Robust Scheduling Strategies for Collaborative Human-UAV Missions

Jeffrey R. Peters and Luca F. Bertuccelli

***Abstract*— Resource allocation in collaborative human-UAV missions has become an important research area in recent years. Traditional deterministic strategies for task scheduling, such as job-shop schemes, can lead to poor performance, since these strategies fail to account for human cognitive requirements or behavioral uncertainty. In response, we present a flexible mixed-integer linear programming framework that can potentially address both of these issues in finite horizon scheduling applications. Specifically, we illustrate how cognitive workload constraints can be formulated as a mixed-integer linear program, and introduce robustness to uncertain processing times through the use of scenarios. We explore the modularity and utility of this simple framework by introducing additional layers of complexity, including receding horizon planning and adaptive estimation. Throughout the discussion, we use simulation studies to discuss the functionality of these algorithms, as well as various issues regarding practical implementation.**

I. Introduction

In futuristic scenarios involving collaboration between humans and unmanned aerial vehicles (UAVs), human operators are often charged with the sequential processing of tasks that are generated by their UAV partners, e.g. [1]. Proper scheduling of these tasks can have a profound impact on both operator and mission performance, since key mission planning decisions often depend on the operator's ability to process tasks quickly and accurately [2]. These types of discrete scheduling problems are NP-hard in the general case [3]. Despite this, the traditional deterministic scheduling problem and its variations have been studied for many years in the context of job-shop applications (e.g., [4]), and high-quality heuristic methods exist for constructing solutions. Common strategies involve integer programming [5], disjunctive graphs [6], and various other heuristics [7]. The quality of deterministic scheduling solutions, however, deteriorates quickly in the presence of uncertainty [8]. As such, schemes designed for deterministic problems are usually ill-suited for human-centered systems, which often involve significant behavioral variance. For example, in the context of visual search, virtually all models of human cognitive processing are stochastic, and thus task processing times carry significant uncertainty. Although some strategies do exist for discrete scheduling or robust optimization in uncertain or dynamic environments (e.g., [9], [10]), existing methods fail to account for crucial elements of human information processing. Indeed, factors such as cognitive workload, fatigue, memory retention, among others, are not generally taken into account in typical dynamic or robust scheduling, although they can easily affect operator performance in persistent task execution missions if not properly mediated [11].

In what follows, we develop a mixed-integer linear program (MILP) framework for constructing solutions to finite horizon task scheduling problems. Our method maximizes the operator's achieved reward in the presence of processing time uncertainty, while simultaneously mediating operator cognitive workload by penalizing schedules that are likely to cause workload levels to fall outside of a pre-specified regime. We focus on cognitive workload, as opposed to other exogenous human factors, since it has well-established trends and links to performance that can feasibly be exploited by mission planners [12]. After establishing the base framework, we illustrate how additional layers of complexity can be added in order to both boost performance and address more general situations. Our work is largely an extension of [13], and is intended to illustrate the practical utility of MILP frameworks in the context of *human supervisory control* [14].

Specifically, our contributions are as follows. First, we present an MILP framework for the scheduling problem that can incorporate workload considerations for simple, trend-based workload models. Then, we illustrate how robustness to task processing time uncertainty can be added into this formulation through the use of scenarios. We show how this strategy allows flexibility in choosing both the desired degree of robustness and the degree that workload is taken into account. We then illustrate how general performance can be improved through the use of a receding horizon approach. Finally, we extend our problem setup to allow for additional uncertainty with regard to task processing time distributions, and illustrate how straightforward estimation schemes can be added to the established receding horizon approach.

II. Scheduling Under Workload Constraints

A. Main Scheduling Objective

Consider $N \in \mathbb{N}$ heterogenous tasks stacked in a queue awaiting the attention of a single operator. Each task T_i, $i \in \{1, \ldots, N\}$, has 1) an associated processing time $t_i \in \mathbb{R}_{>0}$, which defines how long the task will take to complete, 2) an availability time $s_i \in \mathbb{R}_{>0}$, which defines the global time at which the task becomes available to the operator, and 3) an associated reward $r_i \in \mathbb{R}_{\geq 0}$, which is given upon successful completion. We consider an "all or nothing" reward distribution scheme, in which the operator receives the full reward r_i if the task is completed, and no reward otherwise, i.e., there is no pre-emption. Further, we assume that the reward is only obtained if the task is completed within a pre-specified time horizon $T_H \in \mathbb{R}_{>0}$.

We seek a schedule, i.e., ordered sequence of tasks, that maximizes the total reward obtained, provided that the operator starts each task in the sequence at the minimum possible time, i.e., the maximum of the previous task completion time (or $t = 0$ for the first task) and the appropriate availability

constraint. Note that, with this assumption, task availability constraints $\{s_i\}$ are never violated. Formally, a *schedule* is a sequence $S \coloneqq \{T_{\sigma(1)}, T_{\sigma(2)}, \ldots, T_{\sigma(|S|)}\}$, where $|S| \leq N$ and $\sigma : \{1, \ldots, N\} \to \{1, \ldots, N\}$ is some bijective mapping. Let $\mathcal{S}$ denote the set of all possible schedules. Assuming the operator processes tasks sequentially, starting each task at the minimum possible time, we seek $S^* \in \mathcal{S}$ that maximizes the total reward obtained within time T_H.

B. Workload Constraints

The effectiveness of a given schedule also hinges upon the cognitive state of the operator. Therefore, it is desirable to construct a schedule that allows the operator's cognitive state to remain in a regime that is most amenable to high levels of performance. Here, we focus our attention specifically on moderating cognitive workload. Formally, *cognitive workload* is the extent to which a task or set of tasks places a demand on the operator's cognitive resources [15]. Perceived workload is, in turn, closely related to the concept of *operator stress*, which has direct links to performance [11]. These links are typically modeled through classical laws such as the Yerkes-Dodson law, which says that moderate levels of operator stress are the most beneficial [16]. Cognitive workload is often used as a means of mediating operator stress with the hope of achieving greater performance, e.g., [12].

Although perceived cognitive workload is generally subjective, there are a number of objective, quantifiable means that have been used to capture high-level dynamic evolution with some success, e.g., utilization ratio [17], neurophysiological metrics [18], among others. We choose to model workload via an incremental, discrete process, which is driven by the task processing order and processing times. This model is based on the simple observation that, in most situations, when operators are busy (idle), their workload level increases (decreases). To capture these simple dynamics, to each task T_i, we associate an increment δw_i, which represents the amount that the operator's cognitive workload increases when working on T_i for time t_i. Further, we assume that workload decreases when the operator is idle, proportional to idle time. With this model, we seek to solve the nominal scheduling problem under the additional constraint that, if possible, the operator's workload must remain within a desired regime $[\underline{w}, \overline{w}] \subset \mathbb{R}$ at any point $0 \leq t \leq T_H$. Although simplistic, this model captures the essence of workload evolution during task processing. Indeed, many widely accepted workload evolution models are deterministic processes that augment and degrade workload during busy and idle times, respectively (e.g., [17]). Further, in the sequel, we treat processing times as known parameters for optimization (including the scenario-based formulation of Section III-A). Therefore, if desired, the parameters δw_i can be systematically chosen to reflect more sophisticated dynamics. For example, if $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a function that relates time to the amount that workload increments when the operator engages in a general task, then we can set $\delta w_i = f(t_i)$ for all $i \in \{1, \ldots, N\}$. A similar statement can be made regarding workload decrements during idle time.

Despite this flexibility, the incremental workload model considered herein cannot account for dynamics in which increment or decrement magnitudes are dependent upon initial conditions, since the task processing order is unknown *a priori*. Recall, however, that for a given operator, workload evolution will usually be a subjective experience, and thus fine level models may be ill-suited if they are derived from aggregate data. Therefore, in constructing general schemes for use with many different operators, simplistic workload evolution dynamics may be preferable to finer level models. If, however, the model is to be tuned to a specific operator or group of operators, or more precise real-time data can be leveraged (e.g., neurophysiological cues [19]), then alternative models may be preferable and the MILP framework presented herein may not be sufficient.

C. MILP Formulation

We now formulate the finite horizon scheduling problem as an MILP. The primary decision variables are binary indicators $x_{i,j} \in \{0, 1\}$, which specify whether or not task T_i should be executed in the j-th time slot of the output sequence (Fig. 1). Each task T_i is fully specified by the 4-tuple $(t_i, s_i, r_i, \delta w_i)$. In addition to $T_1, \ldots, T_N$, we introduce one additional *null-task* T_{N+1} to represent an operator idle time. Specifically, we define $T_{N+1} \coloneqq (\zeta, 0, 0, -\delta w_0)$, where $\delta w_0 > 0$ is a constant and $\zeta \in \mathbb{R}_{\geq 0}$ is the (fixed) task length. The null-task is the only task that the operator is allowed to execute more than once. As such, the output of our proposed method is an *augmented schedule*, which is formally defined as a sequence $S = \{T_{\sigma(1)}, T_{\sigma(2)}, \ldots, T_{\sigma(|S|)}\}$, where $M \geq |S| \in \mathbb{N}$ with M being a pre-determined parameter representing the maximum number of tasks that can appear in the sequence, and $\sigma : \{1, \ldots, M\} \to \{1, \ldots, N\}$ is some mapping satisfying $|\sigma^{-1}(\{i\})| \leq 1$ for all $i \in \{1, \ldots, N\}$. This differs slightly from a *schedule*[1] in that the null-task can appear more than once, and thus M can exceed the number of non-null tasks, N. To guarantee reasonable outputs, it is necessary to pick M sufficiently large, i.e., to consider sequences that can incorporate a sufficiently large number of terms. Setting $M = \lceil T_H/\zeta \rceil + N$ is sufficient, where $\lceil \cdot \rceil : \mathbb{R} \to \mathbb{Z}$ is the standard ceiling operator. With the goal of capturing workload evolution within an MILP, we augment the set of tasks to be processed with the newly created null task, and define $\mathcal{T} \coloneqq \{T_1, \ldots, T_N, T_{N+1}\}$. Workload is then incorporated as an explicit variable that satisfies appropriate constraints. As such, the scheduling problem reduces to the issue of finding the optimal augmented schedule based on the set $\mathcal{T}$. Consequently, the MILP takes the form (1). In (1), the optimization is performed over the decision variables $x_{i,j}$, C_j, B_j, W_j, β, and γ, where $i \in \{1, \ldots, N\}$, $j \in \{1, \ldots, M\}$, and all other parameters are fixed. This formulation provides a natural extension of [13], which considers an analogous problem

[1]The difference is not crucial to most of the discussion, however, so we reserve the qualifier "augmented" for cases where a distinction is required.

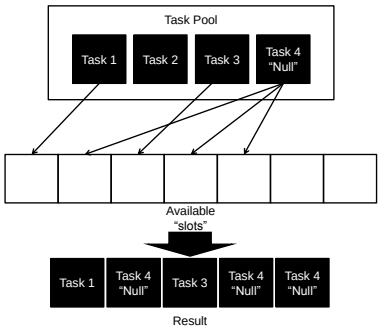


Fig. 1. An illustration of the MILP structure, which assumes a set of tasks $\mathcal{T}$, containing N non-null tasks and one null task, and a finite number of "slots," M, defining the maximum length of any output augmented schedule. Solutions are constructed by matching tasks from the pool with available output slots. The null-task is the only task that can be used more than once.

in the the absence of workload constraints.

$$
\begin{aligned}
&\underset{x_{i,j}\in\{0,1\}}{\text{maximize}} \sum_i \sum_j r_i x_{i,j} - p_\beta \beta - p_\gamma \gamma, \\
&\text{subject to} \sum_{i=1}^{N+1} x_{i,j} \leq 1, && \forall j \in \{1,\dots,M\}, \\
&\qquad \sum_{j=1}^{M} x_{i,j} \leq 1, && \forall i \in \{1,\dots,N\}, \\
&\qquad \sum_{i=1}^{N+1} x_{i,j} s_i \leq B_j, && \forall j \in \{1,\dots,M\}, \\
&\qquad \sum_{i=1}^{N+1} x_{i,j} t_i = C_j - B_j, && \forall j \in \{1,\dots,M\}, \\
&\qquad C_j \leq T_H, && \forall j \in \{1,\dots,M\}, \\
&\qquad 0 \leq B_j - C_{j-1} \leq \zeta, && \forall j \in \{2,\dots,M\}, \\
&\qquad 0 \leq B_1 \leq \zeta, \\
&\qquad W_0 + \sum_i \delta w_1 \, x_{i,1} = W_1, \\
&\qquad W_{j-1} + \sum_i \delta w_i \, x_{i,j} = W_j, && \forall j \in \{2,\dots,M\}, \\
&\qquad W_j - \overline{w} \leq \beta, && \forall j \in \{1,\dots,M\}, \\
&\qquad \underline{w} - W_j \leq \gamma, && \forall j \in \{1,\dots,M\}, \\
&\qquad \beta, \gamma \geq 0.
\end{aligned}
\tag{1}
$$

In (1), the first set of constraints ensures that the solution to the MILP corresponds to a reasonable schedule by ensuring that each time slot, with respect to the sequence order, can contain at most one task. Similarly, the second set of constraints guarantees that each task is only assigned to at most a single location in the sequence, with the exception of the null task. The third through the seventh sets of constraints relate to task availability, start times, and completion times. Here, B_j and C_j denote the start and the completion time, respectively, of the j-th task in the sequence. The third set of constraints ensures that no task is started before time s_i. The fourth set of constraints guarantees that the start and completion times are defined in a reasonable way. The fifth constraint specifies that rewards are only attained for tasks completed within time T_H. The sixth and seventh constraints provide a two-fold contribution. First, the lower bounds specify that a no task can begin before the previous task ends. The upper bounds arise from discretization of operator idle time into intervals of length ζ, and are used to limit the number of "gaps" that are present in the output schedule (Remark 1). The remaining constraints deal with moderating workload. By our incremental definition, the workload level after the j-th task W_j is precisely the workload level W_{j-1} plus the appropriate increment. This is captured by the eighth and ninth sets of constraints. Here, W_0 denotes the (known) initial workload level. Since we have included the null-task in the pool of available tasks, this set of constraints also defines the appropriate change in workload due to idle time. The final sets of constraints require workload levels to remain within the pre-specified bounds, buffered by the decision variables β and γ. The variables β and γ enter into the objective function as penalties for violating the workload bounds, proportional to the parameters $p_\beta, p_\gamma > 0$. Enforcing the workload bounds as soft, rather than strict, constraints, is beneficial for a variety of reasons. First, exact bounds $\underline{w}, \overline{w}$ are usually not known beforehand, and thus enforcement of a strict bound may be ill-advised. Second, the variables p_β and p_γ provide a means of tuning the enforcement of workload bounds. Indeed, higher values of p_β and p_γ indicate larger penalties for bound violations. Finally, soft enforcement of workload bounds ensures feasibility of at least 1 non-degenerate solution, assuming that T_H is sufficiently long. This discussion yields the following result.

Lemma 1 (Feasibility): The MILP (1) is feasible. If there exists $i \in \{1,\dots,N\}$ such that $t_i + s_i < T_H$, then there exists a non-degenerate feasible point, i.e., a point corresponding to an output augmented schedule containing at least 1 task.

Proof: The zero vector is feasible. If there exists i such that $t_i < T_H$, then we construct a decision vector $\hat{x}$ such that $x_{i,1} = 1$, all other binary decision variables are zero, and $\gamma = \beta = C$, where $C > \max\{|W_0 + T_H \delta w_i|, |W_0 - T_H \delta w_0|\}$. Substitution verifies that $\hat{x}$ is feasible. ■

Most formulations of optimal scheduling problems are NP-hard. The MILP (1) presents similar difficulties since it is non-convex and combinatorial in nature; however, effective methods exist for finding high quality solutions, including rounding schemes, branch and bound search strategies, and genetic algorithms [20]. Such heuristics are included in a variety of software packages, including the Matlab optimization toolbox [21] and the cvx software package [22], [23]. In many cases, obtaining reasonable solutions from heuristic solvers is sufficient from a practical standpoint, even though solutions may not be strict global (or even local) optima. The formulation (1) may provide a viable option in such cases.

Remark 1 (Null-Task): Recall that the task pool contains a "null-task" of length ζ. Once a solution to (1) is found and a schedule is extracted, tasks are sequentially executed. Due

to task availability, however, it may be impossible for the operator to start a new task immediately after the previous task is completed. Heuristic solvers may also introduce delays between successive null-tasks. As such, if the time profile of task execution is mapped over T_H, there may be "gaps", i.e., times when no task (even a null-task) is executed. Workload effects due to these gaps are not explicitly considered in (1). However, the sixth and seventh constraints imply that the maximum gap length shrinks to 0 as $\zeta \to 0$, and thus workload effects due to gaps become negligible for small ζ. Of course, shrinking ζ increases required computation, and thus a tradeoff must be considered.

III. Robust Scheduling Strategies

A. Scenario-Based Robust Planning as an MILP

We now focus on designing schemes that are robust to processing time uncertainty. As such, we assume that, for each T_i, the processing time t_i is a random variable, distributed according to a probability density function f_i. We assume for now that each distribution f_i is known with complete certainty. Generally, the choice of appropriate function f_i is task-dependent. However, in many cases, standard distributions, such as log-normal distributions, are effective in the context of collaborative human-UAV missions [24].

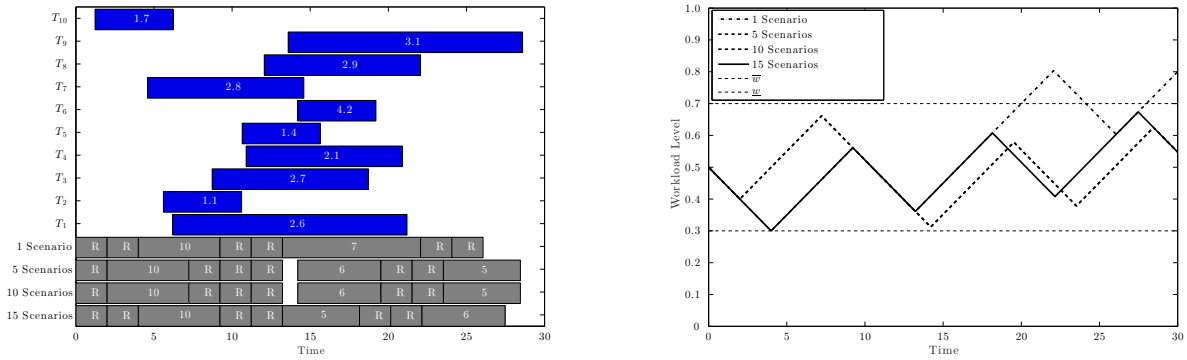
We adopt a scenario-based approach to incorporating uncertainty. In this approach, the processing time distribution of each task is sampled to generate a set of *scenarios*, or candidate processing times. These scenarios are then used to find a solution that remains feasible regardless of the scenario that is chosen to represent the true processing times. If underlying distributions are accurate, then large numbers of scenarios can generate schedules that are robust to particular realizations of the processing time variables. That is, by following the schedule obtained, a wide variety of processing time realizations will still produce rewards that are above the predicted lower bound. Formally, for each T_i, we generate a set of $Q \in \mathbb{N}$ times, $\{t_i^1, t_i^2, \ldots, t_i^Q\}$, where $t_i^m \sim f_i$ for all m. For each $m \in \{1, \ldots, Q\}$, the set $\{t_1^m, \ldots, t_N^m, \zeta\}$ defines a *scenario*, as it represents a possible realization of task processing times (null task included). For generality, we also assume scenario-dependent workload increments and let δw_i^m represent the increment associated with task T_i in the m-th scenario. Given the set of Q scenarios, we expand (1) through the additional requirement that any constraint must be satisfied for *all* of the generated scenarios. To do so, to each m we associate a distinct set of decision variables $\{B_j^m\}, \{C_j^m\}$, and $\{W_j^m\}$, where $j \in \{1, \ldots, m\}$, yet we retain a single set of binary indicators $\{x_{i,j}\}$ to serve all scenarios. Thus, starting times, ending times, and workload values are time or scenario-dependent, while the resulting schedule is independent of these parameters. This allows for the unambiguous extraction of an augmented schedule from the MILP solution, which will then (ideally) be robust to uncertainty in processing times. The robust MILP is expressed in (2). Here, the optimization is over the decision variables α, β, γ, $x_{i,j}$, C_j^m, B_j^m, and W_j^m and it is assumed that $i \in \{1, \ldots, N\}$, $j \in \{1, \ldots, M\}$, and $m \in \{1, \ldots, Q\}$ unless otherwise stated.

$$
\begin{aligned}
\underset{x_{i,j} \in \{0,1\}}{\text{maximize}} \quad & \sum_i \sum_j r_i x_{i,j} - p_\alpha \alpha - p_\beta \beta - p_\gamma \gamma, \\
\text{subject to} \quad & \sum_{i=1}^{N+1} x_{i,j} \leq 1, && \forall j, \\
& \sum_{j=1}^{M} x_{i,j} \leq 1, && \forall i, \\
& \sum_{i=1}^{N+1} x_{i,j} s_i \leq B_j^m, && \forall j, m, \\
& \sum_{i=1}^{N+1} x_{i,j} t_i^m = C_j^m - B_j^m, && \forall j, m, \\
& C_j^m - T_H \leq \alpha, && \forall j, m, \\
& 0 \leq B_j^m - C_{j-1}^m \leq \zeta, && \forall j, m, \\
& 0 \leq B_1^m \leq \zeta, && \forall m, \\
& W_0 + \sum_i \delta w_1^m x_{i,1} = W_1^m, && \forall m, \\
& W_{j-1}^m + \sum_i \delta w_i^m x_{i,j} = W_j^m, && \forall j \neq 1, m \\
& W_j^m - \overline{w} \leq \beta, && \forall j, m, \\
& \underline{w} - W_j^m \leq \gamma, && \forall j, m, \\
& \alpha, \beta, \gamma \geq 0.
\end{aligned}
\tag{2}
$$

In addition to the differences already noted, (2) differs from (1) in that there is an additional decision variable α, which serves as a bound on the amount that task completion times can exceed the horizon T_H. The variable α is factored into the objective function by means of a linear penalty. This "soft" enforcement of the time horizon constraint is advantageous since it provides an additional "tuning" parameter for controlling system robustness. Indeed, by increasing the value of p_α, more penalty is incurred for generating schedules whose completion times are likely to exceed T_H. The parameter α also serves to prevent the optimization from returning degenerate solutions in the case of highly skewed processing time distributions. This discussion readily leads to the following lemma, whose proof is immediate.

Lemma 2: There exists a non-degenerate solution to (2).

Fig. 2 shows a summary of scheduling solutions, calculated via (2), for a sample mission with 10 tasks, $T_H = 30$ (dimensionless units), and log-normally distributed task processing times. Here, we choose $p_\alpha = 10$, $p_\beta = p_\gamma = 15$, $\underline{w} = 0.3$, $\overline{w} = 0.7$, and $W_0 = 0.5$. Fig. 2(a) portrays the problem setup using the bars in the top portion of the figure. For each T_i, the appropriate bar starts at time s_i, and extends for a length corresponding to the expected execution time. The number inside the bar represents the reward r_i. The bars in the lower portion of the plot represent a simulated task execution instance for each scenario condition, based on the solution to (2). The start time and length of the bar represent the time that the task was started and the task duration



(a) (b)

Fig. 2. Simulated task execution using schedules produced via (2) for a sample mission with $N = 10$, $T_H = 30$. Plot (a) shows a pictorial mission summary and the resultant simulated solutions, while (b) shows the corresponding workload evolution (note that the 5 and 10 scenario curves are identical).

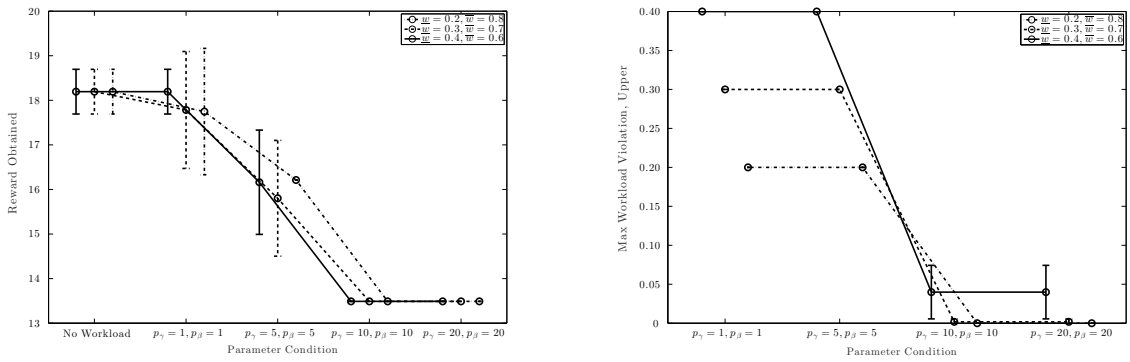
respectively, while the number inside the bar represents the task index, with "R" indicating a null-task. Specifically, task execution was simulated as follows. First, "actual" task execution times were sampled from the appropriate distributions. Then, for each condition, scenarios were generated using Monte-Carlo sampling, and (2) was solved using Matlab's `Intlinprog` function. Given the resultant schedule, task execution was simulated by assuming the operator processes each task sequentially, with each having a duration defined by the "actual" processing time. Each task was started at the earliest possible time (the maximum of the previous task completion time and the availability constraint), and tasks were only included in the lower portion of Fig. 2(a) if its simulated completion time was less than T_H. Note that all task executions satisfy the availability constraints, but their actual duration differs from the expected duration due to uncertainty in processing times. Fig. 2(b) shows the evolution of workload corresponding to the simulated scenario in Fig. 2(a). Here, a linear workload model is considered: if $t \in [C_{j-1}, B_j]$, the workload $W(t)$ at time t is

$$W(t) = \begin{cases} \min\{1, W_{j-1} + 0.05t\} & , \text{ if } T_j \text{ is non-null} \\ \max(0, W_{j-1} - 0.05t\} & , \text{ otherwise.} \end{cases}$$

We choose to constrain workload values to between 0 and 1 both for modeling purposes, and because this normalization arises naturally in common models. For example, if workload were correlated with utilization, i.e., the fraction of recent time that the operator was active, then this normalization is necessary [17]. In *a priori* planning, this cap is not taken into account by the optimization, and thus it is assumed that $\delta w_i^m := \pm 0.05 t_i^m$, where the sign is dependent upon whether the task under consideration is a null-task. Note that in Fig. 2(b), that as the number of scenarios increases, the workload does not exceed the specified bounds.

Fig. 3 illustrates the effects of altering workload bounds on achieved performance. The plot was generated by the following procedure. First, a task set was generated, parameters $T_H = 30$, $W_0 = 0.5$, and $p_\alpha = 10$ were initialized, and a set of 10 scenarios were generated. Then, 10 simulations runs were conducted, where in each run, 1) "actual" task processing times were sampled from the appropriate distributions, 2) a schedule was created using (2) for each set of $\underline{w}, \overline{w}, p_\beta$, and p_γ values ("No Workload" corresponds to $p_\beta = p_\gamma = 0$, i.e., no penalty was assessed for exceeding workload bounds), 3) operator task execution was simulated for each condition using the resulting schedule and "actual" task times in the same manner as that used to generate Fig. 2(a), and 4) the achieved nominal reward was recorded, i.e., the sum of the r_i's for tasks that were executed before time T_H. Fig. 3(a) shows the mean and standard deviation of the rewards obtained for each parameter condition, and Fig. 3(b) shows the maximum amount that workload levels exceeded the upper bound during task execution. The lower workload bound was only violated by a small amount in all conditions, and thus we omit this result. Note that both the obtained reward and workload bound violations decrease as p_β and p_γ are increased, but the reward increases and bound violations decrease as $\overline{w} - \underline{w}$ widens. Even though nominal rewards r_i do not have an explicit dependence on workload levels, the MILP (2) does assesses a linear penalty to workload bound violations. Thus, the formulation (2) can be viewed as a scalarization of a multi-objective optimization, whose goal is to balance nominal reward achieved with workload violations. Therefore, it is implicitly assumed that workload levels either directly or indirectly influence mission success. and thus it may be useful to enforce tighter workload constraints at the expense of lower nominal reward.

Remark 2 (Scenario-Based Robust Optimization): Many schemes exist for solving uncertain optimization problems. In particular, the study of robust optimization provides tools for finding solutions with guaranteed worst-case performance in problems with bounded uncertainty [10]. Scenario-based approaches to handling uncertainty can, in some sense, be viewed as "naive," since they rely on constraint set expansion through random sampling. Despite this, scenario-based approaches function well in a variety of applications due to their simplicity and effectiveness. We adopt such an approach for the following



(a) (b)

Fig. 3. An illustration of the effects of altering the parameters $\underline{w}$, $\overline{w}$, p_β and p_γ for a sample mission involving 10 tasks and a horizon $T_H = 30$. In (a), the nominal reward obtained is plotted as a function of increasing p_β and p_γ, while (b) shows the maximum amount that the upper workload bound was violated for the corresponding experimental conditions. The lower workload bound violations were very small, and thus we omit this result. The absence of standard deviation bars indicates that there was no deviation in the results for the condition in question.

reasons: First, scenario-based optimization is simple and intuitive, making it attractive to many practitioners. Second, most approaches to robust optimization rely on bounded uncertainty sets, or approximations that enforce artificial bounds. Scenario-based approaches do not require such bounds. Third, scenario-based approaches allow designers to easily tune solutions to achieve a desired degree of robustness. Finally, given "regularly" shaped uncertainty distributions, scenario-based approaches usually provide reasonable performance with a modest number of scenarios.

B. Receding Horizon Robust Scheduling

Scenario-based robust scheduling strategies can generate reliable lower bounds on system performance. That is, using a reasonable number of scenarios and assuming that chosen processing time distributions are accurate, the MILP (2) will produce a lower reward bound that will likely hold true in actuality. However, solutions produced by scenario-based optimizations are often very conservative, particularly when uncertain parameter distributions are highly skewed or have high variance. For example, if visual search times are modeled via log-normal distribution, then robust scheduling strategies would most likely be driven by search times occurring in the tail, which are unlikely to occur in actuality. Clearly, following a robust schedule that is calculated *a priori* does not take advantage of all available information during task execution, and performance can potentially be improved by re-planning the mission in an online fashion as realizations of task processing times become known. This observation suggests a receding horizon robust scheduling scheme, which calculates new, robust schedules after each task is processed. Such a scheme is described as follows.

1) collect the tasks to be processed in a set $\mathcal{T}$,
2) formulate and solve the scenario-based robust scheduling problem as in Section III-A,
3) execute the first task in the resultant schedule, and observe the processing time t,
4) if the executed task is a not a null-task: remove it from the set $\mathcal{T}$, subtract t from all remaining task availability constraints, and redefine $T_H = T_H - t$, and
5) repeat steps 2-4 until $\mathcal{T} = \emptyset$ or $T_H \leq 0$.

The receding horizon scheme re-plans each time the uncertainty set is decreased, which, in the robust planning case, will generally lead to better performance. The obvious drawback to this strategy is that it requires an MILP to be solved after each task is executed; thus, total computation time is significantly increased. Therefore, there is a tradeoff between solution quality and computation time that will need to be assessed on a case by case basis.

Remark 3 (Computation Time): Receding horizon robust scheduling schemes may be computationally intensive in their raw form. However, a number of steps can be taken to adjust the schemes to fit them into a desired computational framework. The simplest amendment to the presented algorithm is a reduction of the number of scenarios. This reduces complexity at the expense of robustness. A more sophisticated strategy involves selective re-planning only when a certain criterion is met, as opposed to re-planning after every task. For example, we could choose to re-plan only if the observed execution time is significantly different from any of the processing times predicted by the scenarios. A thorough treatment of such issues is left to future work.

C. Adaptive Receding Horizon Robust Scheduling

Thus far, it has been assumed that task processing time distributions are known exactly. In many applications, this is not the case, and thus it may be necessary to estimate them during the mission. In this section, we illustrate one such estimation scheme that can be implemented in conjunction with the receding horizon approach of Section III-B. Building on the formulation of the previous sections, suppose each task T_i is generated by one of $P \in \mathbb{N}$ sources, e.g., different regions of interest, and each such task can be of $T \in \mathbb{N}$ possible types, e.g., easy or hard. Assuming independent tasks, suppose that associated to each source $p \in \{1, \ldots, P\}$,

there exists an unknown, static probability mass function $g_p : \{1, \ldots, T\} \to [0, 1]$ that captures the likelihood that a task originating from source p is of type ℓ, i.e., the probability of any single task that is generated by source p being of type ℓ is $g_p(\ell)$. Finally, to each type ℓ, we associate an unknown, static probability density function $f_\ell : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, which captures the distribution of possible processing times. The distributions f_ℓ are region independent. With these additions, each task T_i consists of a 6-tuple $T_i = (t_i, s_i, r_i, \delta w_i, \phi_i, \tau_i)$, where s_i, r_i, and δw_i are defined as before, $\phi_i \in \{1, \ldots, P\}$ represents the source that generated the task (known), $\tau_i \sim g_{\phi_i}$ is the task type (unknown), and $t_i \sim f_{\tau_i}$ is the task processing time (unknown).

The underlying distributions associated with both sources and types are unknown to the operator. For each source p and each type ℓ, let $\hat{g}_p$ and $\hat{f}_\ell$ denote estimates of the respective functions. The adaptive receding horizon scheduling approach updates $\hat{g}_p$ and $\hat{f}_\ell$ as the operator processes tasks, and the new estimates are used for subsequent scheduling. The algorithm evolves as follows:

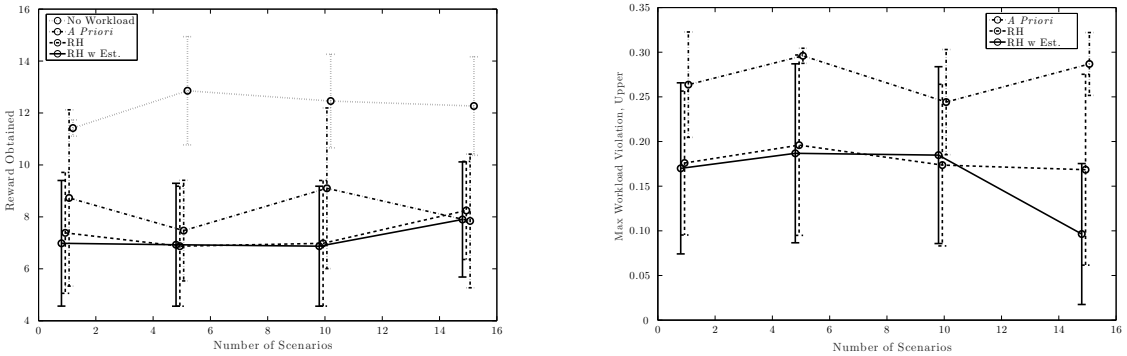
1) collect the tasks to be processed in a set $\mathcal{T}$,
2) generate new scenarios using $\{\hat{g}_p\}$ and $\{\hat{f}_\ell\}$,
3) formulate and solve the scheduling problem (2),
4) execute the first task T_i in the resulting sequence and observe τ_i and t_i,
5) if task T_i is non-null: update the estimates $\hat{g}_{\phi_i}$, $\hat{f}_{\tau_i}$ and remove T_i from $\mathcal{T}$,
6) subtract t_i from the remaining task availability constraints, redefine $T_H = T_H - t_i$, and
7) repeat steps 2-6 until $\mathcal{T} = \emptyset$ or $T_H \leq 0$.

Step 2 in the process above requires explanation. Indeed, the operator does not know the type of any unprocessed task, and thus a decision is necessary about which distributions should be sampled to generate scenarios. Many choices are reasonable, e.g., a "maximum likelihood" method, where each time t_i^m is selected by sampling the distribution $f_{\tau_i^*}$, where $\tau_i^* := \arg\max_\ell \hat{g}_{\phi_i}(\ell)$. That is, each scenario is sampled from the processing time distribution corresponding to the most likely type for task T_i, according to $\hat{g}_{\phi_i}$. We exploit the "maximum likelihood" sampling process in the remaining simulations. With step 2 established, it only remains to clarify the update procedure for the distributions $\hat{g}_p$ and $\hat{f}_\ell$ in step 5. The appropriate update method will generally be governed by the problem assumptions. However, for illustrative purposes, we assume that each element in the set $\{f_\ell\}_{\ell \in \{1,\ldots,T\}} := (\mu_\ell, \Sigma_\ell)$ is log-normally distributed, where μ_ℓ, Σ_ℓ are the standard log-normal distribution parameters. We also assume that prior information is available regarding each distribution g_p and f_ℓ in the form of a set of samples, accumulated prior to the current scheduling mission. Upon completion of task T_i, the distributions $\hat{g}_{\phi_i}$ and $\hat{f}_{\tau_i}$ are then updated by calculating the standard maximum likelihood estimates.

Fig. 4 presents a comparison of various solution methods for a sample mission involving 10 tasks, 2 task sources, and 3 task types. Here, there is uncertainty in task processing times, as well as in each task and type distribution g_p and f_ℓ, respectively. Each distribution g_p was generated randomly, while the set $\{f_\ell := (\mu_\ell, \Sigma_\ell)\}$ was created so that the medians $\{e^{\mu_\ell}\}$ were evenly spaced in $[0, 0.5T_H]$, and $\Sigma_\ell = \Sigma_{\hat{\ell}}$ for all $\ell, \hat{\ell} \in \{1, \ldots, T\}$. We assumed that 10 prior samples were available *a priori* from each distribution g_p, and 5 prior samples were available from each distribution f_ℓ. The prior samples were generated with the appropriate "true" distribution. The distributions $\hat{g}_p$ and $\hat{f}_\ell$ were initialized via standard maximum likelihood estimates. For the adaptive solution method, the estimates $\hat{g}_p$ and $\hat{f}_\ell$ were updated after each task execution. In all cases, scenarios were generated using the "maximum likelihood" scheme and workload evolved as in previous examples. We chose $p_\alpha = 10$, $p_\beta = p_\gamma = 15$, $W_0 = 0.5$, $\underline{w} = 0.3$, $\overline{w} = 0.7$, and $T_H = 30$. Fig. 4(a) shows the rewards obtained, i.e., the sum of the rewards r_i of executed tasks, averaged over 10 simulation runs. The same setup and prior information was used in each run, but each had different realizations of "actual" processing times. For each run, identical scenarios were used across experimental conditions, i.e., solution methodologies. Fig. 4(b) shows the maximum upper workload bound violation for each run (the lower bound was rarely violated, so we omit this case). Large variances in both plots is due to the high uncertainty in the underlying setup. As expected, the no workload condition produced significantly higher rewards than other methods. The methods considering workload performed similarly to one another in terms of nominal reward, particularly in the 15 scenario case; however, the *a priori* method performed much poorer than the other robust schemes in terms of workload bound violation. The adaptive scheme did not quantitatively differentiate from the non-adaptive receding horizon scheme in terms of either metric. However, a qualitative downward trend began to emerge in workload bound violation (and bound violation variance) for the adaptive scheme as the number of scenarios increased. We hypothesize that this trend would become more pronounced in larger problems where more estimation update steps are available, and thus the adaptive scheme may be superior (in a multi-objective sense) to the non-adaptive scheme in these cases. We leave further investigation of this claim to future work.

IV. Conclusion

For human-UAV collaborative missions, it is crucial to develop task scheduling methodologies that 1) consider human cognitive requirements, 2) are robust to uncertainty, 3) can incorporate various design goals, and 4) are practically implementable. The presented MILP framework can potentially accomplish all of these goals. Indeed, this scheme can incorporate cognitive workload, introduce robustness, and expand to handle additional layers of complexity, while still staying within a straightforward optimization framework. Future work should focus on verifying the utility of these scheduling frameworks through human subjects experiments. In particular, these studies should seek to 1) verify that performance actually does, in fact, improve with the introduction of such scheduling systems, 2) investigate whether simple cognitive workload models are sufficient



(a) (b)

Fig. 4. A comparison of performance of different solution methodologies in a sample mission with 10 tasks,$T_H = 30$ (dimensionless units), and uncertain processing times, as well as uncertain source and processing time distributions. In (a), the obtained nominal reward is plotted as a function of the scenarios used. In (b), the maximum amount that the workload level exceeded the bound $\overline{w}$ is plotted, again as a function of the number of scenarios.

for these applications or if more elaborate models should be considered, and 3) identify any unforeseen practical or performance issues. Additional theoretical work can then be done to incorporate further layers of complexity and tailor system designs to human-UAV applications.

ACKNOWLEDGEMENT

This paper is based on work supported by the US Air Force Research Lab (AFRL), Air Force Office of Scientific Research, under contract FA9550-14-C-0022. In addition, this paper is based on work sponsored by the U.S. Army Research Office and the Regents of the University of California, through Contract Number W911NF-09-D-0001 for the Institute for Collaborative Biotechnologies, and that the content of the information does not necessarily reflect the position or the policy of the Government or the Regents of the University of California, and no official endorsement should be inferred.

REFERENCES

[1] M. Hoffman, "New reaper sensors offer a bigger picture," *Air force times*, vol. 19, 2009.

[2] M. L. Cummings and P. Mitchell, "Automated scheduling decision support for supervisory control of multiple UAVs," *Journal of Aerospace Computing, Information, and Communication*, vol. 3, no. 6, pp. 294–308, 2006.

[3] E. Lawler, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys, "Sequencing and scheduling: Algorithms and complexity," *Handbooks in operations research and management science*, vol. 4, pp. 445–522, 1993.

[4] T. Yamada and R. Nakano, "Job shop scheduling," *IEE control Engineering series*, pp. 134–134, 1997.

[5] B. Alidaee and H. Li, "Parallel machine selection and job scheduling to minimize sum of machine holding cost, total machine time costs, and total tardiness costs," *Automation Science and Engineering, IEEE Transactions on*, vol. 11, no. 1, pp. 294–301, 2014.

[6] S. Mason and K. Oey, "Scheduling complex job shops using disjunctive graphs: A cycle elimination procedure," *International Journal of Production Research*, vol. 41, no. 5, pp. 981–994, 2003.

[7] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms-I. Representation," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 983–997, 1996.

[8] S. Lawrence and E. Sewell, "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain," *Journal of Operations Management*, vol. 15, no. 1, pp. 71–82, 1997.

[9] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.

[10] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton University Press, 2009.

[11] C. Wickens, J. Hollands, S. Banbury, and R. Parasuraman, *Engineering Psychology & Human Performance*. Psychology Press, 2015.

[12] M. Cummings and P. Mitchell, "Operator scheduling strategies in supervisory control of multiple UAVs," *Aerospace Science and Technology*, vol. 11, no. 4, pp. 339–348, 2007.

[13] L. F. Bertuccelli and M. L. Cummings, "Scenario-based robust scheduling for collaborative human-UAV visual search tasks," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 5702–5707.

[14] J. Peters, V. Srivastava, G. Taylor, A. Surana, M. P. Eckstein, and F. Bullo, "Mixed human-robot team surveillance: Integrating cognitive modeling with engineering design," *IEEE Control Systems Magazine*, vol. 35, no. 6, pp. 57–80, 2015.

[15] D. Gopher and E. Donchin, "Workload: An examination of the concept," in *Handbook of Perception and Human Performance*, K. R. Boff, L. Kaufman, and J. P. Thomas, Eds. John Wiley & Sons, 1986, vol. 2, pp. 1–49.

[16] K. H. Teigen, "Yerkes-dodson: A law for all seasons," *Theory & Psychology*, vol. 4, no. 4, pp. 525–547, 1994.

[17] K. Savla and E. Frazzoli, "A dynamical queue approach to intelligent task management for human operators," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 672–686, 2012.

[18] G. Borghini, L. Astolfi, G. Vecchiato, D. Mattia, and F. Babiloni, "Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload, fatigue and drowsiness," *Neuroscience & Biobehavioral Reviews*, vol. 44, pp. 58–75, 2014.

[19] Y.-F. Tsai, E. Viirre, C. Strychacz, B. Chase, and T.-P. Jung, "Task performance and eye activity: Predicting behavior relating to cognitive workload," *Aviation, Space, and Environmental Medicine*, vol. 78, no. Supplement 1, pp. B176–B185, 2007.

[20] D. Bertsimas and R. Weismantel, *Optimization over integers*. Dynamic Ideas Belmont, 2005, vol. 13.

[21] MathWorks, "Linear programming and mixed-integer linear programming," http://www.mathworks.com/help/optim/linear-programming-and-mixed-integer-linear-programming.html, Accessed: 2015-08-29.

[22] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds. Springer-Verlag Limited, 2008, pp. 95–110, http://stanford.edu/~boyd/graph_dcp.html.

[23] ——, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[24] D. Southern, "Human-guided management of collaborating unmanned vehicles in degraded communication environments," DTIC Document, Tech. Rep., 2010.