Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2015-025 July 2, 2015

PhD Thesis Proposal: Human-Machine Collaborative Optimization via Apprenticeship Scheduling

Matthew C. Gombolay

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PH.D. THESIS PROPOSAL

DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS

Human-Machine Collaborative Optimization via Apprenticeship Scheduling

Ph.D. Candidate:
Matthew GOMBOLAY

Thesis Committee:
Prof. Hamsa BALAKRISHNAN
Prof. Bilge MUTLU
Prof. Julie SHAH (Chair)
Prof. Peter SZOLOVITS
Prof. Andrea THOMAZ

External Evaluator:
Prof. Warren HOBURG

June 28, 2015

Abstract

Resource optimization in health care, manufacturing, and military operations requires the careful choreography of people and equipment to effectively fulfill the responsibilities of the profession. However, resource optimization is a computationally challenging problem, and poorly utilizing resources can have drastic consequences. Within these professions, there are human domain experts who are able to learn from experience to develop strategies, heuristics, and rules-of-thumb to effectively utilize the resources at their disposal. Manually codifying these heuristics within a computational tool is a laborious process and leaves much to be desired. Even with a codified set of heuristics, it is not clear how to best insert an autonomous decision-support system into the human decision-making process. The aim of this thesis is to develop an autonomous computational method for learning domain-expert heuristics from demonstration that can support the human decision-making process. We propose a new framework, called apprenticeship scheduling, which learns and embeds these heuristics within a scalable resource optimization algorithm for real-time decision-support. Our initial investigation, comprised of developing scalable methods for scheduling and studying shared control in human-machine collaborative resource optimization, inspires the development of our apprenticeship scheduling approach. We present a promising, initial prototype for learning heuristics from demonstration and outline a plan for our continuing work.

I. Introduction

Resource optimization and scheduling is a costly, challenging problem that affects almost every aspect of our lives. One example that affects each of us is health care. Poor systems design and scheduling of resources can have drastic consequences on patient wait times. Patients with non-urgent needs who experience prolonged wait times have higher rates of noncompliance and missing appointments [53], [78]. Prolonged wait times and other inefficiencies in patient care contribute to the dissatisfaction and burnout of health care providers [92]. This issue is so critical that the Institute of Medicine recently released a report highlighting the need for better practices in scheduling and resource optimization [12].

In automotive manufacturing, BMW produces approximately 1 car every 60 seconds in their facility in Spartanburg, South Carolina. In this plant, approximately 80% of the cars are customized for individual customers. This customization requires the tight choreography of supply chain management and assembly. When a part shortage for one car on an assembly line occurs, every car is held until the conflict can be resolved. Every 60 seconds spent re-scheduling work in response to a disturbance costs the company tens of thousands of dollars [1]. The Boeing Company similarly offers a high level of customization to its customers. Building an airplane is a complex process. For example, construction of a Boeing 747 requires the assembly of 6 million individual parts, a subset of which are customized for each patron. Every minute re-scheduling in response to dynamic disruptions in the build process can cost up to $\$100,000$.[2].

The military is also highly invested in the effective use of resources. In naval conflict, defending one's ship from enemy anti-ship missiles is a challenging task. This problem requires the weighing of the pros and cons of using interceptors to attempt to destroy the incoming anti-ship missile engagement versus deploying decoys to attempt to divert the attack. The relative cost and effectiveness of these assets necessitates deliberate use. However, the cost of taking time to weigh options and the cost of suboptimal resource allocation can cost the life of the decision-maker and the lives of his/her shipmates.

The fields of artificial intelligence (AI) planning and scheduling and operations research (OR) strive to develop algorithms to improve resource allocation. However, there are two primary challenges to improving resource allocation. First, the problem of optimal task allocation and sequencing with upper and lower-bound temporal constraints (i.e., deadlines and wait constraints) is NP-Hard [7]. The computational complexity of optimally assigning A agents to N tasks and ordering those tasks is $O\left(2^{AN}N!\right)$. Some of the best work in developing autonomous techniques to assign agents (or resources) to perform a set of tasks employ decomposition techniques, which allow for more efficient computation [13], [44]. When constraints between agents, tasks, and resources become highly interdependent, these techniques can lose their advantage. Near-optimal approximation techniques for these problems exist [98], [27], [36], [51]. These methods rely on the algorithm designer crafting clever heuristics based on domain expertise to decompose or structure the scheduling problem and prioritize the manner in which resources are allocated and tasks are sequenced. However, manually capturing this domain knowledge and developing effective algorithms for resource optimization remains a challenging process, and the solution techniques are often domain-specific.

The second challenge is fully defining the optimization problem and encoding the knowledge of the domain experts into a mathematically precise formulation. For example, recent work in developing human in-the-loop decision support systems for the scheduling of aircraft carrier deck operations shows that the heuristics applied by the naval personnel outperform a mathematical program solver [85]. The key here is that the mathematical program did not have full knowledge of the hard and soft constraints and the model of the uncertainty in the environment that the naval personnel had learned over many months or years of experience and apprenticeship.

The aim of this thesis is to develop an autonomous system that 1) learns the heuristics and implicit rules-of-thumb developed by domain experts from years of experience, 2) embeds and leverages this domain knowledge within a scalable resource optimization framework to reduce the computational search space, 3) and provides decision support in a way that engages users and benefits them in their decision-making process. By intelligently leveraging the ability of humans to learn heuristics and the speed of modern computation, we can solve the challenging requirements of real-time decision-support for these domains.

The ability to seamlessly learn the heuristics from demonstration via domain experts and embed this within a scalable resource optimization framework to solve multi-agent, real-world problems remains an open problem. While there has been extensive prior work in scheduling, machine learning, and human factors engineering, there is a gap in the literature that effectively brings these sciences together to learn and utilize expert heuristics for resource optimization. For example, Ryan *et al.* found that domain-expert heuristics, manually documented, can outperform the performance of a hand-crafted mathematical program [85]. However, we propose an autonomous agent that would learn these heuristics and directly embed them within the resource optimization (or mathematical program).

In Section II, we motivate our resource optimization problem with an application in the obstetrics field by coordinating resources on the labor and delivery floor at Beth Israel Deaconess Medical Center. We define the problem statement and contributions of this thesis in Section III and discuss our approach in the context of the prior literature in Section

[1]BMW's facility in Spartanburg, SC produced $36,580$ cars in March 2015. The base price of the cheapest car produced at the facility is $\approx \$38,500$. Assuming a $24/7$ work week, that results in $\approx \$31,548$ of revenue earned/lost every minute. Numbers are courtesy of BMW USA.

[2]Boeing's facility in Renton, WA is increasing production of Boeing-737 aircraft to 47 per month by 2017. RyanAir and Boeing recently agreed to a purchase of 100 Boeing 737-Max aircraft for $\$11$ billion, which is approximately $\$110$ million per plane. Assuming a $24/7$ work week, that results in $\approx \$108,603$ of revenue earned/lost every minute. Numbers are courtesy of The Boeing Company.

IV. We present our work to date on a scheduling domain in manufacturing in Sections V and VI. This work focuses on a manufacturing domain where we use our personal domain expertise to carefully decompose and structure the problem for a fast, empirically near-optimal approximation method. We also demonstrate how this method can be used to coordinate manufacturing resources, vís a vís a human-robot team, to improve team performance and the desire of human workers to utilize this autonomous capability. This work motivates the use of autonomous algorithms for resource optimization and the need for clever heuristics to make the problem computationally tractable. We then present a prototype for how we can autonomously learn such heuristics from domain experts in Section VII. In Section VIII, we discuss a synthetic data set and a real-world data set we will use to validate our method for learning resource optimization heuristics from domain experts. In Section X, we discuss our proposed work to be completed, and we present a time line to finish this work in Section XI.

II. Motivation: Resource Nurse in Labor & Delivery

A labor and delivery floor in an obstetrics department typically consists of nurses and doctors whose activities must efficiently coordinate their activities to care for a set of patients with finite resources. A single *resource nurse* is responsible for coordinating these resources during any given work shift. Specifically, a resource nurse must decide the following:

1) Nurse Assignment: Each patient needs to be assigned to a labor nurse while the patient is on the labor floor. Nurses can handle caring for a set of patients in parallel provided that the acuity of the patient does not exceed the limits of the nurse's bandwidth.
2) Resource Assignment: It is the responsibility of the resource nurse to assign a room to each patient at each moment in time. The type of room suitable for assignment depends on the type of treatment that needs to be administered to the patient. For example, a patient requiring a cesarean section requires an operating room for the surgery, whereas a patient who will have a spontaneous vaginal delivery requires only a normal labor room.
3) Start and Finish Times: While the duration of events on the labor floor are not generally controllable, the resource nurse is able to partially control when certain phases of care begin and end. For example, a nurse can delay a cesarean section for one patient because a second, more acute patient has a more urgent need for a cesarean section.
4) Patient Acceptance or Diversion: The resource nurse must decide whether each patient can be treated by the labor floor. The resource nurse may decide to close the labor floor to incoming patients, for example, if the labor floor is too busy.

We can readily formulate the resource nurse's job as a stochastic optimization problem where the goal is to minimize z (Equation 1) such that the probability of satisfying Equations 2-10 is greater than or equal to $1-\epsilon$.

$$\min z,$$

$$z \geq fn\left(\{{}^tA^a_{\tau_i^j}\},\{G^a_{\tau_i^j}\},\{{}^tR^r_{\tau_i^j}\},\{H_{\tau_i}\},\{s_{\tau_i^j},f_{\tau_i^j}\}\right) \quad (1)$$

$$\sum_{a\in A} {}^tA^a_{\tau_i^j} \geq 1 - M\left(1-H_{\tau_i}\right), \quad \forall \tau_i^j \in \boldsymbol{\tau}, \forall t \in \{0,1,\ldots,T\} \quad (2)$$

$$M\left(2-{}^tA^a_{\tau_i^j}-H_{\tau_i}\right) \geq -U_{\tau_i^j} + {}^tG^a_{\tau_i^j} \geq M\left({}^tA^a_{\tau_i^j}+H_{\tau_i}-2\right), \forall \tau_i^j \in \boldsymbol{\tau}, \forall t \in \{0,1,\ldots,T\} \quad (3)$$

$$\sum_{\tau_i^j \in \boldsymbol{\tau}} {}^tG^a_{\tau_i^j} \leq C_a, \forall a \in A, \forall t \in \{0,1,\ldots,T\} \quad (4)$$

$$\sum_{r\in R} {}^tR^r_{\tau_i^j} \geq 1 - M\left(1-H_{\tau_i}\right), \quad \forall \tau_i^j \in \boldsymbol{\tau}, \forall t \in \{0,1,\ldots,T\} \quad (5)$$

$$\sum_{\tau_i^j \in \boldsymbol{\tau}} {}^tR^a_{\tau_i^j} \leq 1, \forall r \in R, \forall t \in \{0,1,\ldots,T\} \quad (6)$$

$$ub_{\tau_i^j} \geq f_{\tau_i^j} - s_{\tau_i^j} \geq lb_{\tau_i^j}, \forall \tau_i^j \in \boldsymbol{\tau} \quad (7)$$

$$s_{\tau_x^y} - f_{\tau_i^j} \geq W_{\langle \tau_i,\tau_j\rangle}, \forall \tau_i,\tau_j \in \boldsymbol{\tau} | , \forall W_{\langle \tau_i,\tau_j\rangle} \in \boldsymbol{TC} \quad (8)$$

$$f_{\tau_x^y} - s_{\tau_i^j} \leq D^{rel}_{\langle \tau_i,\tau_j\rangle}, \forall \tau_i,\tau_j \in \boldsymbol{\tau} | \exists D^{rel}_{\langle \tau_i,\tau_j\rangle} \in \boldsymbol{TC} \quad (9)$$

$$f_{\tau_i^j} \leq D^{abs}_{\tau_i}, \forall \tau_i \in \boldsymbol{\tau} | \exists D^{abs}_{\tau_i} \in \boldsymbol{TC} \quad (10)$$

We model each step in the care of a patient as a subtask τ_i^j. Task τ_i then represents the set of steps required to care for patient i. Labor nurses are modeled as agents who have a finite capacity to process tasks in parallel, where each subtask requires a variable amount of this capacity. Rooms in the labor floor (e.g., a labor room, an operating room, etc.) are modeled as resources, which process subtasks in series. Agents and resource assignments to subtasks are preemptable, meaning that the agent and resource assigned to care for any patient during any step in the care process may be changed during that step.

In this formulation, ${}^tA^a_{\tau_i^j} \in \{0,1\}$ is a binary decision variable for assigning agent a to subtask τ_i^j, the j^{th} subtask of task τ_i, during time epoch $[t,t+1)$. Similarly, ${}^tG^a_{\tau_i^j}$ is an integer decision variable for assigning a certain portion of the effort of agent a to task i during phase j during time epoch $[t,t+1)$. ${}^tR^r_{\tau_i^j} \in \{0,1\}$ is a binary decision variable for whether subtask τ_i^j is assigned resource r during time epoch $[t,t+1)$. $H_{\tau_i} \in \{0,1\}$ is a binary decision variable for whether task τ_i and its corresponding subtasks are to be completed. $U_{\tau_i^j}$ specifies the effort required from any agent to work on τ_i^j. As before in the formulation in Section V-A, $s_{\tau_i^j}, f_{\tau_i^j} \in [0,\infty)$ are the start and finish times of τ_i^j.

Equation 1 is a general objective that is a function of the decision variables $\{{}^tA^a_{\tau_i^j}\}$, $\{G^a_{\tau_i^j}\}$, $\{{}^tR^r_{\tau_i^j}\}$, $\{H_{\tau_i}\}$, and $\{s_{\tau_i^j}, f_{\tau_i^j} | \tau_i^j \in \boldsymbol{\tau}\}$. Equation 2 enforces that each subtask τ_i^j during each time epoch $[t, t+1)$ is assigned one agent. Equation 3 ensures that each subtask τ_i^j receives a sufficient portion of the effort of its assigned agent a during time epoch $[t, t+1)$. Equation 4 ensures that agent a is not oversubscribed. In other words, the sum of the burden of each subtask τ_i^j assigned to agent a during time epoch $[t, t+1)$ is less than or equal to the capacity C_a of agent a. Equation 5 ensures that each subtask τ_i^j of each task τ_i that is to be completed (i.e., $H_{\tau_i} = 1$) is assigned one resource r. Equation 6 ensures that each resource r is assigned to only one subtask during each epoch $[t, t+1)$

As in the problem formulation in Section V-A, Equation 7 requires the duration of subtask τ_i^j to be less than or equal to $ub_{\tau_i^j}$ and at least $lb_{\tau_i^j}$ units of time. Equation 8 requires that τ_x^y occurs at least $W_{\langle \tau_i^j, \tau_x^y \rangle}$ units of time after τ_i^j. Equation 9 requires that the duration between the start of τ_i^j and the finish of τ_x^y is less than $D^{rel}_{\langle \tau_i^j, \tau_x^y \rangle}$. Equation 10 requires that τ_i^j finishes before $D^{abs}_{\tau_i^j}$ units of time has expired since the start of the schedule.

The stochasticity of the problem arises from the uncertainty in the upper and lowerbound of the durations $(ub_{\tau_i^j}, lb_{\tau_i^j})$ of each of the steps in caring for a patient, the number and types of patients $\boldsymbol{\tau}$, and the temporal constraints $\boldsymbol{TC}$ relating the start and finish of each step of care. These variables are a function of the resource allocation variables $\{{}^tR^a_{\tau_i^j}\}$. For example, a patient assigned to the labor ward might have her labor more actively managed (e.g., expedited with oxytocin or some other intervention), which would in turn change the duration of that stage of labor. In other words, the duration of the subtask, τ_i^j, representing that stage of labor would be affected vís a vís random variables $\{ub_{\tau_i^j}, lb_{\tau_i^j}\}$. Mathematically, we have $\left(\{ub_{\tau_i^j}, lb_{\tau_i^j} | \tau_i^j \in \boldsymbol{\tau}\}, \boldsymbol{\tau}, \boldsymbol{TC}\right) \sim P(\{{}^tR^a_{\tau_i^j}, \forall t \in [0, 1, \ldots, T]\})$.

We can determine the computational complexity of satisfying the constraints in Equations 2-10. Given $|A|$ agents where each agent has an integer processing capacity of C_a, n tasks τ_i each with m_i subtasks, $|R|$ resources, and an integer-valued planning horizon of T units of time, the complexity of satisfying these constraints is $O\left(2^{|A||R|T^2} C_a^{|A|T}\right)$. In practice, there are approximately 10 nurses (agents) who can carry up to two patients at a time (i.e., $C_a = 2, \forall a \in A$), 20 different rooms (resources) of varying types (e.g., operating room, recovery room, triage, etc.), 20 patients (tasks) at any one time, and a planning horizon of 12 hours or 720 minutes, which yields a worst-case complexity of $\sim 2^{10*20*720^2} 2^{10*720} \geq 2^{10^6}$.

It is natural that one might seek to reduce the complexity by increasing the duration of each time epoch, which decreases the total number of epochs. Another approach would be to constrain the tasks to be non-preemptable, which would allow for the removal of time epochs completely although additional sequencing variables and constraints would need to be introduced. However, in labor and delivery, as with many other domains, the scheduling mechanism needs to be able to account for minute-to-minute (or even second-to-second) changes to the state of the labor floor and preemption is often necessary. For example, the health of patients can sometimes require emergent intervention where a delay of even minutes can result in death (e.g., umbilical cord prolapse, fetal bradycardia, antepartum hemorrhage, etc.). However, even if we ignore the planning horizon (i.e., $T = 1$), then the computational complexity would still be intractable at $2^{10*20} 2^{10} \sim 1.6 \times 10^{63}$.

Furthermore, the variance of the time and resources required to care for patients is so great that determining a schedule with a high probability of feasibility *a priori* may not be possible except for the trivial solution of rejecting most or all patients (i.e., $H_{\tau_i} = 0, \forall \tau_i$). Even if the variance allowed for the generation of a schedule with a high probability of success, human factors studies of decision-making shows human experts typically make these complex scheduling decisions in real-time to satisfy relevant constraints rather than taking the time to search for the optimal solution [56]. These experts develop implicit rules-of-thumb, heuristics to make their scheduling decisions on-the-fly. As such, we seek to develop a computational technique to learn from demonstration these heuristics in the form of a "policy" to provide the ability to autonomously coordinate resources in labor and delivery, manufacturing, military applications, etc.

III. Problem Statement and Contributions

The goal of this work is to bring together scheduling, machine learning, and human factors research to create an autonomous capability to optimize resource utilization in complex domains with temporospatial constraints. The proposed work focuses on high-intensity domains, such as health care, manufacturing, and military operations, where human experts must coordinate a finite number of resources to accomplish a set of tasks as efficiently as possible. Resource optimization in these domains often requires significant *a priori* planning and constant adjustment online in response to dynamic disturbances. This planning is often performed by people because exact computational methods suffer from scalability issues, and encoding domain knowledge in a computational tool is difficult. Furthermore, proprietary techniques, which are developed through an army of consultants, are financially costly and leave much to be desired [33]. These domains can benefit greatly from algorithms that embed autonomously-learned heuristics and rules of thumb used by domain experts within an optimization framework to improve resource utilization in real-time.

Our motivating application is the problem of resource scheduling on a labor & delivery floor at Beth Israel Deaconess Medical Center, as described in Section II. We investigate this application because 1) it serves as one of the hardest types of problems in scheduling according to the taxonomy defined by Korsah, Stenz, and Dias [58] and 2)

we have access to a cohort of well-trained resource nurses who currently perform this role.

There are three building blocks to achieving and demonstrating the goals of this thesis. First, we must understand how to develop scalable representations for scheduling with complex temporospatial constraints. As such, we investigate a more restricted version of the resource-nurse scheduling problem. This problem, which we describe in Section V, requires the coordination of a set of agents to complete a set of non-preemptive tasks in time and space. Furthermore, a scheduling decision by one agent affects the utility of the other agents' scheduling decisions. This type of problem arises in real-world settings, such as manufacturing. We develop scalable methods for scheduling resources with temporospatial constraints. The key to this work is the formulation of an analytical test, which leverages problem structure, to ensure that each scheduling commitment will not result in schedule infeasibility [36].

Second, we need to demonstrate that an autonomous scheduling algorithm can improve resource utilization and gain the appreciation of the human workers whose work will be controlled by this autonomous algorithm. In Section VI, we describe a human-subject experiment where subjects work on a team of three: the subject, a human teammate, and a robotic teammate. The robot uses our autonomous scheduling algorithm from Section V to coordinate team activities. We study the effect of decision-making authority (i.e., which team members have control over which scheduling decisions) on team performance and the satisfaction of the human team members. We demonstrate that our scheduling techniques can both improve team performance and gain the support of the human team members by providing fully-autonomous scheduling support [34].

Third, we need to develop a framework to autonomously learn scheduling and resource optimization heuristics from expert demonstrators, which can be embedded within scalable, autonomous methods for scheduling those resources. This work is challenging for several key reasons. First, the amount of data we will be able to collect is small relative to the size of the state space. Passive learning may not be enough to accurately encode the wealth of knowledge domain experts employ in resource optimization decisions. Thus, it may be necessary to perform active learning, where the autonomous algorithm carefully crafts scenarios to reduce model uncertainty. This is a common challenge in many algorithms that learn from demonstration [19], [20]. Second, human experts often use satisficing strategies based upon prior experience. These demonstrations may be near-optimal, but human experts do not perform an exhaustive search to find the true optimal solution [56]. Third, human experts may reasonably disagree with one another about the correct way to allocate resources. These disagreements will require intelligently separating the data to learn accurate models for resource optimization. Generating a separate model for each demonstrator will make learning in a high-dimensional space much more difficult, so we need to identify areas where data across operators can be combined.

The potential contributions of this thesis thus include 1) scalable models for resource optimization, 2) the ability to autonomously learn varying scheduling heuristics from expert demonstrators, and 3) the ability to incorporate these heuristics within an autonomous optimization framework.

IV. Literature Review

Our work in resource optimization via learning expert heuristics from demonstration will draw upon work in multi-agent planning and scheduling, data-driven model learning, human in-the-loop decision making, and recommender systems. We now review prior work in these areas.

A. Multi-Agent Planning and Scheduling

Korsah, Stenz, and Dias provide a comprehensive taxonomy [58] for the multi-robot task allocation and scheduling problem, but the same taxonomy holds for scheduling with human or other resources. The problem of resource management in labor and delivery consists of nurses who can manage multiple patients at once (multi-task agents [MT]) and patients whose care requires one nurse and a room depending on the patient's needs (multi-agent tasks [SR]). Nurses must also consider the future actions required to ensure a high level of care (time-extended allocation [TA]). Lastly, a resource nurse can decompose the higher level task of caring for a patient into one of many possible formulations (Complex Dependencies, CD). Consider a scenario where the labor floor is full of patients and there is a patient in the waiting room who will likely deliver imminently. The resource nurse may choose to move a patient who is not in active labor from the labor floor to an antepartum bed until more room is available so that the active labor patient can be delivered. At a later point, when the floor is less full, the resource nurse might bring the antepartum patient back to the labor floor. This example is but one of many decompositions that are defined with the assignment of resources ${}^{t}R^{a}_{\tau_i^j}$ in our definition in Section II. Thus, the problem fits within the XD [MT-MR-TA] category, which is the most challenging of all problems described in the Korsah-Stenz-Dias taxonomy [58].

We note that it is specifically the preemptability of tasks that allows for task decomposition. The path a patient takes through the hospital within a specific stage of labor, vís a vís the resource assignment variables ${}^{t}R^{a}_{\tau_i^j}$, changes how the patient's labor is managed. In turn, the probability distribution from which comes the duration of each stage τ_i^j in the labor progress for patient τ_i changes based on the history of the resource assignments. For example, a patient may receive oxytocin on the labor floor, which tends to expedite labor. If the labor floor is full and the patient is progressing slowly, the patient may be moved to the antepartum ward and taken off of oxytocin. The patient would return to the labor floor at a more opportune time. If the labor floor had not been full, the patient might have remained on the labor floor despite her slow progress.

As reported by Korsah, Stenz, and Dias, there is no well-known, general mathematical definition in the combinatorial optimization literature for problems within the XD [MT-MR-TA] category class. However, there is some work in

application-specific solution techniques. Jones *et al.* consider a disaster response scenario where a set of fires, which must be extinguished by firefighters. Some routes to the fires are blocked by rubble, which may be cleared by bulldozers. Thus, one must decide which roads are to be cleared and, then, how to optimally plan a schedule for the firefighters to extinguish the fires [51]. To solve this problem, Jones *et al.* present both a multi-tiered auction approach and a genetic algorithm. This problem is slightly easier in that each agent can only perform one task, and each task only requires one agent.

Another auction-based approach, called TraderBots, is proposed by Dias [27]. In this work, agents can bid on *task trees*, or possible task decompositions, rather than only on simple, fully decomposed tasks. Agents may also bid on decompositions suggested by other agents. The agents, or bots, then negotiate, form centralized subgroups, and self-organize to select an assignment of agents to tasks and decompositions of those tasks to reduce the plan cost. For problems with tasks requiring multiple agents, Tang and Parker present an approach for coalition formation where only the immediate assignment of agents to tasks is required (i.e., not a time-extended schedule). Tang and Parker present a greedy, centralized approach, ASyMTRe, where each robot uses network *schemas* to solve the coordination problem [98]. These schemas consist of a list of input and output variables and an associated behavior that maps inputs to outputs. ASyMTRe then greedily connects the input and outputs across the robots to form coalitions which can act together to complete the task [98].

The approaches presented here commonly employ greedy techniques to solve tasks with complex dependencies. These approaches require the algorithms' architects to solicit domain knowledge to cleverly simplify and craft heuristics to generate a solution. These heuristic approaches lend support to the need to autonomously learn policies developed by domain experts and encode this knowledge directly within the optimization framework.

B. Data-Driven Model Learning

Machine learning is an active area of study in both solving optimization problems and learning policies from demonstration. One common approach to learning from observation that has been quite successful is Inverse Reinforcement Learning. The problem of inverse reinforcement learning (IRL) is to take as input 1) a Markov Decision Process without a known Reward Function R and 2) a set of m expert demonstrations $O = \{(s_o, a_o), (s_1, a_1), \ldots, (s_m, a_m)\}$ and then determine a reward function R that explains the expert demonstrations. The computational bottleneck of IRL and dynamic programming, in general, is the size of the state-space. The algorithms that solve these problems typically work by iteratively updating the estimate of the future expected reward of each state until convergence. For many problems of interest, the number of states is too numerous to hold in the memory of modern computers, and the time required for the expected future reward to converge can be impractical.

Researchers have extended the capability of IRL algorithms to be able to learn from operators with differing skill levels [80] and identify operator subgoals [70]. IRL is able to leverage the structure of the Markov-Decision Process to bind the rationality of the agent. Other researchers have investigated how robots can learn from demonstration via reinforcement learning where the operator provides real-time feedback for the reward an agent should receive at each location in the state space [72], [100]. Nikolaidis and Shah use reinforcement learning and an approach, called *cross-training*, in which the human and robot switch roles on a team to improve the rate of learning [72]. Thomaz and Breazeal have shown how humans teach robots by providing feedback as a reward signal not just for the current state of the robot but also for anticipated future states [100]. Thomaz and Breazeal extend a reinforcement learning framework to handle the anticipatory feedback from human instructors [100]. However, resource optimization or scheduling is highly non-Markovian: the next state of the environment is dependent upon the history of actions taken to arrive at the current state and the current time. Some researchers have tried to extend the traditional Markov Decision Process to characterize temporal phenomena, but these techniques do not scale up efficiently [11], [25], [107].

Another main effort in machine learning has been to learn a function which maps states to actions. Chernova and Veloso develop a Gaussian Mixture Model, which is able to interactively learn from demonstration [19]. Their algorithm first learns a reasonable policy for a given task (e.g., driving a car on a highway). Next, the algorithm solicits user feedback by constructing scenarios where there is a high level of uncertainty. Chernova and Veloso also explore using support vector machines to learn when an autonomous agent should request more demonstrations [20].

Sauppé and Mutlu study the problem of learning a model for the human behavior of back-channeling, which is a form of communication by an addressee to facilitate turn-taking and acknowledge speakership [99]. They demonstrate that a regression-based approach can predict the exhibition of these behaviors [99]. In more recent work, Huang and Mutlu study how humans employ multi-modal communication behaviors to present information in the form of a story [46]. They note that previous attempts at modeling typically employ a laborious process of hand-crafting rules and heuristics that lack generality [46]. Huang and Mutlu develop a robotic platform that uses a dynamic Bayesian network to learn how people choreograph their speech, gaze, and gestures for narration [46].

Ramanujam and Balakrishnan investigate the problem of learning a discrete-choice model for how air traffic controllers decide which set of runways to use for arriving and departing aircraft based on weather, arrival and departure demand, and other environmental factors [81]. Ramanujam and Balakrishnan train a discrete-choice model on real data from air traffic controllers and show how the model can accurately predict the correct runway configuration for the airport [81].

Sammut *et al.* apply a decision tree model for learning

an autopilot to control an aircraft from expert demonstration [88]. Their approach generates a separate decision tree for each of the following control inputs: elevators, ailerons, flaps, and thrust. In their investigation, they note that each pilot demonstrator could execute a planned flight path differently. These demonstrations could be in disagreement, thus making the learning problem significantly more difficult. To cope with the variance between pilot executions, a separate model was learned for each pilot. Inamura *et al.* use a Bayesian Network to learn which behavior to use and when to use that behavior via demonstration and interaction with that demonstrator [48]. Saunders *et al.* use a k-Nearest Neighbors approach with feature selection to learn which features to use in each scenario to predict the appropriate action [90]. In [86], Rybski *et al.* employ a Hidden Markov Model to learn which of a pre-learned set of actions must be applied and in what order to complete a demonstrated box-sorting task.

Data-driven model learning techniques offer much promise for learning policies from expert demonstrators. However, many approaches, such as dynamic programming, require the enumeration of a large state-space, leading to computational intractability, and do not naturally capture the temporal structure of scheduling problems. Approaches that use regression or classification-based methods for learning policies from expert demonstrators have promise, but they typically have not been applied to complex scheduling problems. For these methods to succeed, the correct modeling of the scheduling problem is required. In Section VII, we describe how we model the scheduling problem to take advantage of data-driven model learning techniques.

C. Human In-The-Loop Decision Making

The field of human factors has pursued complementary efforts, which focus on developing interfaces between human supervisors and the agents they are tasking [8], [30], [68], [87], [104]. The human-robot interface has long been identified as a major bottleneck for the utilization of these robotic systems to their full potential [17]. As a result, significant research efforts have been aimed at easing the use of these systems in the field, including the careful design and validation of supervisory and control interfaces [4], [24], [37], [52], [45].

Many researchers have focused on the inclusion of a human in the decision-making loop to improve the quality of task plans and schedules for robots or semi-autonomous systems [22], [24], [28]. This is particularly important if the human operators have knowledge of factors not explicitly captured by the system model or if scheduling decisions have life or death consequences. In a study of aircraft carrier flight deck operations, veteran operators used heuristics to quickly generate an efficient plan and outperformed optimization algorithms [85]. Other works aimed to leverage the strengths of both humans and machines in scheduling by soliciting user input in the form of quantitative, qualitative, hard, or soft constraints over various scheduling options. Recommended schedules were then autonomously compiled and provided to users [3], [39], [41], [67], [108]. Researchers also develop models for how people weigh exploit-versus-explore actions when seeking reward in an uncertain environment [82]. Within the medical domain, Szolovits *et al* describe work in developing algorithms that mimic the reasoning of human domain experts [96] rather than explicitly codifying rules [97]. These attempts to model reasoning are better able to isolate a key set of possible diagnoses, utilize pathophysiologic reasoning, and model the complexities of the illnesses of specific patients [96].

Supervisory systems have also been developed to assist human operators in the coordination of the activities of either four-robot or eight-robot teams [18]. Experiments demonstrated that operators were less able to detect key surveillance targets when controlling a larger number of robots. Similarly, other studies have investigated the perceived workload and performance of subjects operating multiple ground mobile-based robots [1]. Findings indicated that a number of robots greater than two greatly increased the perceived workload and decreased the performance of the human subjects.

Researchers in scheduling have also tried to create an autonomous scheduling assistant that can learn the preferences of the user [6]. However, the authors of this work report that the scheduling preferences of users are hard to learn, and these autonomous scheduling assistants have thus far not shown to be helpful enough to employ in practice. Another example is work by De Grano *et al.*, who present a method to optimize scheduling shifts for nurses by soliciting nurses' preferences through an auction process [38]. Other work in scheduling preferences has focused on developing techniques to efficiently generate schedules once the preferences have been solicited [94], [106].

One key area of focus is in modeling preferences as a set of *ceteris paribus* (all other things being equal) preference statements [9], [10], [73]. In this work, researchers solicit preferences from users typically in the form of binary comparisons. For example, consider the problem of determining which food and drink to serve a guest [9]. You may know the following:

- The guest prefers to drink red over white wine when eating a steak.
- The guest prefers steak over chicken.
- The guest prefers white wine when eating chicken.

Interestingly, determining the optimal food-drinking pairing can be found in polynomial-time; however, determining the relative optimality of one pairing over another (known as dominance testing) is NP-complete [9].

D. Recommender Systems

Finally, the study of recommender systems is an important area of consideration. Recommender systems have become ubiquitous in the Internet age with services such as Netflix [57]. Recommender systems generally fall into one of two categories: collaborative filtering (CF) and content-based filtering (CB) [77]. In essence, collaborative filtering is a technique in which an algorithm learns to predict content to a single user based upon that user's history and that of other

users who share common interests [77]. However, CF suffers from sparsity and scalability problems [77]. Content-based filtering works by comparing the similarity between content that the user has previously viewed and new content [23], [43], [89]. The challenge of content-based filtering lies in the difficulty in measuring the similarities between two items, and these systems can often over-fit, only predicting content that is very similar to what the user has previously used [5], [91]. Researchers have employed association rules [21], clustering [63], [64], decision trees [55], k-nearest neighbor algorithms [54], neural networks [2], [47], link analysis [15], regression [69], and general heuristic techniques [77] to recommend content to users.

Ranking the relevance of web pages is a key focus within systems that recommend suggested topics to users [16], [40], [42], [49], [74], [75], [62], [101], [103]. The seminal paper on page ranking, by Page *et al.*, started the computational study of web page ranking with an algorithm, PageRank, which assesses the relevance of a web page by determining the number of pages that link to the page in question [74]. Since this paper, many have focused on developing better models for recommending web pages to users, which can then be trained using various machine learning algorithms [40], [42], [49], [75]. Within this discipline, there are three primary approaches to modeling the importance of a web page: point-wise, pair-wise, and list-wise ranking. In point-wise ranking, the goal is to determine a score, via regression analysis, for a web page given features describing its contents [62], [74]. Pair-wise ranking is typically a classification problem where the aim is to predict whether one page is more important than another [49], [75]. More recent efforts have focused on list-wise ranking, where researchers develop loss-functions based on entire lists of ranked web pages rather than individual pages or pair-wise comparisons between pages [16], [101], [103]. As we discuss in Section VII, our prototype for apprenticeship learning of heuristics for resource optimization is directly inspired by work in the area of machine learning for web page ranking. Intuitively, the problem of determining the next job to schedule or the best next action to take in a resource optimization problem is similar to predicting the most relevant web page in a query. Instead of ranking web pages, one is now ranking possible scheduling actions.

V. Work to Date: Fast Methods for Human-Robot Team Coordination

In our quest to develop an autonomous capability to learn expert heuristics and embed those rules of thumb within a dynamic resource optimization framework, we first explore an area where we have particular domain expertise: assembly of commercial airplanes. We leverage this domain knowledge to develop efficient, near-optimal computational methods for coordinating resources [36].

Consider the example of the Boeing Fully Autonomous Upright Build (FAUB) for assembly of a Boeing 777, as shown in Figure 1. A set of robots must work together to complete a set of riveting tasks. These robots must adapt their schedules in response to part shortages or delays, robotic equipment failure, and human workers who need to enter the space to perform work on the fuselage or quality assurance inspections. These robots typically can perform one task at a time, and the tasks are non-preemptable.

Fig. 1: The Boeing Fully Autonomous Upright Build for the Boeing 777. Image courtesy of Boeing Research and Technology.

A. Problem Formulation

We can readily formulate this problem as a mixed-integer linear program (MILP), as shown in Equations 11-24.

$$\min z, z = g\left(\{A^a_{\tau_i^j} | \tau_i \in \boldsymbol{\tau}, a \in A\}, \{J_{\langle \tau_i^j, \tau_x^y \rangle} | \tau_i^j, \tau_x^y \in \boldsymbol{\tau}\}, \{s_{\tau_i^j}, f_{\tau_i^j} | \tau_i^j \in \boldsymbol{\tau}\}\right) \quad (11)$$

subject to

$$\sum_{a \in A} A^a_{\tau_i^j} = 1, \forall \tau_i^j \in \boldsymbol{\tau} \quad (12)$$

$$ub_{\tau_i^j} \geq f_{\tau_i^j} - s_{\tau_i^j} \geq lb_{\tau_i^j}, \forall \tau_i^j \in \boldsymbol{\tau} \quad (13)$$

$$f_{\tau_i^j} - s_{\tau_i^j} \geq lb^a_{\tau_i^j} - M\left(1 - A^a_{\tau_i^j}\right), \forall \tau_i^j \in \boldsymbol{\tau}, a \in A \quad (14)$$

$$s_{\tau_x^y} - f_{\tau_i^j} \geq W_{\langle \tau_i, \tau_j \rangle}, \forall \tau_i, \tau_j \in \boldsymbol{\tau} |, \forall W_{\langle \tau_i, \tau_j \rangle} \in \boldsymbol{TC} \quad (15)$$

$$f_{\tau_x^y} - s_{\tau_i^j} \leq D^{rel}_{\langle \tau_i, \tau_j \rangle}, \forall \tau_i, \tau_j \in \boldsymbol{\tau} | \exists D^{rel}_{\langle \tau_i, \tau_j \rangle} \in \boldsymbol{TC} \quad (16)$$

$$f_{\tau_i^j} \leq D^{abs}_{\tau_i}, \forall \tau_i \in \boldsymbol{\tau} | \exists D^{abs}_{\tau_i} \in \boldsymbol{TC} \quad (17)$$

$$s_{\tau_x^y} - f_{\tau_i^j} \geq M\left(A^a_{\tau_i^j} + A^a_{\tau_x^y} - 2\right) + M\left(J_{\langle \tau_i^j, \tau_x^y \rangle} - 1\right), \forall \tau_i, \tau_j \in \boldsymbol{\tau}, \forall a \in A \quad (18)$$

$$s_{\tau_i^j} - f_{\tau_x^y} \geq M\left(A^a_{\tau_i^j} + A^a_{\tau_x^y} - 2\right) - M\left(J_{\langle \tau_i^j, \tau_x^y \rangle}\right), \forall \tau_i, \tau_j \in \boldsymbol{\tau}, \forall a \in A \quad (19)$$

$$s_j - f_{\tau_i^j} \geq M\left(J_{\langle \tau_i^j, \tau_x^y \rangle} - 1\right), \forall \tau_i, \tau_j \in \boldsymbol{\tau} | R_{\tau_i} = R_{\tau_j} \quad (20)$$

$$s_{\tau_i^j} - f_j \geq -M\left(J_{\langle \tau_i^j, \tau_x^y \rangle}\right), \forall \tau_i, \tau_j \in \boldsymbol{\tau} | R_{\tau_i} = R_{\tau_j} \quad (21)$$

$$J_{\langle \tau_i^j, \tau_x^y \rangle} = 1, \forall \tau_i^j, \tau_x^y \in \boldsymbol{\tau} | \exists W_{\langle \tau_i^j, \tau_x^y \rangle} \in \boldsymbol{TC} \quad (22)$$

$$J_{\langle \tau_x^y, \tau_i^j \rangle} = 0, \forall \tau_i^j, \tau_x^y \in \boldsymbol{\tau} | \exists W_{\langle \tau_i, \tau_j \rangle} \in \boldsymbol{TC} \quad (23)$$

$$J_{\langle \tau_a^b, \tau_x^y \rangle} - J_{\langle \tau_a^b, \tau_i^j \rangle} - J_{\langle \tau_i^j, \tau_x^y \rangle} + 1 \geq 0, \forall \tau_i, \tau_j, \tau_k \in \boldsymbol{\tau} \quad (24)$$

In this formulation, $A^a_{\tau_i^j} \in \{0,1\}$ is a binary decision variable for the assignment of agent a to task τ_i, $J_{\langle \tau_i^j, \tau_x^y \rangle} \in \{0,1\}$ is a binary decision variable specifying whether τ_i comes after or before τ_j, and $s_{\tau_i^j}, f_{\tau_i^j} \in [0, \infty)$ are the start and finish times of τ_i. Equation 11 is a general objective that is a function of the decision variables $\{A^a_{\tau_i^j} | \tau_i^j \in \boldsymbol{\tau}, a \in A\}$, $\{J_{\langle \tau_i^j, \tau_x^y \rangle} | \tau_i, \tau_j \in \boldsymbol{\tau}\}$, and $\{s_{\tau_i^j}, f_{\tau_i^j} | \tau_i^j \in \boldsymbol{\tau}\}$. Equation 12 ensures that each task is assigned to a single agent. Equation 13 ensures that the duration of each $\tau_i \in \boldsymbol{\tau}$ does not exceed its upper and lowerbound durations. Equation 14 requires that the duration of task τ_i^j, $f_{\tau_i^j} - s_{\tau_i^j}$, is no less than the time required for agent a to complete task τ_i. Equation 15 requires that τ_x^y occurs at least $W_{\langle \tau_i^j, \tau_x^y \rangle}$ units of time after τ_i^j. Equation 16 requires that the duration between the start of τ_i^j and the finish of τ_x^y is less than $D^{rel}_{\langle \tau_i^j, \tau_x^y \rangle}$. Equation 17 requires that τ_i^j finishes before $D^{abs}_{\tau_i^j}$ units of time has expired since the start of the schedule. Equations 18-19 enforce that agents can only execute one task at a time. Equations 20-21 enforce that each resource R_i can only be accessed one agent at a time.

Equations 22-24 are additional constraints to reduce the computational search space. Equations 22 and 23 leverage each wait constraint $W_{\langle \tau_i^j, \tau_x^y \rangle}$ to know that τ_i^j must precede τ_x^y (i.e., $J_{\langle \tau_i^j, \tau_x^y \rangle} = 1$ and $J_{\langle \tau_x^y, \tau_i^j \rangle} = 0$). Equation 24 leverages a triangle inequality: if τ_a^b precedes τ_i^j (i.e., $J_{\langle \tau_a^b, \tau_i^j \rangle} = 1$) and τ_i^j precedes τ_x^y (i.e., $J_{\langle \tau_i^j, \tau_x^y \rangle} = 1$), then τ_a^b must precede τ_x^y. Thus, we know that $J_{\langle \tau_a^b, \tau_x^y \rangle}$ must equal 1.

The worst-case time complexity of a complete solution technique for this problem is dominated by the binary decision variables for allocating tasks to agents ($A^a_{\tau_i^j}$) and sequencing ($J_{\langle \tau_i^j, \tau_x^y \rangle}$), and the complexity is given by $O\left(2^{|A||\boldsymbol{\tau}|^3}\right)$, where $|A|$ is the number of agents and $|\boldsymbol{\tau}|$ is the number of tasks. Agent allocation contributes $O\left(2^{|A||\boldsymbol{\tau}|}\right)$, and sequencing contributes $O\left(2^{|\boldsymbol{\tau}|^2}\right)$.

This problem formulation consists of non-preemptive tasks. In our motivating example in Section II, we subdivided agent assignment variables for each subtask τ_i^j into discrete epochs to allow for τ_i^j's agent assignment to be preempted at any time t. In this formulation, each agent can only process one subtask at a time, which we can enforce by introducing linear sequencing constraints (Equations 18-19) and binary sequencing variable $J_{\langle \tau_i^j, \tau_x^y \rangle}$. While we introduce $|\boldsymbol{\tau}|^2$ binary decision variables $J_{\langle \tau_i^j, \tau_x^y \rangle}$ in this more restricted problem, we actually eliminate $(T-1) * |A| * |\boldsymbol{\tau}|$ agent assignment variables. This problem has one more simplification in that resource assignments are fixed, which reduces the search space by a factor of $T * |R| * |\boldsymbol{\tau}|$.

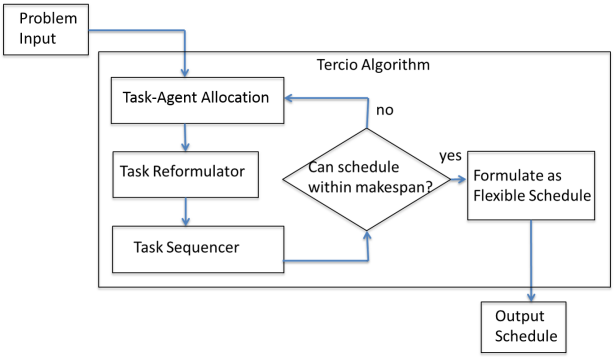


Fig. 2: Tercio receives a task assignment and scheduling problem as input and returns a flexible temporal plan, if one can be found. The input problem is decomposed into a task allocation and a task sequencing problem. The algorithm works by iterating through agent allocations until a schedule can be found that satisfies the maximum allowable makespan for the problem.

B. Tercio: Overview

We describe our approach, which we call Tercio, to solving this scheduling problem. Tercio's algorithmic architecture is shown in Figure 2. Tercio takes as input a set of tasks, temporal interval constraints, agents, and an objective function. The algorithm first computes an optimal agent allocation by solving a mixed-integer program, which includes terms for balancing the amount of work across each agent. Next, Tercio reformulates the problem into a more restricted structure, which we describe in Section V-D. Given the agent allocation and a more restricted task structure, Tercio sequences the tasks using an analytical test to ensure that all temporal constraints are satisfied. If the schedule is not within a user-specified makespan (i.e., overall process duration), Tercio attempts to find the next-best agent allocation, excluding all previously-tested allocations. If, however, the schedule does satisfy the user-specified makespan, then agent and spatial-resource sequencing constraints (interval form of $[0, \infty)$) are added to the problem. The resulting Simple Temporal Problem (STP), composed of the interval temporal constraints, is compiled into a dispatchable form [71], [105], which guarantees that for any consistent choice of a time point within a flexible window, there exists a solution that can be found through one-step propagation of interval bounds. The dispatchable form maintains flexibility to increase robustness to disturbances, and has been shown to decrease the amount of time spent re-computing solutions in response to disturbances for randomly generated structured problems by up to 75% [105].

C. Tercio: Agent Allocation

The Tercio agent allocation subroutine performs agent-task allocation by solving an MILP that includes Equations 12 and 13 ensuring each task is assigned to exactly one agent and that the agent-task allocation does not violate the upper and lowerbound temporal constraints. The objective function

is formulated as shown in Equation 25, and is application-specific.

$$\min(z),\ z = g_1(A, {}^{P}A, \boldsymbol{\tau}) + (g_2(A) - g_3(A)) \tag{25}$$

For the purposes of this work, we use a g that includes three terms. The first term minimizes $g_1(A, {}^{P}A, \boldsymbol{\tau})$, the number of differences between the previous ${}^{P}A^{a}_{\tau_i^j}$ and the new agent assignment $A^{a}_{\tau_i^j}$. Minimizing this quantity helps to avoid oscillation among solutions with equivalent quality during rescheduling.

$$g_1(A, {}^{P}A^{a}_{\tau_i^j}) = \sum_{\tau_i \in \boldsymbol{\tau}} \left(\left(\sum_{a \in A | {}^{P}A^{a}_{\tau_i^j} = 1} 1 - A^{a}_{\tau_i^j} \right) + \left(\sum_{a \in A | {}^{P}A^{a}_{\tau_i^j} = 0} A^{a}_{\tau_i^j} \right) \right) \tag{26}$$

The second and third terms, g_2 and g_3, perform work balancing by minimizing the maximum work assigned to any one agent (Equation 27) and by maximizing the minimum work assigned to any one agent (Equation 28). We find in practice that the addition of these term guides the optimization toward agent allocations that yield a low makespan.

$$g_2(A) \geq \sum_{\tau_i \in \boldsymbol{\tau}} A^{a}_{\tau_i^j} \times lb^{a}_{i}, \forall a \in \boldsymbol{A} \tag{27}$$

$$g_3(A) \leq \sum_{\tau_i \in \boldsymbol{\tau}} A^{a}_{\tau_i^j} \times lb^{a}_{i}, \forall a \in \boldsymbol{A} \tag{28}$$

Note that by decomposing task allocation from sequencing, Tercio cannot directly minimize the makespan. The functions $g_2(A, \boldsymbol{\tau})$ and $g_3(A, \boldsymbol{\tau})$ are used as a heuristic for guiding Tercio toward solutions that minimize the makespan. To establish a basis for comparison in the empirical evaluation, we solve the MILP defined in Section V-A using a complete solution technique, and instead replace $g_2(A, \boldsymbol{\tau})$ and $g_3(A, \boldsymbol{\tau})$ with $g_{2,3-MILP}(s, f)$ as shown in Equation 29.

$$g_{2,3-MILP}(s, f) = \max_{\langle \tau_i^j, \tau_x^y \rangle} \left(f_i^j - s_x^y \right). \tag{29}$$

D. Multi-Agent Task Sequencer

The Tercio sequencing subroutine works by scheduling through simulation. At each time epoch t, the sequencer iterates over the set of agents currently idle. For each idle agent, the sequencer determines highest priority subtask τ_i^j to schedule next according to a set of dynamic priority heuristics. However, in the presence of resource constraints and deadline constraints, it may not be feasible to schedule τ_i^j at time t. To ensure correctness, the sequencing subroutine uses our analytical schedulability test to determine whether scheduling subtask τ_i^j at time t might result in a violated constraint later in the schedule. If the schedulability test determines that τ_i^j might result in such a constraint violation, the agent determines the next-highest-priority subtask and repeats the test until the agent either schedules a subtask or returns a sequence if no subtasks remain.

1) Task Model: While Tercio operates on a general task network, as described by the MILP in section V-A, we restrict the structure of task network. With this structure, we are able to formulate a schedulability test to ensure schedule feasibility, which we describe in Section V-D.3. For brevity, we omit the details for how we reformulate a general task network into our more restricted form. Instead, we refer the reader to [33].

The basis for our framework is the self-suspending task model [61], as described in Equation 30.

$$\tau_i : \left(\left(C_i^1, E_i^1, C_i^2, E_i^2, \ldots, E_i^{m_{i-1}}, C_i^{m_i} \right), T_i, D_i \right) \tag{30}$$

In this model, there is a task set $\boldsymbol{\tau} = \{\tau_i | i \in \{1, 2, \ldots, n\}\}$ contains n tasks τ_i that must be processed by a set of computer processors. Each task has m_i subtasks with $m_i - 1$ self-suspension intervals. C_i^j is the expected duration (cost) of τ_i^j, and E_i^j is the expected duration of the j^{th} self-suspension interval of τ_i. T_i and D_i are the period and deadline of τ_i, respectively.

While this self-suspending task model provides a solid basis for describing many real-world processor scheduling problems of interest, we augment this model to better capture problem structure inherent in the manufacturing environment. First, we set the period T_i and deadline D_i of each task τ_i equal to a constant, T (i.e., $T_i = D_i = T, \forall i$). This modification models many assembly line manufacturing processes where the set of tasks at one location is repeated once every "pulse" of the production line. Second, we allow for phase offsets ϕ for each task τ_i, where a phase offset is a delay between the epoch time and the release of the given task. Third, we allow for intra-task and subtask deadlines. An intra-task deadline $D^{rel}_{\langle \tau_i^a, \tau_i^b \rangle}$ constrains the start, s_i^a and finish time f_i^b of two subtasks τ_i^a and τ_i^b for a given task τ_i by duration $d^{rel}_{\langle \tau_i^a, \tau_i^b \rangle}$, as shown in Equation 31. A subtask deadline $D^{abs}_{\tau_i^j}$ upperbounds the duration between the epoch and the finish time f_i^j of subtask τ_i^j by duration $D^{abs}_{\tau_i^j}$, as shown in Equation 32.

$$D^{rel}_{\langle \tau_i^a, \tau_i^b \rangle} : \left(f_i^b - s_i^a \leq D^{rel}_{\langle \tau_i^a, \tau_i^b \rangle} \right) \tag{31}$$

$$D^{abs}_{\tau_i^j} : \left(f_i^j \leq D^{abs}_{\tau_i^j} \right) \tag{32}$$

These deadline constraints provide additional expressiveness to encode binary temporal constraints relating tasks in the manufacturing process and are commonly included in AI and OR scheduling models [7], [26], [71], [105].

We also extend the model to include shared memory resources $\boldsymbol{R}$. Each subtask τ_i^j requires that a set of shared memory resources R_i^j be utilized in performing that subtask (e.g., for memory shared among multiple processors), where $\boldsymbol{R}$ is the set of all shared memory resources. In the manufacturing analogy, a resource $r \in R_i^j$ corresponds to a region of space in the factory that must be physically unoccupied for an agent to execute a subtask in that location.

2) Dynamic Priority Heuristics: The Tercio sequencing subroutine employs a set of heuristics to prioritize the order in which subtasks are scheduled. Consider first that we want

as many agents as possible to work concurrently; however, the scheduling of one agent restricts the available subtask options for other agents. As such, we introduce a heuristic function, $\pi_A(\tau_i^j)$, that prioritizes a subtask τ_i^j assigned to agent a in inverse proportion to the number of available subtasks assigned to a. Second, when subtasks share a resource (i.e., are located in close, physical proximity), agents assigned to those tasks may have to idle due to resource contention. We introduce a heuristic function, $\pi_R(\tau_i^j)$, that eases resource contention by prioritizing subtasks that require higher-demand resources over subtasks that require lower-demand resources. Third, consider a scenario with two subtasks, τ_i^j and τ_i^{j+k} with agent assignments a and a'. If a does not execute τ_i^j in a timely manner, then a' will idle. We introduce a heuristic function, $\pi_P(\tau_i^j)$, to ease bottlenecks due to inter-agent precedence constraints. The heuristic prioritizes the execution of a subtask τ_i^j in proportion to the number of following subtasks in task τ_i that are assigned to other agents. Lastly, we utilize the Earliest-Deadline First priority heuristic, $\pi_{EDF}(\tau_i^j)$, which has been shown in the study of real-time systems to have many benefits when scheduling against deadline constraints [95]. Tercio uses a weighted, linear combination of these heuristics. The weighting of these heuristics can be tuned for specific applications.

3) Schedulability Test: The key to increasing the computational speed of Tercio is a novel, analytical schedulability test within the Tercio sequencing subroutine, which allows us to near-optimally sequence tasks against deadline constraints in polynomial time. During the scheduling simulation, we perform an online consistency check that ensures that scheduling of τ_i^j at time t will not cause τ_x^y to violate a temporal or shared memory resource constraint. This work extends our single-agent online consistency test [35] to handle multiple agents and shared memory resource constraints.

This work is inspired by the field of real-time systems research in which dynamic priority heuristics are use to restrict scheduling behavior [60], [79], [84] and analytical schedulability tests are employed to determine whether an additional scheduling commitment can be undertaken [65], [66], [84]. These tests are designed to be polynomial-time, sacrificing completeness, to allow for real-time system performance. To our knowledge, our schedulability test is the first analytical test for non-preemptive, self-suspending tasks where multiple tasks have more than one self-suspension with shared memory resource constraints [60], [61], [83]. [3]

To describe our schedulability test, we introduce a definition for an *active deadline*, which we use to describe our online consistency test.

Definition 1: Active Deadline - Consider an intra-task deadline $D^{rel}_{(i,j),(i,b)}$, or an absolute deadline $D^{abs}_{\tau_i^j}$. An intra-task deadline is considered *active* between $0 \leq t \leq \min\left(f_i^j, D^{rel}_{\langle \tau_i^j, \tau_i^b \rangle}\right)$, and an absolute deadline is considered *active* between $0 \leq t \leq \min\left(f_i^j, D^{abs}_{\tau_i^j}\right)$. $\boldsymbol{D^{active}}$ is the set of all active deadlines.

We readily formulate our online consistency test as a constraint satisfaction problem, as shown in Equations 33-35, where $\boldsymbol{D^*}$ is the union of all active deadlines and the deadline we are considering activating. Equation 33 determines whether a subtask executed by an agent a and using resource r can be scheduled without resulting in the immediate or eventual violation of an active deadline. $\xi_a(i,j,k,a)$ and $\beta_a(i,j,k,a)$ refer respectively the next and last subtask to which agent a is assigned to in $\{\tau_i^j, \ldots, \tau_i^k\}$. $\xi_r(i,j,k,r)$ and $\beta_r(i,j,k,r)$ refer respectively the next and last subtask in $\{\tau_i^j, \ldots, \tau_i^k\}$ that require resource r. $d_i^{\beta_a(i,j,k,a)}$ and $d_i^{\beta_r(i,j,k,r)}$ are the absolute deadlines of the last subtasks assigned to agent a and that require resource r, respectively.

$$\begin{aligned}&\left(\delta^a_{(i,j),(i,k)} \geq d_x^{\beta_a(x,y,z,a)} - t \vee \delta^a_{x,y:z} \geq d_x^{\beta_a(x,y,z,a)} - t\right)\\ \wedge &\left(\delta^r_{(i,j),(i,k)} \geq d_x^{\beta_r(x,y,z,r)} - t \vee \delta^r_{x,y:z} \geq d_x^{\beta_r(x,y,z,r)} - t\right),\\ &\forall D^{rel}_{(i,j),(i,k)} \in \boldsymbol{D^*}, \forall D^{abs}_{(i,k)} \in \boldsymbol{D^*}, \forall a, \forall r\end{aligned} \tag{33}$$

$$\begin{aligned}\delta^a_{(i,j),(i,k)} = d_i^{\beta_a(i,j,k,a)} - t - \Bigg(&C_i^{\beta_a(i,j,k,a)}\\ &+ \sum_{\psi=\xi_a(i,j,k,a)}^{\beta_a(i,j,k,a)-1} \left(C_i^\psi + E_i^\psi\right)\Bigg)\end{aligned} \tag{34}$$

$$\begin{aligned}\delta^r_{(i,j),(i,k)} = d_i^{\beta_r(i,j,k,r)} - t - \Bigg(&C_i^{\beta_r(i,j,k,r)}\\ &+ \sum_{\psi=\xi_r(i,j,k,r)}^{\beta_r(i,j,k,r)-1} \left(C_i^\psi + E_i^\psi\right)\Bigg)\end{aligned} \tag{35}$$

$\delta^a_{(i,j),(i,k)}$ (Equation 34) and $\delta^r_{(i,j),(i,k)}$ (Equation 35) are the *slack time* for agent a and resource r, respectively, in relation to an active deadline $D^{rel}_{(i,j),(i,k)}$ or $D^{abs}_{(i,k)}$. An agent's or resource's slack time for a deadline is a bound on the amount of time that the agent or resource may feasibly commit to the execution of subtasks not associated with that deadline. Temporal feasibility is ensured if, for each deadline, we can nest the time before one deadline within the slack of another (or vice versa) for all agents and resources associated with that deadline. Our multi-agent sequencer uses a polynomial-time version of this online consistency test to evaluate the feasibility of scheduling subtasks. The complexity of this consistency check is $O(n(a+r))$ where n is the number of tasks, a is the number of agents, and r is the number of resources. We refer the reader to [33] for further details.

E. Evaluation and Discussion

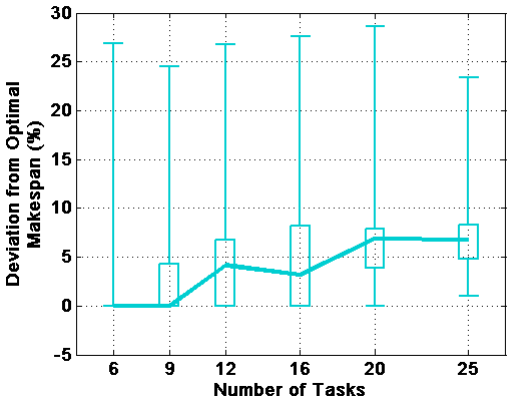
In this section, we empirically validate the computational speed and optimality of Tercio for the multi-agent task assignment and scheduling problem with temporospatial constraints. We first compare the solution quality and computational speed of Tercio to the full MILP formulation. We generate a set of random problems, which we solve

[3] Our online consistency test is similar in spirit to resource edge-finding [59], [102].

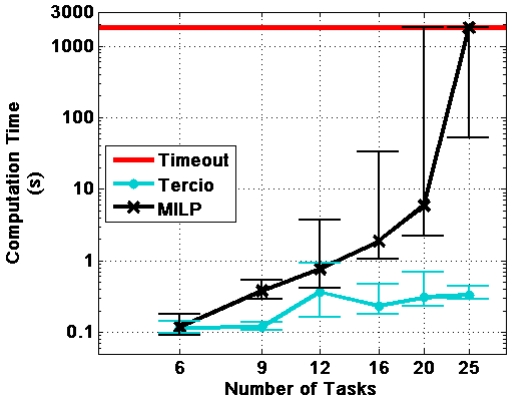
optimally. We then run Tercio where we set the makespan cutoff to be within 10% of the optimal solution. We also set Tercio to return the best solution after 5 iterations if a solution within 10% of the optimal solution has not been found. Figure 3a shows that we are able to generate solutions within 10% of the optimal makespan, on average. Figure 3b shows that we are able to find these solutions in less than one second for up to 25 subtasks. Because the optimization tool, Gurobi, we used to solve for the exact solution is unable to test the optimality of Tercio for problem sizes above 25 tasks, we compare the first solution returned by Tercio to a theoretical lowerbound on the makespan (Equation 36). As shown in Figure 3, the optimal makespan returned by Tercio empirically approaches this theoretical lowerbound.

$$makespan \geq \frac{1}{|A|} \sum_{\tau_i^j \in \boldsymbol{\tau}} C_i^j \tag{36}$$

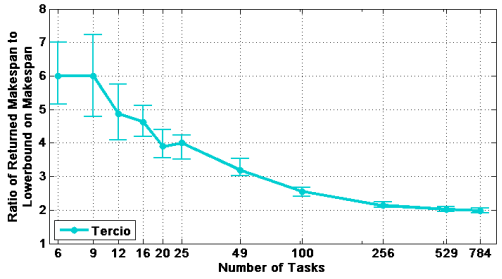
We demonstrate the use of Tercio to plan work performed by two KUKA Youbots, as shown in Figure 4a. Video of the demonstration is available at http://tiny.cc/t6wjxw. In this demonstration, two robots were working to assemble a mock airplane fuselage. The robots performed their subtasks at specific locations on the factory floor. To prevent collisions, each robot reserved both the physical location for its subtask, as well as the immediately adjacent subtask locations. Initially, the robots planned to evenly split 12 identical tasks down the middle of the fuselage. After the robots finished their first subtasks, a person then requested time to inspect the work completed on the left half of the fuselage. In the problem formulation, this corresponds to adding a resource reservation for the left half of the fuselage for a specified period of time. Tercio re-planned in response to the addition of this new constraint, and reallocated the work between the robots in a reasonable manner to make productive use of both robots and to keep the number of interfaces low. We also demonstrate Tercio using a simulated problem of larger size, as shown in Figure 4b. In this demonstration, five robots coordinated to perform 110 tasks around a large cylindrical structure. Tercio was applied to re-plan in response to a request for a person to enter the space, in response to a robot breakdown, and in response to changing task deadlines. Video of the demonstration is available at http://tinyurl.com/m5bsx6g.



(a) Boxplot showing Tercio's deviation from the optimal makespan. Whisker bars on the box plots represent the extreme values



(b) The median and quartiles of Tercio's computational speed when minimizing the makespan.

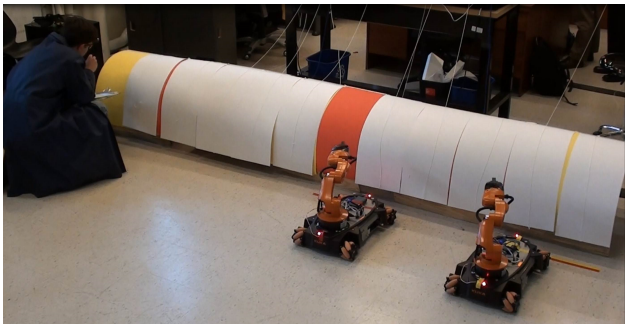


(c) The median and quartiles of Tercio's deviation from a theoretical lowerbound on the makespan.

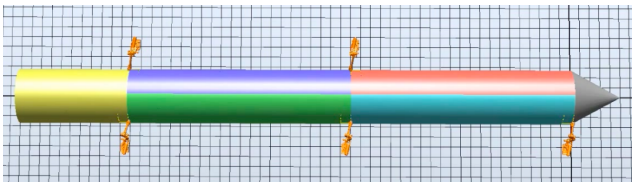
Fig. 3: Computational speed and solution quality are shown for problems with five agents.

VI. Work to Date: Integrating Autonomous Scheduling into Human-Agent-Robot Teams

While Tercio provides the technical capability of performing resource optimization for human-robot teaming in manufacturing, it is crucial that we demonstrate that our autonomous scheduling algorithm improves resource optimization *and* garners the support of the end-users. For example, human workers often develop a sense of identity and security from their roles or jobs in a factory, and many are used to having some degree of autonomy in decision-making. As a result, a human worker who is tasked by an automated scheduling algorithm may feel devalued. Even if the algorithm increases process efficiency at first, taking control away from human workers may alienate them and, in turn, ultimately damage overall productivity. On the other hand, workers may find the process of scheduling to be burdensome, and prefer to be part of an efficient team rather than have a role in the scheduling process, if maintaining such a role decreases their efficiency. While autonomous scheduling algorithms can provide near-optimal schedules within seconds, we also want to determine how much decision-making authority humans should have in the task allocation process, so that they feel appreciated while still maintaining a high level of team efficiency. We conducted



(a) Tercio coordinates a set of robots to allow a human quality assurance agent the time and space necessary to inspect the work.



(b) Tercio is applied to re-plan in response to a request for a person to enter the space, a robot breakdown, and changing task deadlines.

Fig. 4: Demonstrations of Tercio where a team of robots is assigned to tasks involving a mock fuselage and must adapt to dynamic disturbances.

a set of human-subject experiments to determine the right level of autonomy for scheduling algorithm to improve team performance and the satisfaction of the human workers. We hypothesized that team performance would increase with decision-making authority afforded to a robot using Tercio. On the other hand, we also hypothesized that subjects would prefer partial authority over the task process rather than total control, and that having no control is preferable to having complete control.

A. *Experimental Design*

Our human-robot manufacturing team consisted of the human subject, a robotic assistant, and a human assistant. The human subject was capable of both fetching and building, and the robot assistant was only capable of fetching. One of the experimenters played the role of a third teammate for all subjects and was capable of both fetching and building. The third human teammate was included to more realistically represent the composition of a human-robot team in a manufacturing setting. The human subjects either performed the task allocation or shared the decision-making authority over task allocation with the robotic teammate, depending on the experimental condition. The robot assistant was always responsible for sequencing the team's work. The third teammate did not provide any decision-making assistance to the subject or the robot. We used a Willow Garage PR2 platform, as shown in Figure 5, as the robotic assistant for our human-robot team. The robot used Adaptive Monte Carlo Localization (AMCL) [31] and the standard *Gmapping* package in the Robot Operating System (ROS) for navigation.

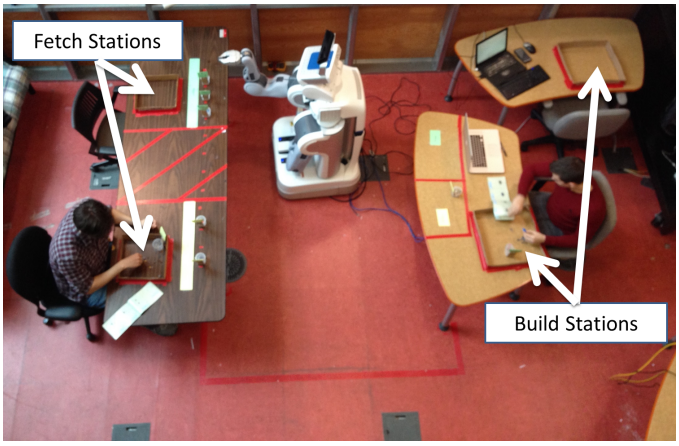


Fig. 5: This figure depicts a diagram of the laboratory room where the experiment took place. There are two locations where the human and robot workers can inspect part kits during a fetching task, and two locations where the human workers can build the part kits.

In our scenario, there are two types of tasks: fetching parts kits and assembling parts kits. Fetching a part kit required walking to one of two inspection stations where the kits were located, inspecting the part kit and carrying it to the build area. The architecture of our fetching task is analogous to what is required in many manufacturing domains: to adhere to strict quality assurance standards, fetching a part kit requires verification from one to two people that all correct parts are in the kit, and certification from another person that the kit has been verified.

There were a number of constraints imposed on the analog assembly process, in order to model relevant constraints encountered during assembly manufacturing: First, a part kit must have been fetched before it could be built. Also, no two agents were able to occupy the same fetching or build station at the same time. There were two fetching and two build stations, as shown in Figure 5. Four part kits were located at one fetching station, and four kits were located at the second fetching station. When fetching a part kit, inspection of that kit must have occurred at the fetching station where it was initially located.

Because there were an equal number of building stations and agents able to build, there were no additional constraints imposed exclusively on build tasks. However, because there were three agents who could fetch and only two fetching stations, the agents were required to take turns using the fetching stations. Allowing workers to sort through parts from multiple kits at the same location risked mixing the wrong part with the wrong kit. We imposed a 10-minute deadline from the time that the fetching of a part kit began until that part kit had been built, for similar reasons. In manufacturing, if a part or part kit is missing from an expected location for too long, work in that area of the factory will temporarily halt until the missing pieces are found.

Assembly of the Lego model involved eight tasks $\boldsymbol{\tau} =$

$\{\tau_1, \tau_2, \ldots, \tau_8\}$, each of which was composed of a *fetch* and *build* subtask $\tau_i = \{\tau_i^{fetch}, \tau_i^{build}\}$. The time each subject took to complete each subtask $C_i^{subject-fetch}$ and $C_i^{subject-build}$ was measured during an experiment training round. The timings for the robot $C_i^{robot-fetch}$ and human assistant $C_i^{assist-fetch}$ and $C_i^{assist-build}$ (performed by an experimenter) were collected prior to the experiments.

To enable the robot to schedule with varying degrees of decision-making input from the subject, we adapted Tercio. In the autonomous condition, the robot (via Tercio) optimized both the allocation and sequencing of subtasks. In the semi-autonomous condition, the subjects chose only the subtasks that they would complete themselves, and the robot allocated the remaining subtasks. In the manual condition, subjects specified the assignment of $agent \in \{subject, robot, assist\}$ to each $subtask \in \{fetch, build\}$ of $\tau_i \in \boldsymbol{\tau}$. In all three experimental conditions (autonomous, semi-autonomous, and manual) the robot sequenced all subtasks $\{J_{<\tau_i^j, \tau_n^m>}\}$. Subjects were provided each agent's expected time to complete each of the sixteen subtasks in the semi-autonomous and manual conditions.

B. Results

Twenty-four participants were included in the experiment. Each participant worked on the human-robot manufacturing team under each level of decision-making authority for a within-subjects design. We consider both objective and subjective measures of the team's performance. Objective measures of team fluency consist of *assembly time* and *rescheduling time*. Assembly time is defined as the difference between the time the last task was completed and the time the first task was initiated. Rescheduling time is defined as the sum of the time it took the subject to allocate tasks when the subject was involved and the time it took the robot to complete the remainder of the scheduling work. Subjective measures consisted of the results of a set of questionnaires administered after each experimental condition to measure the subjects' views on the traits of the robotic teammate and the quality of the alliance amongst the teammates. Subjects also received a post-test questionnaire after completing the three experimental conditions to measure their overall preference for the level of robot autonomy.

We hypothesized that the team would be more fluent, in terms of both assembly and rescheduling time, when the robot has more control authority over task allocation. Rescheduling and assembly times are depicted in Figure 6. Analysis of variance demonstrated statistically significant differences in the distribution of rescheduling time as a function of decision-making authority, $F(2, 69) = 55.1$, $p < 0.01$. Rescheduling time in the autonomous condition ($M = 30$ s, $SD = 0$ s) was lower than in the semi-autonomous condition ($M = 108$ s, $SD = 69$ s), $t(23) = 7.24$, $p < 0.01$. Likewise, rescheduling time in the semi-autonomous condition was lower than in the manual condition ($M = 315$ s, $SD = 154$ s), $t(23) = 7.23$, $p < 0.01$. Repeated-measure analysis of variances demonstrated significant differences in assembly time, as a function of condition $F(2, 46) = 3.84$, $p = .03$. Assembly time in the autonomous condition ($M = 520$ s, $SD = 60.6$ s) was faster than in the semi-autonomous ($M = 564$ s, $SD = 83.9$ s), $t(23) = 2.37$, $p = 0.01$, and manual conditions ($M = 582$ s, $SD = 115$ s), $t(23) = 2.18$, $p = 0.02$.

We also hypothesized that subjects would prefer partial authority over the task process rather than total control, and that having no control is preferable to having complete control. An omnibus Friedman test confirmed a statistically significant difference in the distribution of a subset of the Likert-scale responses for the three conditions. A pair-wise Friedman test confirmed our hypothesis that subjects were more satisfied under the autonomous and semi-autonomous conditions than the manual condition. However, there did not exist a single question for which subjects favored the semi-autonomous condition over the autonomous condition. A post-hoc Friedman test with a requisite Bonferroni correction of $\frac{\alpha}{3}$ indicated that subjects were significantly more satisfied with team performance ($p = 0.008$) under the autonomous condition than the semi-autonomous condition. Likewise, subjects agreed more strongly under the autonomous condition that the team performed the tasks within the least amount of time ($p = .002$). For a list of specific questions, we refer the reader to [34].

The post-test questionnaire included three questions designed to determine whether subjects would be more likely to work with the robot again given the level of decision-making authority allotted to the subject and the robot. We observed a statistically significant difference in subjects' responses to these questions ($p < 0.001$). Post-hoc analysis, using pair-wise Friedman test with a Bonferroni correction, confirmed that subjects agreed that they were more likely to work with the robot again if the robot performed task allocation autonomously than if the subject and the robot shared task allocation authority ($p < 0.001$) or if the subject had complete task allocation authority ($p < 0.01$).Similarly, subjects were more likely to report they would work with the robot again if the robot and the human shared task allocation authority than if the subject had sole authority for task allocation ($p < 0.01$).

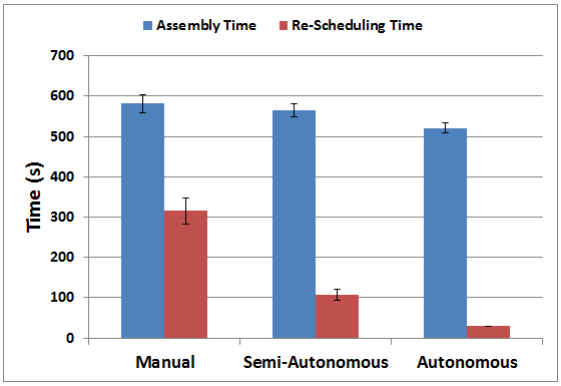


Fig. 6: This figure shows the average and standard error for the assembly times in each condition.

C. Future Work

There are two key areas we wish to pursue in future work. While we know that people prefer working with an autonomous robot, we want to know how that would change if the robot explicitly took into account the preferences of the human team members. We hypothesize that people have attitudes towards certain types of tasks (e.g., a preference for building Lego kits over fetching Lego kits) that may influence their opinion on their desire to work with a robot teammate who is scheduling the team. For example, if a person strongly prefers to build, yet the robot decides that it is best for the team for that person to fetch, the person may resent the robotic teammate. On the other hand, people may dislike idling while the rest of the team works. However, people may be willing to idle to get to work on more-preferred tasks or vice versa. Understanding how a robotic teammate, responsible for scheduling decisions, should incorporate the preferences of human teammates is an interesting area for future study. We plan to run a human-subject experiment varying the way in which the robot schedules the team to discern the impact of human preferences on team fluency.

Second, we also want to give a robotic teammate the ability to teach human teammates how to make better scheduling decisions. Situational awareness is an important contributor to team fluency for human in-the-loop systems [29], [50], [76]. If the robotic system malfunctions, it is important that a human can take up the responsibility of making the scheduling decisions. Even if the autonomous system does not malfunction, there are still domains, such as health care, where a person will retain ultimate control authority for legal and ethical reasons. However, we can envision an autonomous agent as an instructor or advisor for people who are learning to perform resource allocation, such as a nurse who is new to labor and delivery. In this way, resource utilization can be enhanced by leveraging advanced optimization algorithms and human operators can improve and maintain their proficiency. As such, we want to understand how an autonomous instructor for resource optimization could guide people towards better scheduling decisions. We propose a human-subject experiment where a robot reviews a schedule generated by a novice scheduler and offers suggestions to improve the schedule. We hypothesize that people will benefit from a small number of constructive changes but will grow weary of an autonomous instructor that changes many of the decisions by the novice.

VII. Prototype for Learning Heuristics

Based on work in human factors that shows expert decision-makers plan on-the-fly using prior experience [56], we develop a machine learning framework for learning the heuristics and rules-of-thumb via observations. We emphasize that the following formulation is a prototype for how we can learn heuristics for resource optimization from expert demonstrators. This prototype is a strong basis for continuation, as we validate in Section IX, and we expect improvements in future work.

Many approaches to learning models for predicting the best action to take given a current state are based on Markov models, such as reinforcement learning and inverse reinforcement learning [70], [72], [80]. These models, however, do not capture the temporal dependencies between states and are computationally intractable for large problem sizes. To predict the most important subtask, we draw inspiration from the domain of page ranking [74], as discussed in Section IV. The problem of predicting the most relevant web page, based on a search query, is analogous to predicting the highest-priority subtask to be scheduled based on the current resource utilization and task parameters. An 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) [74]. This connectivity is a suitable analogy for the complex temporal dependencies (i.e, precedence, wait, and deadline constraints) relating subtasks in a scheduling problem.

Recent approaches in page ranking have focused on pair-wise and list-wise models, which have been shown to have advantages over point-wise models [101]. In list-wise page ranking, the goal is to generate a ranked list of web pages directly [16], [101], [103], whereas a pair-wise approach determines ranking by performing pair-wise comparisons between individual web-pages [49], [75]. We choose to model the problem of predicting the most important subtask to schedule at time t as a pair-wise, page-rank problem. The pair-wise model has key advantages over the list-wise approach. First, classification algorithms (e.g., support vector machines) can be directly applied [16]. Second, a pair-wise approach is non-parametric in that the cardinality of the input vector is not dependent on the number of subtasks (or actions) that can be taken in any instance. Third, training examples of pair-wise comparisons in the data can be readily solicited [16]. From a given observation in which a subtask was scheduled, we only know which subtask was most important - not the relative importance between all subtasks. Thus, we create training examples based on pair-wise comparisons between the scheduled subtask and the unscheduled subtasks. Because we lack a context to determine the relative rank between two unscheduled subtasks, the pair-wise approach is most natural.

Consider a set of subtasks $\tau_i^j \in \boldsymbol{\tau}$, which each have a set of features $\phi_{\tau_i^j}$. For example, $\phi_{\tau_i^j}$ could include scheduling-relevant features such as the deadline, the earliest time the subtask is available, the duration of the subtask, which resource r is required by this subtask, etc. Next consider a set of m observations $O = \{O_1, O_2, \ldots, O_m\}$, such that $O_m = (s_m, a_m, t_m)$. Observation O_m consists of a feature vector $\gamma_{\tau_i^j} \forall \tau_i^j \in \boldsymbol{\tau}$ describing the state of each subtask, the action taken by the expert demonstrator, and the time at which the action was taken. This action can consist of either a scheduled subtask with the associated agents and resources assigned to that subtask or a null task (i.e., the expert does not schedule any task at this time step). This action can also be thought of as a *label* in machine learning parlance. We then need to learn a policy that will correctly predict which action to take. Within each observation where a subtask was scheduled, we do not know the relative importance of each action; rather, we only know which subtask was scheduled and the assigned agents and resources.

We decompose the problem of predicting the correct action into two steps:

Step 1) For each agent-resource pair, determine the most important subtask.

Step 2) For each such subtask, determine whether to schedule the subtask from the current state.

Learning to predict the most important subtask to schedule next (Step 1), we transform each observation O_m into a new set of observations by performing pair-wise comparisons between the scheduled subtask τ_i^j and the set of subtasks that were not scheduled (Equations 37-38). In Equation 37, we create a positive example for each observation in which a subtask τ_i^j was scheduled. The example consists of the input feature vector $\phi^m_{\langle \tau_i^j, \tau_x^y \rangle}$ and a positive label $y^m_{\langle \tau_i^j, \tau_x^y \rangle} = 1$. The input feature vector $\phi^m_{\langle \tau_i^j, \tau_x^y \rangle}$ represents the difference between the features describing the scheduled subtask τ_i^j and a subtask τ_x^y not scheduled in O_m. Intuitively, this can be thought of as the features of scheduled subtask τ_i^j relative to an unscheduled subtask τ_x^y. In Equation 38, we create a set of negative examples. We set the label to be negative, $y^m_{\langle \tau_x^y, \tau_i^j \rangle} = 0$. For the input vector, we take the difference between the unscheduled subtask τ_x^y and the scheduled subtask τ_i^j. In other words, $\phi^m_{\langle \tau_i^j, \tau_x^y \rangle} = -\phi^m_{\langle \tau_x^y, \tau_i^j \rangle}$.

$$\begin{aligned} \phi^m_{\langle \tau_i^j, \tau_x^y \rangle} &:= \gamma_{\tau_i^j} - \gamma_{\tau_x^y}, \\ y^m_{\langle \tau_i^j, \tau_x^y \rangle} &= 1, \\ \forall \tau_x^y \in \boldsymbol{\tau} \backslash \tau_i^j, \forall O_m \in O | \tau_i^j &\text{ scheduled in } O_m \end{aligned} \tag{37}$$

$$\begin{aligned} \phi^m_{\langle \tau_x^y, \tau_i^j \rangle} &:= \gamma_{\tau_x^y} - \gamma_{\tau_i^j}, \\ y^m_{\langle \tau_x^y, \tau_i^j \rangle} &= 0, \\ \forall \tau_x^y \in \boldsymbol{\tau} \backslash \tau_i^j, \forall O_m \in O | \tau_i^j &\text{ scheduled in } O_m \end{aligned} \tag{38}$$

By generating a set of pair-wise comparisons, however, we lose the *context*, or high level features that describe the overall task set $\boldsymbol{\tau}$. For example, consider a scenario where there is one resource required by many different subtasks, which serves as a resource bottleneck. If the agents work on subtasks requiring a resource other than the resource at the bottleneck, then, at the end of the schedule, the agents will have to wait in a queue to access the bottleneck resource to complete the remaining subtasks. This global information about whether a resource might be a bottleneck would be lost if we simply generated observations where the examples consisted of the difference between features of individual subtasks. To provide this high-level information, we adjust our definition in Equations 37-38 as shown in Equations 39-40, where $\xi_{\boldsymbol{\tau}}$ is a set of high-level features describing the set of subtasks for observation O_m.

$$\begin{aligned} {}^{rank}\phi^m_{\langle \tau_i^j, \tau_x^y \rangle} &:= [\xi_{\boldsymbol{\tau}}, \gamma_{\tau_i^j} - \gamma_{\tau_x^y}], \\ {}^{rank}y^m_{\langle \tau_i^j, \tau_x^y \rangle} &= 1, \\ \forall \tau_x^y \in \boldsymbol{\tau} \backslash \tau_i^j & \end{aligned} \tag{39}$$

$$\begin{aligned} {}^{rank}\phi^m_{\langle \tau_x^y, \tau_i^j \rangle} &:= [\xi_{\boldsymbol{\tau}}, \gamma_{\tau_x^y} - \gamma_{\tau_i^j}], \\ {}^{rank}y^m_{\langle \tau_x^y, \tau_i^j \rangle} &= 0, \\ \forall \tau_x^y \in \boldsymbol{\tau} \backslash \tau_i^j & \end{aligned} \tag{40}$$

We can use these observations to train a classifier $f_{priority}(\tau_i^j, \tau_x^y) \in \{0, 1\}$ to predict whether subtask τ_i^j is higher priority than τ_x^y.

Given a pair-wise classifier that can determine whether subtask τ_i^j is more important than subtask τ_x^y, we can predict which single subtask τ_i^j is the highest priority subtask τ_i^{j*} according to Equation 41. In essence, the subtask which is most often predicted to be the higher priority subtask in comparison to all other subtasks is the subtask that is predicted to have the highest overall priority.

$$\widehat{\tau_i^{j*}} = \arg\max_{\tau_i^j \in \boldsymbol{\tau}} \sum_{\tau_x^y \in \boldsymbol{\tau}} f_{priority}(\tau_i^j, \tau_x^y) \tag{41}$$

The second challenge (Step 2) is then to determine whether or not $\widehat{\tau_i^{j*}}$ should be scheduled. We train a second classifier, $f_{act}(\tau_i^j) \in \{0, 1\}$, which predicts whether or not τ_i^j should be scheduled given that it is the highest priority subtask. In our observations set O, we only have examples of scheduled subtasks, and examples when no task was scheduled. To train this classifier, we need negative examples describing which subtask was the highest priority at each time step, and why this subtask was not scheduled. While we do not know the relative priority of subtasks during some observation O_m where no subtask was scheduled, we still know which subtask will be scheduled next in some future epoch $t' > t_m$. As such, we construct a new set of examples according to Equation 42-43.

$${}^{act}\phi^m_{\tau_i^j} := [\xi_{\boldsymbol{\tau}}, \gamma_{\tau_i^j}] \tag{42}$$

$${}^{act}y^m_{\tau_i^j} = \begin{cases} 1 : \tau_i^j \text{ scheduled in } O_m \\ 0 : \tau_\emptyset \text{ scheduled in } O_m \wedge \\ \qquad \tau_i^j \text{ scheduled in } O_{m+1} \end{cases} \tag{43}$$

We can now construct a simple prediction algorithm to act as an apprentice scheduler, as shown in Figure 7. In Lines 1-8, we iterate through time. In each time step, we iterate over all agents (Line 2). For each agent, we predict the highest priority subtask $\widehat{\tau_i^{j*}}$ in Line 3. In Line 4, we predict whether we should schedule subtask $\widehat{\tau_i^{j*}}$. If our model predicts we schedule subtask $\widehat{\tau_i^{j*}}$, then we schedule subtask $\widehat{\tau_i^{j*}}$ in Line 5.

We emphasize that this formulation for an apprentice scheduler is merely a prototype. While this formulation shows promise on a synthetic data set, as we describe in Section VIII-A, we see room for improvement in future work.

VIII. Data Sets

To validate our computational method for apprenticeship scheduling, we will utilize two data sets: a synthetic data set

Apprentice Scheduler(τ, A, TC, $\tau_{\mathbf{R}}$, AC)

1: **for** $t = 0$ to T **do**
2: **for** all agents $a \in A$ **do**
3: $\widehat{\tau_i^{j*}} = \arg\max_{\tau_i^j \in \boldsymbol{\tau}} \sum_{\tau_x^y \in \boldsymbol{\tau}} f_{priority}(\tau_i^j, \tau_x^y)$ //Predict Highest Priority Task
4: **if** $f^{act}(\widehat{\tau_i^{j*}}) == 1$ //Predict Whether to Schedule Task **then**
5: Schedule $\widehat{\tau_i^{j*}}$
6: **end if**
7: **end for**
8: **end for**

Fig. 7: Pseudocode for an Apprentice Scheduler.

and a real-world data set. Our first data set will consider the scheduling problem in Section V-A. Based on our domain knowledge, we will craft heuristics to efficiently coordinate resources for a range of relevant problem parameters. Our second data set will consider the resource optimization problem for labor and delivery in Section II. To generate this real-world data set, we will collect observations of real resource nurses making scheduling decisions in a high-fidelity simulation of the labor floor.

A. Synthetic Data Set

We consider a synthetic data set based on the scheduling problem defined in Section V-A with the addition that the agents require a finite amount of time to travel between resources that is proportional to the distance between those resources. This problem is similar in spirit to the vehicle routing problem with time windows [14], [32], [93] except that there are dependencies between tasks. In our test problem, there are twenty, partially ordered subtasks, and there are two homogeneous agents able to perform these tasks. Subtasks are located in a 20×20 grid.

We construct a set of heuristics to select which agent $a \in A$ should perform each subtask $\tau_i^j \in \boldsymbol{\tau}$, as shown in Figure 8. Our heuristics are based on our prior work in scheduling [36] (Section V-B) and prior work in study of the vehicle routing problem with time windows [93]. In Line 1, the heuristic retrieves all alive and enabled subtasks $\tau_i^j \in \boldsymbol{\tau_{AE}}$. A subtask τ_i^j is alive and enabled if all of its wait constraints have been satisfied (i.e., $t \geq f_{\tau_x^y} + W_{\tau_x^y, \tau_i^j}, \forall W_{\tau_x^y, \tau_i^j}$). Next, the heuristic iterates over each agent and each subtask to find the highest priority subtask τ_i^{j*} to schedule for each agent a. In Lines 3-14, the algorithm determines which heuristic is most appropriate to apply.

If the speed of the agents is sufficiently slow (Line 1), then the travel distance will become the major bottleneck. If the agents are fast, but there are one or more resources that are heavily utilized, then these resources can become the bottleneck. Otherwise, the duration of the subtasks and their associated wait constraints are generally the most important to consider.

In Line 3, the algorithm decides travel distance as the most important bottleneck. As such, the algorithm applies a heuristic rule to find the task that maximizes a weighted,

MockHeuristic(τ, A, TC, $\tau_{\mathbf{R}}$, AC)

1: $\boldsymbol{\tau_{AE}} \leftarrow$ all alive and enabled $\tau_i^j \in \boldsymbol{\tau}$
2: **for** all agents $a \in A$ **do**
3: **if** Speed $\leq 1m \backslash s$ **then**
4: $l_v \leftarrow$ location of τ_x^y
5: $l_a \leftarrow$ location of agent a
6: $l_b \leftarrow$ location of agent $b \in A \backslash a$
7: $\tau_i^{j*} = \arg\min_{\tau_x^y \in \boldsymbol{\tau_{AE}}} \left(\alpha_1 \|l_v - l_a\| + \alpha_2 \frac{acos(l_v \cdot l_a)}{\|l_v\| \|l_a\|} + \alpha_3 \|l_b - l_a\| + \alpha_4 \frac{acos(l_b \cdot l_v)}{\|l_b\| \|l_v\|} + 1_{\left(\exists W_{\langle \tau_x^y, \tau_a^b \rangle}\right)} \right)$
8: **else if** $\sum_{\tau_i^j \in \boldsymbol{\tau}} \sum_{\tau_x^y \in \boldsymbol{\tau}} 1_{R_{\tau_i^j} = R_{\tau_x^j}} \geq \epsilon$ **then**
9: $\tau_i^{j*} = \arg\max_{\tau_x^y \in \boldsymbol{\tau_{AE}}} \left(\alpha_1' \left(\sum_{\tau_p^q} 1_{\left(R_{\tau_x^y} = R_{\tau_p^q}\right)} \right) + \alpha_2'' \left(\max\left(d_{\tau_a^b}\right) - d_{\tau_x^y} \right) \right)$
10: **else**
11: $\tau_i^{j*} = \arg\min_{\tau_x^y \in \boldsymbol{\tau_{AE}}} \left(d_{\tau_x^y}\right)$
12: **end if**
13: **if** a and r can schedule τ_i^{j*} at time t **then**
14: schedule τ_i^{j*}
15: **end if**
16: **end for**

Fig. 8: Pseudocode for the Mock Heuristic.

linear combination of hand crafted features comprised of the distance and angle relative to the origin between agent a and τ_x^y as well as the distance, the angle relative to the origin between agent τ_x^y and agent b, and an indicator term for whether τ_x^y must be executed to satisfy a wait constraint for another subtask τ_a^b. This rule is based on prior work on the vehicle routing problem [32], [93] and on Tercio's heuristic function $\pi_P(\tau_i^j)$ (Section V-D.2).

In Line 8, the algorithm determines that there may be a resource bottleneck and tries to alleviate this potential bottleneck. As such, the algorithm applies a heuristic rule which returns the subtask $\tau_i^{j*} \in \boldsymbol{\tau_{AE}}$ that maximizes a weighted, linear combination of the commonality of the subtask's required resource and its deadline. This rule is based on Tercio's heuristic functions $\pi_R(\tau_i^j)$ and $\pi_{EDF}(\tau_i^j)$ (Section V-D.2).

Lastly, if neither travel distance or resource contention are perceived to be the major bottlenecks, the algorithm applies an Earliest Deadline First rule (i.e., $\pi_{EDF}(\tau_i^j)$ in Section V-D.2).

With the heuristic shown in Figure 8, we generate a set of training data with $30,000$ unique task sets, $10,000$ for each type of bottleneck identified by the heuristic. Scheduling these task sets provides $533,737$ observations. This data set is skewed in such a way that a subtask τ_i^j is scheduled only 4% of the time. We expect to find a similar trend with observations of resource nurses, where the majority of the time is waiting for the next opportunity to make a scheduling commitment.

B. Real-World Data Set

We are currently developing a scheduling simulation to collect expert demonstrations from resource nurses at Beth Israel Deaconess Medical Center (BIDMC). A screen capture of the simulation is shown in Figure 9. This simulation is being developed in close collaboration with obstetricians from Beth Israel. The simulation will provide a high-fidelity virtual representation of the labor and delivery floor, which provides resource nurses with the ability to make the same scheduling decisions they would in their normal course of action.

At BIDMC, the labor floor has a central operating area where information is displayed and updated on a large white board. This white board contains information on each patient's age, gestational age, gravidity and parity, comorbidities, etc. Resource nurses also have access to information about which nurses will be in the hospital and for which shifts, the schedule for the elective procedures, and the occupancy in the various obstetrics wards (i.e., ante-partum ward, post-partum wards, and the neonatal intensive care unit). This information is represented virtually in our simulation (Figure 9).

The simulation allows resource nurses to take actions to control virtual labor floor resources. The resource nurses playing the simulation can assign a nurse to a patient and assign a patient to a room in triage, the labor floor, ante-partum ward, or post-partum ward. The labor nurse also has loose control over clinical management of each patient. For example, the labor nurse can ask a physician to consider performing an amniotomy to expedite labor. A resource nurse may encourage an amniotomy if the labor floor is nearing capacity in order to quicken the labor process and open up more labor beds. In addition to an amniotomy, a resource nurse can ask a patient's physician to consider a cesarean section or send a patient home where appropriate.

We plan to field this simulation during the Summer of 2015 for an initial data collection period. After this data collection period, we plan to test our prototype for learning heuristics, address any limitations, and validate the model we learn for performing the role of a resource nurse.

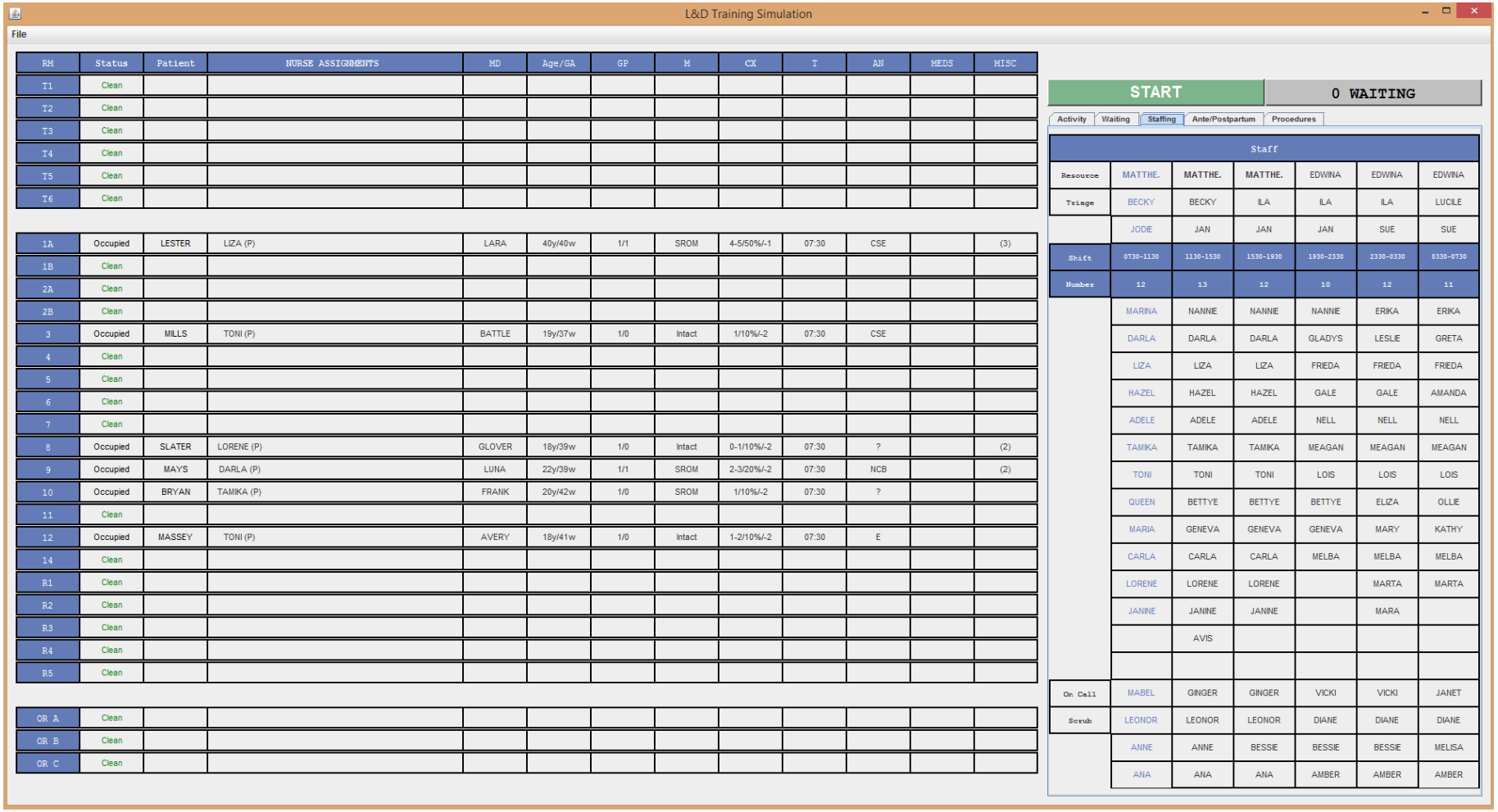
IX. Empirical Validation: Synthetic Data Set

We validate our prototype for apprenticeship learning on our synthetic data set. This validation is only an initial step, and we expect future algorithmic improvements and accompanying validation. In this empirical validation, we train our model using set of machine learning algorithms to learn when a subtask should be scheduled and which subtask is the most important to schedule next. Specifically, we train a decision tree, KNN classifier, logistic regression model, support vector machine with a radial basis function kernel (SVM-RBF), and a neural network. We used 85% of the data for training and 15% for testing. We compare the performance of our pair-wise approach with a point-wise approach and a naïve approach. In the point-wise approach for predicting the priority of a subtask, the training data consists of a set of examples where the input vector consists of the high-level features of the task set $\xi_{\boldsymbol{\tau}}$ and subtask-specific features $\gamma_{\tau_i^j}$ as well as a binary label for whether subtask τ_i^j was scheduled in that observation. In the naïve approach, the examples are comprised of an input vector that concatenates the high-level features of the task set $\xi_{\boldsymbol{\tau}}$ and the subtask-specific features $\gamma_{\tau_i^j}$ for all subtasks in the form of $\{\gamma_{\tau_i^j} | \forall \tau_i^j \in \boldsymbol{\tau}\} \cup \xi_{\boldsymbol{\tau}}$. Each example has a label consisting of the index of the subtask that was scheduled in the corresponding observation.

Table I shows the sensitivity (i.e., true positive rate) and specificity (i.e., true negative rate) of the model. Overall, we find that a pair-wise model outperforms the point-wise and naïve approaches. Within the pair-wise model, a decision tree provides the best performance. The trained decision tree is able to identify the correct subtask and when to schedule that subtask 95% of the time, and the decision tree is able to predict when no subtask should be scheduled for 96% of the time.

A decision tree model seems to work well for a couple reasons. First, the heuristic described in Figure 8 has several modes of operation depending on the speed of the agents and the degree of resource contention. The decision tree is able to accurately capture these modes by learning accurate splitting criteria for the features describing these modes. Second, the heuristic operates according to temporal and resource constraints: each agent performs only one subtask at a time, each resource can be accessed by only one agent at a time, and a certain amount of time must pass for an agent to travel from one subtask to the next. The decision tree can learn which features describe the satisfaction of these constraints. However, a decision tree is limited in a key regard: a decision tree is a less natural mechanism for learning a linear (or polynomial) model, as in Lines 7 and 9 of the mock heuristic in Figure 8. We hypothesize that a model that bridges a decision tree, for learning the modes of the heuristic and operational constraints, and regression algorithm, for learning prioritization within these modes, would more accurately mimic the behavior of operator heuristics.

Furthermore, the pair-wise model we propose as our prototype in Section VII seems to work well because it is able to effectively capture the information contained within the data. As we noted in Section VII, for a given observation in which a subtask was scheduled, we only know which subtask was scheduled (i.e., highest rank) and which subtasks were not scheduled (i.e., lower rank). Training a model based on pair-wise comparison between the scheduled subtask and the unscheduled subtasks effectively captures the ability to predict the subtask that should be scheduled next. A point-wise approach, however, where each subtask is assigned a relative priority, suffers because a regression model where training labels are binary (i.e., high for scheduled and low for unscheduled) loses much of the information contained within the observations. The naïve approach, in which the features of all subtasks are provided as an all-encompassing feature vector, also suffers some major challenges. First, the size of the input space grows exponentially with the number of subtasks. In general, one then needs significantly



RM	Status	Patient	NURSE ASSIGNMENTS	MD	Age/GA	GP	M	CX	T	AN	MEDS	MISC
T1	Clean											
T2	Clean											
T3	Clean											
T4	Clean											
T5	Clean											
T6	Clean											
1A	Occupied	LESTER	LIZA (P)	LARA	40y/40w	1/1	SROM	4-5/50%/-1	07:30	CSE		(3)
1B	Clean											
2A	Clean											
2B	Clean											
3	Occupied	MILLS	TONI (P)	BATTLE	19y/37w	1/0	Intact	1/10%/-2	07:30	CSE		
4	Clean											
5	Clean											
6	Clean											
7	Clean											
8	Occupied	SLATER	LORENE (P)	GLOVER	18y/39w	1/0	Intact	0-1/10%/-2	07:30	?		(2)
9	Occupied	MAYS	DARLA (P)	LUNA	22y/39w	1/1	SROM	2-3/20%/-2	07:30	NCB		(2)
10	Occupied	BRYAN	TAMIKA (P)	FRANK	20y/42w	1/0	SROM	1/10%/-2	07:30	?		
11	Clean											
12	Occupied	MASSEY	TONI (P)	AVERY	18y/41w	1/0	Intact	1-2/10%/-2	07:30	E		
14	Clean											
R1	Clean											
R2	Clean											
R3	Clean											
R4	Clean											
R5	Clean											
OR A	Clean											
OR B	Clean											
OR C	Clean											

Resource	MATTHE.	MATTHE.	MATTHE.	EDWINA	EDWINA	EDWINA
Triage	BECKY	BECKY	ILA	ILA	ILA	LUCILE
	JODIE	JAN	JAN	JAN	SUE	SUE
Shift	0730-1130	1130-1530	1530-1930	1930-2330	2330-0330	0330-0730
Number	12	13	12	10	12	11
	MARINA	NANNIE	NANNIE	NANNIE	ERIKA	ERIKA
	DARLA	DARLA	DARLA	GLADYS	LESLIE	GRETA
	LIZA	LIZA	LIZA	FRIEDA	FRIEDA	FRIEDA
	HAZEL	HAZEL	HAZEL	GALE	GALE	AMANDA
	ADELE	ADELE	ADELE	NELL	NELL	NELL
	TAMIKA	TAMIKA	TAMIKA	MEAGAN	MEAGAN	MEAGAN
	TONI	TONI	TONI	LOIS	LOIS	LOIS
	QUEEN	BETTYE	BETTYE	BETTYE	ELIZA	OLLIE
	MARIA	GENEVA	GENEVA	GENEVA	MARY	KATHY
	CARLA	CARLA	CARLA	MELBA	MELBA	MELBA
	LORENE	LORENE	LORENE		MARTA	MARTA
	JANINE	JANINE	JANINE		MARA	
		AVIS				
On Call	MABEL	GINGER	GINGER	VICKI	VICKI	JANET
Scrub	LEONOR	LEONOR	LEONOR	DIANE	DIANE	DIANE
	ANNE	ANNE	BESSIE	BESSIE	BESSIE	MELISA
	ANA	ANA	ANA	AMBER	AMBER	AMBER

Fig. 9: Screen Capture of our Simulation of a labor and delivery Floor.

	Prototype (Pair-wise)	Point-wise	Naïve Approach
Decision Tree	95.0% / 96.0%	29.4% / 99.3%	2.81% / 70.6%
KNN	37.5% / 64.8%	4.89% / 27.29%	7.87% / 68.5%
Logistic Regression	73.8% / 69.6%	4.35% / 67.7%	- / -
SVM-RBF	4.38% / 99.3%	2.17% / 99.6%	1.69% / 99.4%
Neural Network	71.9% / 60.7%	0.00% / 99.9%	2.81% / 94.3%
Random	6.49% / 50.00%	6.49% / 50.00%	6.49% / 50.00%

TABLE I: The empirical performance (i.e., sensitivity / specificity) of a set of machine learning algorithms when predicting the priority of a subtask τ_i^j and whether or not to schedule that subtask.

more data to learn an accurate model. Second, knowledge about the structure of the problem is lost. For example, the model does not know that certain subsets of the input vector correspond to the same features of different subtasks. For example, consider a scenario where features $1-20$ describe the distance agent a will have to travel to subtasks $1-20$, and features $21-40$ describe the duration of subtasks $1-20$. By binning the subtasks' features, the relationship within and between differing classes of features is hidden and must be recovered during training. This relationship could theoretically be recovered with sufficient data and the right machine learning algorithm. However, the problem of learning an accurate model becomes much more difficult.

X. Proposed Work

While our prototype for learning heuristics may work in our synthetic data set, there are challenges remaining to directly applying our prototype on the real data from resource nurses. First, resource nurses have a finite reaction time. The state in which a resource nurse may decide to act will be after the stimulus that prompted the action. Thus, we need a mechanism to link the stimulus with the action. Second, a nurse's heuristic may find that any one of a set of scheduling actions would be equally valid, and any one of those actions may be selected at random. We recall from Section II that resource nurses can assign nurses to patients, assign patients to rooms (i.e., resources), loosely control the start and finish time of the various steps in the care process, and accept or reject patients. We suspect that we may be able to learn which type of room to assign a patient (e.g., operating room versus labor room), but not necessarily which room out of a set of common rooms. Similarly, we may be able to learn that resource nurses would give a patient to a nurse who currently has no other patients. However, if there are multiple nurses who have no patients, it may be difficult to distinguish which of them would be given the patient.

To develop and demonstrate an effective apprenticeship scheduling framework, we propose a two-step process: Step 1 is a passive learning phase and Step 2 is an active learning phase. In Step 1, we will perform an initial data collection phase using our high-fidelity simulation of resource nursing on the labor and delivery floor at Beth Israel Deaconess Medical Center. With this data, we will test our prototype model for learning resource optimization heuristics from Section VII. We anticipate extending this model to handle the temporal distance between the stimulus and the response from the demonstrating resource nurse. We hypothesize that this model will be able to capture important information about when the state of the labor floor requires action and what type of action to take (e.g., assign a free nurse to a given patient). However, we will likely not be able to discern at the lowest level between an equivalent set of actions (e.g., which free nurse should be assigned).

In Step 2, we will embed an active learning capability within our apprenticeship scheduling technique to improve model accuracy. With the knowledge we gained during our first step, we can field an autonomous agent embedded within the labor floor simulation. Nurses will see the actions recommended by the autonomous agent and offer correction when necessary. In this way, by testing which of a set of seemingly equal-priority subtasks could be scheduled next, we can tease out which actions are truly of equal value or if there is a more subtle bias in the manner in which resource nurses are acting. This second step will require an algorithmic extension to handle the information provided from the demonstrators in real-time as a part of an iterative, active learning framework.

By demonstrating that our apprenticeship scheduling approach performs well on both a synthetic data set, which captures elements across many important scheduling problems (Section VIII-A), and a real-world data set collected in one of the most challenging scheduling domains (Section VIII-B), we believe that we can demonstrate a fundamental advancement in resource optimization.

XI. Time Line

- May 2015: Thesis proposal defense.
- Summer 2015: Initial data collection.
- Fall 2015: Analyze data from initial data collection phase and improve/extend apprentice scheduler prototype as a part of Step 1.
- Spring 2016: Develop an active learning framework for apprenticeship scheduling as part of Step 2.
- Summer 2017: Collect data and validate the active learning framework as a continuation of Step 2.
- Fall 2016: Thesis writing and revision.
- January 2017: Ph.D. thesis defense.
- February 2017: Graduation.

XII. Acknowledgments

This work was supported by Boeing Research and Technology and by the National Science Foundation (NSF)

Graduate Research Fellowship Program (GRFP) under grant number 2388357.

REFERENCES

[1] J. A. Adams. Multiple robot-single human interaction: effects on perceived workload and performance. *Behavior and Information Technology*, 28(2):183–298, 2009.

[2] U. Anders and O. Korn. Model selection in neural networks. *Neural Networks*, 12(2):309 – 323, 1999.

[3] L. Ardissono, G. Petrone, G. Torta, and M. Segnan. Mixed-initiative scheduling of tasks in user collaboration. In *Proceedings of the Eighth International Conference on Web Information Systems and Technologies*, pages 342–351, 2012.

[4] M. J. Barnes, J. Y. C. Chen, F. Jentsch, and E. S. Redden. Designing effective soldier-robot teams in complex environments: training, interfaces, and individual differences. In *Proceedings of the International Conference on Engineering Psychology and Cognitive Ergonomics (EPCE)*, pages 484–493. Springer, 2011.

[5] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720. AAAI Press, 1998.

[6] P. Berry, B. Peintner, K. Conley, M. Gervasio, T. Uribe, and N. Yorke-Smith. Deploying a personalized time management agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1564–1571, New York, NY, USA, 2006. ACM.

[7] D. Bertsimas and R. Weismantel. *Optimization over Integers*. Dynamic Ideas, Belmont, 2005.

[8] E. Blickensderfer, J. A. Cannon-Bowers, and E. Salas. Cross-training and team performance. In *Making decisions under stress: Implications for individual and team training*, pages 299–311. American Psychological Association, Washington, DC, 1998.

[9] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Pool. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, February 2004.

[10] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 71–80, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[11] S. J. Bradtke and M. O. Duff. Reinforcement learning methods for continuous-time markov decision problems. In *Advances in Neural Information Processing Systems*, pages 393–400. MIT Press, 1994.

[12] L. Brandenburg, P. Gabow, G. Steele, J. Toussaint, and B. J. Tyson. Innovation and best practices in health care scheduling. Technical report, February 2015.

[13] L. Brunet, H.-L. Choi, and J. P. How. Consensus-based auction approaches for decentralized task assignment. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference (GNC)*, Honolulu, HI, 2008.

[14] L. Bush, B. Williams, and N. Roy. Unifying plan-space value-based approximate dynamic programming policies and open loop feedback control. In *Proceedings of the AIAA Infotech@Aerospace Conference*, 2012.

[15] D. Cai, X. He, J.-R. Wen, and W.-Y. Ma. Block-level link analysis. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 440–447, New York, NY, USA, 2004. ACM.

[16] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[17] J. Casper and R. R. Murphy. Human-robot interaction in rescue robotics. *IEEE Transaction on Systems, Man, and Cybernetics (SMCS)*, 34(2):138–153, 2004.

[18] J. Y. Chen, M. J. Barnes, and Z. Qu. Roboleader: an agent for supervisory control of mobile robots. In *Proceedings of the International Conference on Human-Robot Interaction (HRI)*, 2010.

[19] S. Chernova and M. Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 233:1–233:8, New York, NY, USA, 2007. Association for Computing Machinery.

[20] S. Chernova and M. Veloso. Multi-thresholded approach to demonstration selection for interactive robot learning. In *3rd ACM IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, March 2008.

[21] Y. H. Cho, J. K. Kim, and S. H. Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23(3):329 – 342, 2002.

[22] A. S. Clare, M. L. Cummings, J. P. How, A. K. Whitten, and O. Toupet. Operator objective function guidance for a real-time unmanned vehicle scheduling algorithm. *AIAA Journal of Aerospace Computing, Information and Communication*, 9:161–173, 2012.

[23] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR'99 Workshop on Recommender Systems*, 1999.

[24] M. L. Cummings, A. S. Brzezinski, and J. D. Lee. Operator performance and intelligent aiding in unmanned aerial vehicle scheduling. *IEEE Intelligent Systems*, 22(2):52–59, March 2007.

[25] T. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck. Solving semi-markov decision problems using average reward reinforcement learning. *Management Science*, 45:560–574, 1999.

[26] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *AI*, 49(1):61–91, 1991.

[27] M. B. Dias. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University, January 2004.

[28] E. H. Durfee, J. C. B. Jr., and J. Sleight. Using hybrid scheduling for the semi-autonomous formation of expert teams. *Future Generation Computer Systems*, July 2013.

[29] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64, 1995.

[30] E. E. Entin and D. Serfaty. Adaptive team coordination. *Human Factors*, 41:312–325, 1999.

[31] D. Fox. Adapting the sample size in particle filters through kld-sampling. *International Journal of Robotics Research (IJRR)*, 22:985–1003, 2003.

[32] L. M. Gambardella, Éric Taillard, and G. 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.

[33] M. C. Gombolay. Fast methods for scheduling with applications to real-time systems and large-scale robotic manufacturing of aerospace structures. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2013.

[34] M. C. Gombolay, R. A. Gutierrez, G. F. Sturla, and J. A. Shah. Decision-making authority, team efficiency and human worker satisfaction in mixed human-robot teams. In *Proceedings of the Robots: Science and Systems (RSS)*, Berkeley, California, July 12-16, 2014.

[35] M. C. Gombolay and J. A. Shah. Multiprocessor scheduler for task sets with well-formed precedence relations, temporal deadlines, and wait constraints. In *Proceedings of the AIAA Infotech@Aerospace*, 2012.

[36] M. C. Gombolay, R. J. Wilcox, and J. A. Shah. Fast scheduling of multi-robot teams with temporospatial constrints. In *Proceedings of the Robots: Science and Systems (RSS)*, Berlin, Germany, June 24-28, 2013.

[37] M. A. Goodrich, B. S. Morse, C. Engh, J. L. Cooper, and J. A. Adams. Towards using UAVs in wilderness search and rescue: Lessons from field trials. *Interaction Studies, Special Issue on Robots in the Wild: Exploring Human-Robot Interaction in Naturalistic Environments*, 10(3):453–478, 2009.

[38] M. L. D. Grano, D. J. Medeiros, and D. Eitel. Accommodating individual preferences in nurse scheduling via auctions and optimization. *Healthcare Management Science*, 12:228–242, September 2009.

[39] M. Hamasaki, H. Takeda, I. Ohmukai, and R. Ichise. Scheduling support system for academic conferences based on interpersonal networks. In *Proceedings of ACM Hypertext*, 2004.

[40] T. H. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the 11th International Conference on World Wide Web*, pages 517–526, New York, NY, USA, 2002. ACM.

[41] T. Haynes, S. Sen, N. Arora, and R. Nadella. An automated meeting scheduling system that utilizes user preferences. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 308–315, New York, NY, USA, 1997. ACM.

[42] R. Herbrich, T. Graepel, and K. Obermayer. *Large Margin Rank Boundaries for Ordinal Regression*, chapter 7, pages 115–132. MIT Press, January 2000.

[43] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, Jan. 2004.

[44] J. N. Hooker. Logic-based benders decomposition. *Mathematical Programming*, 96:33–60, 2003.

[45] E. R. Hooten, S. T.Hayes, and J. A. Adams. A comparison of communicative modes for map-based tasking. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2011.

[46] C.-M. Huang and B. Mutlu. Learning-based modeling of multimodal behaviors for humanlike robots. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, pages 57–64, New York, NY, USA, 2014. ACM.

[47] M. Ibnkahla. Applications of neural networks to digital communications - a survey. *Signal Processing*, 80(7):1185 – 1215, 2000.

[48] T. Inamura, M. Inaba, and H. Inoue. Acquisition of probabilistic behavior decision model based on the interactive teaching method. In *"Proceedings of the International Conference on Advanced Robotics*, 1999.

[49] R. Jin, H. Valizadegan, and H. Li. Ranking refinement and its application to information retrieval. In *Proceedings of the 17th International Conference on World Wide Web*, pages 397–406, New York, NY, USA, 2008. Association for Computing Machinery.

[50] D. G. Jones and M. R. Endsley. Sources of situational errors in aviation. *Aviation, Space, and Environmental Medicine*, 67(6):507–512, 1996.

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

[52] H. L. Jones, S. M. Rock, D. Burns, and S. Morris. Autonomous robots in SWAT applications: Research, design, and operations challenges. *Association for Unmanned Vehicle Systems International*, 2002.

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

[54] H. K. Kim, J. K. Kim, and Y. Ryu. Personalized recommendation over a customer network for ubiquitous shopping. *IEEE Transactions on Services Computing*, 2, April 2009.

[55] J. K. Kim, Y. H. Cho, W. J. Kim, J. R. Kim, and J. H. Suh. A personalized recommendation procedure for internet shopping support. *Electronic Commerce Research and Applications*, 1(34):301 – 313, 2002.

[56] G. A. Klein. *A recognition-primed decision (RPD) model of rapid decision making*. Ablex Publishing Corporation, New York, NY, USA, 1993.

[57] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[58] G. A. Korsah, A. Stentz, and M. B. Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[59] P. Laborie. Algorithms for propagating resource constraints in AI planning and scheduling: existing approaches and new results. *Artificial Intelligence*, 143(2):151–188, February 2003.

[60] K. Lakshmanan, S. Kato, and R. R. Rajkumar. Open problems in scheduling self-suspending tasks. In *Proceedings of the Real-Time Scheduling Open Problems Seminar (RTSOPS)*, Brussels, Belgium, July 6 2010.

[61] K. Lakshmanan and R. R. Rajkumar. Scheduling self-suspending real-time tasks with rate-monotonic priorities. In *Proceedings of the Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Stockholm, Sweden, April 12-15 2010.

[62] P. Li, Q. Wu, and C. J. Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 897–904. MIT Press, Cambridge, MA, 2007.

[63] W. Lihua, L. Lu, L. Jing, and L. Zongyong. Modeling user multiple interests by an improved {GCS} approach. *Expert Systems with Applications*, 29(4):757 – 767, 2005.

[64] G. S. Linoff and M. J. Berry. *Data Mining Techniques: For Marketing, Sales and Customer Relationship Management*. Wiley, Hoboken, New Jersey, 2004.

[65] C. Liu and J. H. Anderson. Task scheduling with self-suspensions in soft real-time multiprocessor systems. In *Proceedings of the 30^{th} IEEE Real-Time Systems Symposium (RTSS)*, Washington DC, U.S.A., December 1-4 2009.

[66] C. Liu and J. H. Anderson. Improving the schedulability of sporadic self-suspending soft real-time multiprocessor task systems. In *Proceedings of the 16^{th} IEEE International Conference on Real-Time Computing Systems and Applications (RTCSA)*, Hong Kong, China, August 23-25 2010.

[67] S. Macho, M. Torrens, and B. Faltings. A multi-agent recommender system for planning meetings. In *Proceedings of the ACM Conference on Autonomous Agents, Workshop on Agent-based Recommender Systems*, 2000.

[68] C. F. Mackenzie, Y. Xiao, and R. Horst. Video task analysis in high performance teams. *Cognition, Technology, and Work*, 6:139–147, 2004.

[69] N. K. Malhotra. *Marketing Research: an Applied Orientation*. Prentice Hall, Upper Saddle River, New Jersey, 2010.

[70] B. Michini and J. P. How. Bayesian nonparametric inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, volume 7524 of *Lecture Notes in Computer Science*, pages 148–163. Springer Berlin Heidelberg, 2012.

[71] N. Muscettola, P. Morris, and I. Tsamardinos. Reformulating temporal plans for efficient execution. In *Proceedings of the 6^{th} International Conference on Principles of Knowledge Representation and Reasoning (KR&R)*, Trento, Italy, June 2-5, 1998.

[72] S. Nikolaidis and J. Shah. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *Proceedings of the International Conference on Human-Robot Interaction (HRI)*, pages 33–40, 2013.

[73] M. Öztürké, A. Tsoukiàs, and P. Vincke. Preference modelling. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*, pages 27–59. Springer New York, 2005.

[74] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

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

[76] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. Situation awareness, mental workload, and trust in automation: Viable, empirically supported cognitive engineering constructs. *Journal of Cognitive Engineering and Decision Making*, 2(2):140–160, 2008.

[77] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059 – 10072, 2012.

[78] S. D. Pizer and J. 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.

[79] R. R. Rajkumar. Dealing with self-suspending period tasks. Technical report, IBM, Thomas J. Watson Research Center, Armonk, New York, 1991.

[80] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2586–2591, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[81] V. Ramanujam and H. Balakrishnan. Estimation of maximum-likelihood discrete-choice models of the runway configuration selection process. In *American Control Conference (ACC), 2011*, pages 2160–2167, June 2011.

[82] P. Reverdy, V. Srivastava, and N. E. Leonard. Modeling human decision-making in generalized gaussian multi-armed bandits. *CoRR*, abs/1307.6134, 2013.

[83] P. Richard. On the complexity of scheduling real-time tasks with self-suspensions on one processor. In *Proceedings of the 15^{th} Euromicro Conference on Real-Time Systems (ECRTS)*, Porto, Portugal, July 2-4 2003.

[84] F. Ridouard and P. Richard. Negative results for scheduling independent hard real-time tasks with self-suspensions. In *Proceedings of the Real-Time and Network Systems (RTNS)*, Poitiers, France, May 30-31 2006.

[85] J. C. Ryan, A. G. Banerjee, M. L. Cummings, and N. Roy. Comparing the performance of expert user heuristics and an integer linear program in aircraft carrier deck operations. *IEEE Transaction on Cybernetics*, PP(9), August 2013.

[86] P. E. Rybski and R. M. Voyles. Interactive task training of a mobile robot through human gesture recognition. In *IEEE International Conference on Robotics and Automation*, pages 664–669, 1999.

[87] E. Salas, J. E. Fowlkes, R. J. Stout, D. M. Milanovich, and C. Prince. Does CRM training improve teamwork skills in the cockpit?: Two evaluation studies. *Human Factors*, 41:326–343, 1999.

[88] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 385–393, July 1992.

[89] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system - a case study. In *Proceedings of the ACM WEBKDD workshop*, 2000.

[90] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, pages 118–125, New York, NY, USA, 2006. ACM.

[91] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web*, pages 291–324. Springer-Verlag, Berlin, Heidelberg, 2007.

[92] S. A. Shipman and C. A. Sinsky. Expanding primary care capacity by reducing waste and improving efficiency of care. *Health Affairs (Millwood)*, 32(11):1990–1997, 2013.

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

[94] M. Soomer and G. Franx. Scheduling aircraft landing using airlines' preferences. *European Journal of Operational Research*, 190:277–291, October 2008.

[95] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo. In *Deadline Scheduling for Real-Time Systems*, volume 460 of *The Springer International Series in Engineering and Computer Science*. 1998.

[96] P. Szolovits, R. S. Patil, and W. B. Schwartz. Artificial intelligence in medical diagnosis. *Annals of Internal Medicine*, 108(1):80–87, 1988.

[97] P. Szolovits and S. G. Pauker. Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence*, 11(12):115 – 144, 1978. Applications to the Sciences and Medicine.

[98] F. Tang and L. E. Parker. ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings of the International Conference on Robotics and Automation*, May 2005.

[99] A. Terrell and B. Mutlu. A regression-based approach to modeling addressee backchannels. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 280–289, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[100] A. L. Thomaz and C. Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1000–1005. AAAI Press, 2006.

[101] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing NDCG measure. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1883–1891. Curran Associates, Inc., 2009.

[102] P. Vilím, R. Barták, and O. Čepek. Extension of O(n log n) filtering algorithms for the unary resource constraint to optional activities. *Constraints*, 10(4):403–425, 2005.

[103] M. N. Volkovs and R. S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1089–1096, New York, NY, USA, 2009. ACM.

[104] C. Volpe, J. Cannon-Bowers, E. Salas, and P. Spector. The impact of cross-training on team functioning: an empirical investigation. *Human Factors*, 38:87–100, 1996.

[105] R. J. Wilcox, S. Nikolaidis, and J. A. Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. In *Proceedings of Robotics: Science and Systems (RSS)*, Sydney, Australia, July 9-13 2012.

[106] R. J. Wilcox and J. A. Shah. Optimization of multi-agent workflow for human-robot collaboration in assembly manufacturing. In *Proceedings of the AIAA Infotech@Aerospace*, 2012.

[107] S.-Z. Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215 – 243, 2010. Special Review Issue.

[108] H. Zhang, E. Law, R. Miller, K. Gajos, D. Parkes, and E. Horvitz. Human computation tasks with global constraints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 217–226, New York, NY, USA, 2012. ACM.