):40:1–40:22, July 2011.

[Brunet *et al.*, 2008] Luc Brunet, Han-Lim Choi, and Jonathan P. How. Consensus-based auction approaches for decentralized task assignment. In *Proc. AIAA GNC*, Honolulu, HI, 2008.

[Busoniu *et al.*, 2008] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. SMC Part C*, 38(2):156–172, March 2008.

[Cao *et al.*, 2007] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proc. ICML*, pages 129–136. ACM, 2007.

[Chen and Askin, 2009] Jiaqiong Chen and Ronald G. Askin. Project selection, scheduling and resource allocation with time dependent returns. *European Journal of Operational Research*, 193:23–34, 2009.

[Cheng *et al.*, 2006] Tsang-Hsiang Cheng, Chih-Ping Wei, and Vincent S. Tseng. Feature selection for medical data mining: Comparisons of expert judgment and automatic approaches. In *Proc. CBMS*, pages 165–170, 2006.

[Chernova and Veloso, 2007] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proc. AAMAS*, pages 233:1–233:8. ACM, 2007.

[Gambardella *et al.*, 1999] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. MACS-VRPTW: A multiple colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.

[Gombolay and Shah, 2014] Matthew Gombolay and Julie Shah. Schedulability analysis of task sets with upper- and lower-bound temporal constraints. *Journal of Aerospace Information Systems*, 11(12):821–841, 2014.

[Gombolay *et al.*, 2013] Matthew Gombolay, Ronald Wilcox, and Julie Shah. Fast scheduling of multi-robot teams with temporospatial constrints. In *Proc. RSS*, Berlin, Germany, June 24-28 2013.

[Huang and Mutlu, 2014] Chien-Ming Huang and Bilge Mutlu. Learning-based modeling of multimodal behaviors for humanlike robots. In *Proc. HRI*, pages 57–64, 2014.

[Jin *et al.*, 2008] Rong Jin, Hamed Valizadegan, and Hang Li. Ranking refinement and its application to information retrieval. In *Proc. Conference on WWW*, pages 397–406. ACM, 2008.

[Jones *et al.*, 2011] E. Gil Jones, M. Bernardine Dias, and Anthony Stentz. Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous Robots*, 30(1):41–56, 2011.

[Kehle *et al.*, 2011] Shannon M. Kehle, Nancy Greer, Indulis Rutks, and Timothy Wilt. Interventions to improve veterans? access to care: A systematic review of the literature. *Journal of General Internal Medicine*, 26(2):689–696, 2011.

[Konidaris and Barto, 2007] George Konidaris and Andrew Barto. Building portable options: Skill transfer in reinforcement learning. In *Proc. IJCAI*, pages 895–900, 2007.

[Konidaris *et al.*, 2011] George Konidaris, Sarah Osentoski, and Philip Thomas. Value function approximation in reinforcement learning using the fourier basis. In *Proc. AAAI*, pages 380–385, 2011.

[Korsah *et al.*, 2013] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *IJRR*, 32(12):1495–1512, 2013.

[Landwehr *et al.*, 2005] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic model trees. *Machine Learning*, 59(1-2):161–205, 2005.

[Lee *et al.*, 2003] Zne-Jung Lee, Shun-Feng Su, and Chou-Yuan Lee. Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *IEEE Trans. SMC Part B*, 33(1):113–121, Feb 2003.

[Liu and Shell, 2013] Lantao Liu and Dylan A. Shell. Optiml market-based multi-robot task allocation via strategic pricing. In *Proc. RSS*, Berlin, Germany, June 24-28 2013.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

[Muscettola *et al.*, 1998] Nicola Muscettola, Paul Morris, and Ioannis Tsamardinos. Reformulating temporal plans for efficient execution. In *Proc. KR&R*, Trento, Italy, June 2-5, 1998.

[Nunes and Gini, 2015] Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *Proc. AAAI*, pages 2110–2116, 2015.

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

[Page *et al.*, 1999] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. SIDL-WP-1999-0120.

[Pahikkala *et al.*, 2007] Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jorma Boberg, and Tapio Salakoski. Learning to rank with pairwise regularized least-squares. In *SIGIR Workshop on Learning to Rank for Information Retrieval*, pages 27–33, 2007.

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

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

[Raghavan *et al.*, 2006] Hema Raghavan, Omid Madani, and Rosie Jones. Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7:1655–1686, December 2006.

[Ramanujam and Balakrishnan, 2011] Varun Ramanujam and Hamsa Balakrishnan. Estimation of maximum-likelihood discrete-choice models of the runway configuration selection process. In *Proc. ACC*, pages 2160–2167, June 2011.

[Sammut *et al.*, 1992] Claude Sammut, Scott Hurst, Dana Kedzier, and Donal Michie. Learning to fly. In *Proc. ICML*, pages 385–393, July 1992.

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

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

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

[Valizadegan *et al.*, 2009] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. Learning to rank by optimizing NDCG measure. In *NIPS*, pages 1883–1891, 2009.

[Vogel *et al.*, 2012] Adam Vogel, Deepak Ramach, Rakesh Gupta, and Antoine Raux. Improving hybrid vehicle fuel efficiency using inverse reinforcement learning. In *Proc. AAAI*, pages 384–390, 2012.

[Volkovs and Zemel, 2009] Maksims N. Volkovs and Richard S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In *Proc. ICML*, pages 1089–1096, 2009.

[Wang and Usher, 2005] Yi-Chi Wang and John M. Usher. Application of reinforcement learning for agent-based production scheduling. *Eng. Appl. Artif. Intell.*, 18(1):73–82, February 2005.

[Wu *et al.*, 2011] Jun Wu, Xin Xu, Pengcheng Zhang, and Chunming Liu. A novel multi-agent reinforcement learning approach for job scheduling in grid computing. *Future Generation Computer Systems*, 27(5):430 – 439, 2011.

[Zhang and Dietterich, 1995] Wei Zhang and Thomas G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proc. IJCAI*, pages 1114–1120. Morgan Kaufmann, 1995.

[Zheng *et al.*, 2015] Jiangchuan Zheng, Siyuan Liu, and Lionel Ni. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *Proc. AAAI*, pages 2198–2205, 2015.

[Ziebart *et al.*, 2008] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438, 2008.