Determining the Value of Information for Collaborative Multi-Agent Planning

Determining the Value of Information for Collaborative Multi-Agent Planning

David Sarne
Computer Science Department,
Bar-Ilan University, Ramat-Gan 52900 Israel
sarned@cs.biu.ac.il

Barbara J. Grosz
School of Engineering and Applied Sciences,
Harvard University, Cambridge MA 02138 USA
grosz@eecs.harvard.edu

Abstract

This paper addresses the problem of computing the value of information in settings in which the people using an autonomous-agent system have access to information not directly available to the system itself. To know whether to interrupt a user for this information, the agent needs to determine its value. The fact that the agent typically does not know the exact information the user has and so must evaluate several alternative possibilities significantly increases the complexity of the value-of-information calculation. The paper addresses this problem as it arises in multi-agent task planning and scheduling with architectures in which information about the task schedule resides in a separate "scheduler" module. For such systems, calculating the value to overall agent performance of potential new information requires that the system component that interacts with the user query the scheduler. The cost of this querying and inter-module communication itself substantially affects system performance and must be taken into account. The paper provides a decision-theoretic algorithm for determining the value of information the system might acquire, query-reduction methods that decrease the number of queries the algorithm makes to the scheduler, and methods for ordering the queries to enable faster decision-making. These methods were evaluated in the context of a collaborative interface for an automated scheduling agent. Experimental results demonstrate the significant decrease achieved by using the query-reduction methods in the number of queries needed for reasoning about the value of information. They also show the ordering methods substantially increase the rate of value accumulation, enabling faster determination of whether to interrupt the user.

1 Introduction

Advances in autonomous-agent capabilities have the potential to significantly increase the power of multi-agent planning and scheduling systems [4, 54]. Such capabilities could be especially useful in highly dynamic environments in which the schedule of multiple participants must be coordinated, the individuals are separated geographically or there is limited communications bandwidth [59, 57, 32]. These characteristics arise, often all together, in a range of important application settings, including first-response [58, 50], planetary exploration, cleanup of hazardous sites and military conflicts [33]. The complex, stochastic nature of changes in these settings and the pace at which they happen make it unlikely that any single individual or agent could have a complete global view of a scheduling problem [48, 61]. As a result, a combination of localized reasoning and group coordination mechanisms are required for planning and scheduling decisions [53, 58, 4]. Autonomous agents that operate as a team, suggesting alternative courses of action to each other and negotiating to find a solution, could help people operating in such environments to achieve team objectives more effectively [27, 49, 43].

In this paper, we use the term "fast-paced environment" to refer to a multi-agent task environment that changes rapidly so that task schedules are frequently, often continuously, being revised; "coordination management system" to refer to a system in which a set of autonomous agents supports the coordinated scheduling of a group of people, who may be distributed geographically, working together to accomplish a shared task; "automated scheduling agent" (ASA) to refer to the individual autonomous agents that this system comprises; and "owner" to refer to the person whom the ASA directly assists (e.g., the leader of some group). The overarching goal to which this paper contributes is that of building ASAs able to coordinate modifications to their owners' task schedules as the environment changes, thus enabling the people carrying out tasks to focus on actual task performance [61, 5].

Figure 1 illustrates the key characteristics of a coordination management system operating in a fast-paced environment. It shows several different types of teams — firefighters, policemen and medical personnel — working together. Each ASA manages the schedule of one of these teams, with the owner being, for example, the team leader of the first-response team or the unit commander of the firefighters. The ASAs' responsibility is limited to scheduling tasks. The tasks themselves are executed by people, either an owner or one of the units overseen by an owner. As shown by the dotted boxes in the figure, each ASA has full visibility of its own team's schedule, which it manages. It has only partial visibility of the other teams' schedules, as indicated by gaps and question marks. To support team coordination, the

ASAs cooperate, and their goal is to maximize some general objective function. For example, medical staff may be partially synchronized with the schedule and planned activities of firefighters and vice versa, and their shared objective function might be successfully evacuating all injured from the arena.

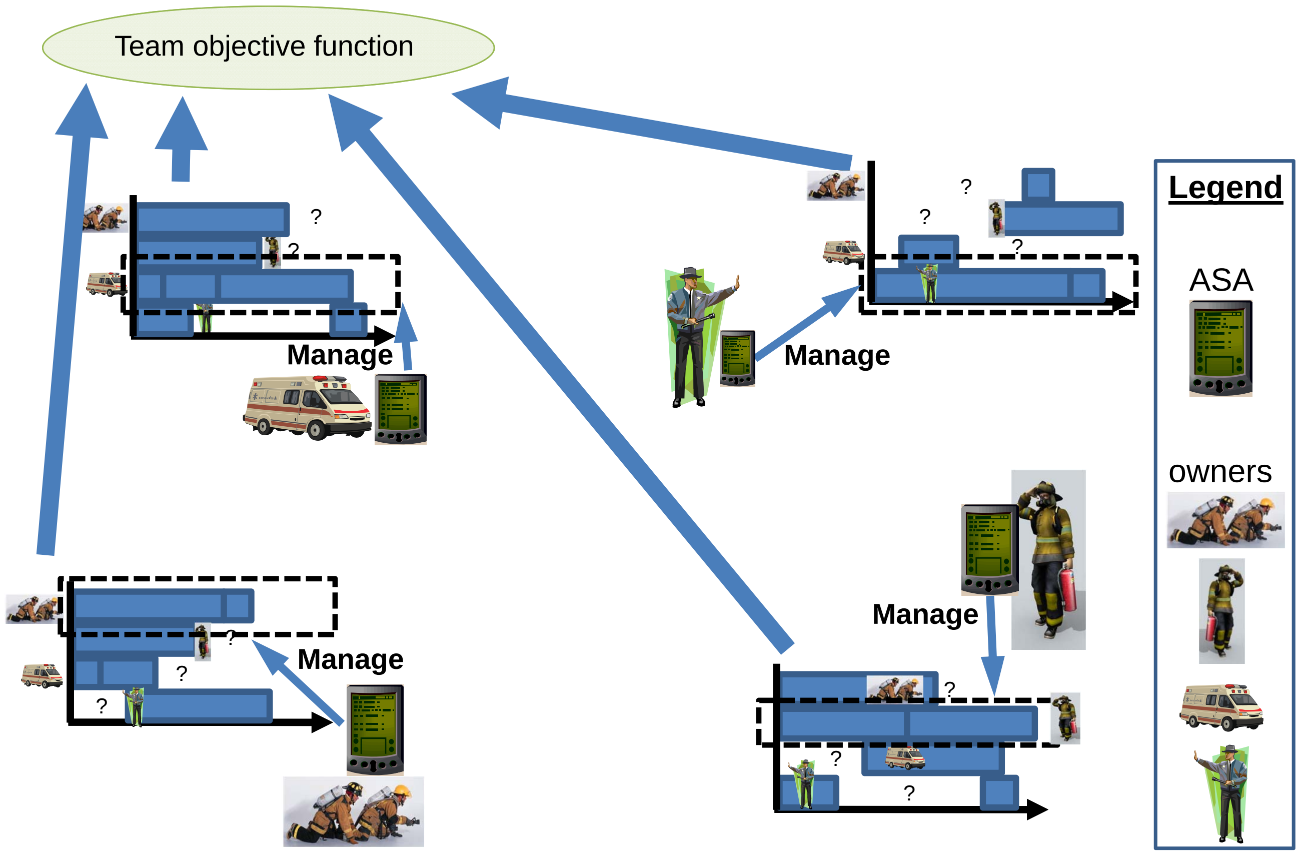


Figure 1: A 4-team rescue effort coordination scenario.

As this example illustrates, in coordination management domains, scheduling information and constraints are distributed, and each ASA has a different view of the tasks and structures that constitute the full multi-agent problem. Each ASA needs to reason about changes in the timing or outcome of tasks, not only for its owner's team's tasks, but also in light of potential effects on other ASAs' owners' tasks. In fast-paced domains, there may be severe time pressures, and typically ASAs must make decisions in real time, concurrently with their owners' (or their owners' units') execution of tasks.

The knowledge that ASAs have of their team's performance as they execute tasks and the environment in which they are working significantly influences the quality of the schedules a coordination management system produces. Such information as changes in physical surroundings and the status of the different people or other agents working on a distributed, coordinated task, may affect scheduling constraints, either relaxing them or imposing new ones that better reflect the actual situation. Much of this kind of information may be outside the purview of an ASA, i.e., beyond what an ASA can sense on its own. Importantly, the people whose work ASAs are helping to coordinate may have more timely, accurate information about the state of the environment than their ASAs. For example, a human driver can see changes in weather conditions that affect route selection as they occur, while current automated navigation systems cannot. A scientist may be able to induce that a colleague, whose children go to the same school as his, will be unavailable to meet, because there is a school play at the proposed meeting time. Although such information may eventually be revealed to the system (e.g., when the scientist's scheduling assistant finds out that the colleague is unavailable for the meeting), the ASA does not have the same immediate access to it as its owner.

If an ASA could obtain new, task-execution-relevant information without incurring any costs on its or its owner's part, it would always choose to get the information as soon as possible. Typically, however, there are costs to gaining such information. In particular, to ask for information, the ASA needs to interrupt the owner. Interruptions are disruptive in nature, degrading performance [12, 19, 22, 20, 61], and both parties incur interaction-related resource costs (e.g., communication costs, computational resources). As a consequence of such costs, the interruptions required to obtain more current information from an owner must be managed appropriately; they cannot simply be triggered automatically or they will overburden people [50, 48, 10, 7]. The research in this paper supports the development of a collaborative interface (CI) for an ASA that queries the owner only if the net expected benefit of the interaction is positive, i.e., if the expected value of the information it would request multiplied by the expected probability that the owner has such information is greater than the cost of the interruption [18, 11]. It addresses in particular the problem of effectively computing the value of information an owner may have.[1]

Determining the value of information requires scheduling and task knowledge, because information is valuable only to the extent it influences schedule changes. In many contexts in which ASAs operate, such task and scheduling knowledge resides in a scheduler module which is external to the CI [5, 42, 53]. This separation of the scheduler in the systems' architecture enables other ASA modules to use the scheduler for reasoning about task schedules. It also follows common modular system design principles for sharing functionality that is needed by multiple modules rather than duplicating it in each module. As a result of this architecture, however, the CI needs to query the scheduler to assess the impact of information on a schedule, as illustrated in Figure 2. For instance, to determine the changes in the optimal schedule that result from changes in the duration of a task that the owner is currently executing, the CI must query

[1]Related work has addressed the calculation of the costs of interruption and determining the probability that an owner has the necessary information, both of which depend on the owner's state [45, 11, 12, 52].

the scheduler for a new schedule which merges the new task duration into the current world state. To avoid degrading overall system performance, the CI must consider the resource demands it makes on the scheduler. A key challenge to ASA design is thus to develop methods that support the calculation of value of information while consuming as few scheduler resources as possible.

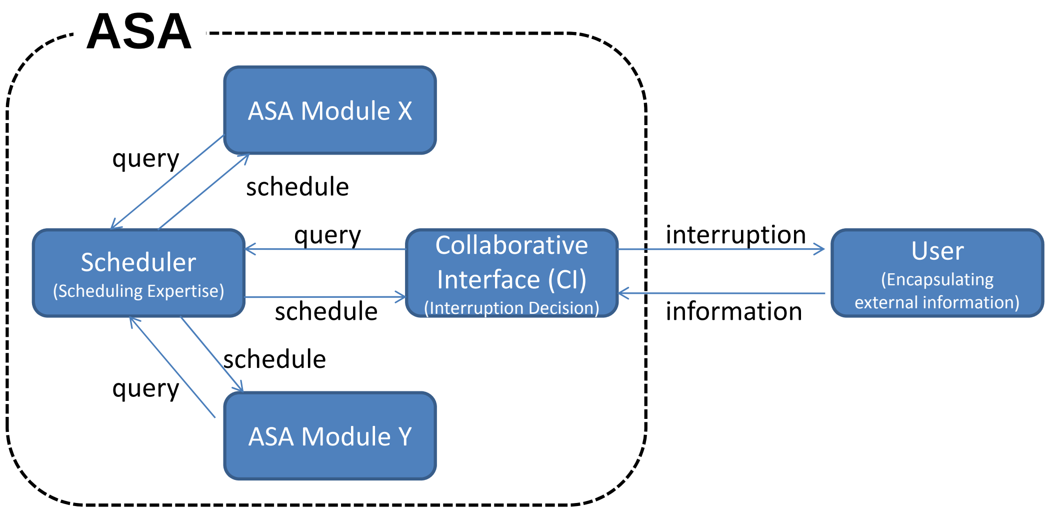


Figure 2: CI's Relationship with the owner and the scheduler.

We note that this kind of reasoning challenge is not restricted to scheduling systems or even to multi-agent systems. It arises in any situation in which the determination of the value of information requires the help of a separate entity (system or person) that is able to evaluate the effect of different possibilities on a given decision in a way the decision-maker cannot do. For instance, it would arise for a patient who needs to decide whether to have an elective biopsy requiring complex surgery. Different biopsy findings will affect the proposed treatment and thus the expected benefit to the patient. To make a decision, the patient needs, in effect, to query the physician for each possible outcome (and its probability) and the treatment associated with it.

This paper makes three major contributions. First, it presents a decision-theoretic algorithm for determining the value of information that may be obtained from an external source, in this case, the ASA's owner. The approach addresses the challenge of calculating this value when doing so depends on invoking, repeatedly, a separate system module. Second, it describes two novel methods for dealing with computational resource constraints on querying the scheduler. The goal of these two methods is to enable the CI to activate the scheduler more efficiently, by decreasing the number of queries required for calculating the exact value of a specific piece of information the owner may have. Third, it describes two methods that enable the CI to reduce the number of queries when it needs to determine only whether the value of information necessarily exceeds (or necessarily is less than) the cost of obtaining the information.

The algorithm generalizes to other situations in which the computation of information value requires that a separate module be invoked to evaluate new situations. The other methods generalize to situations in which some regularity exists in the way an increase or a decrease in the value of different outcome characteristics affects whether knowing the outcome has a non-zero value. In particular, although the paper addresses the problem of calculating value of information in a multi-agent setting, the methods apply as well to single-agent scheduling when an agent is responsible for multiple, linked threads of activity. For instance, they would be useful if resources rather than people were being scheduled. The greater the parallelism and interdependency, the more likely information about one task will affect some part of the schedule of other tasks and the greater number of tasks that may be affected. The challenges are thus highlighted in the multi-agent case.

In the next section, we provide an overview of the ASA decision-making environment, introducing terminology we will use throughout the paper. Section 3 presents the principles underlying the computation of the value of information. Section 4 describes mechanisms for improving the efficiency of the scheduler-querying process when the CI needs the exact value of information and also for settings in which the CI needs to calculate only whether the information value exceeds a given threshold. Experimental results for all of these methods are given in Section 5, which also briefly describes the Coordinators application domain in which they were tested and the basic architecture of the CI module. Section 6 discusses related work and Section 7 concludes.

2 The Basic Decision Making Setting

For exposition purposes, we will use as an example an expanded description of the first-response-type effort of fighting a fire. Multiple fire companies arrive at the scene, each with its own set of tasks. These different tasks are interleaved as part of what must be a highly coordinated team effort to control the fire. For instance, the schedule of the Engine (pumper) arriving first would include stretching a fire hose called the “attack line” to the appropriate entrance of the structure in preparation for making an attack on the fire. This task would need to be coordinated with the main task of the first ladder-truck that arrives at the scene, which has responsibility for ventilation of the structure that is burning. The tasks scheduled to be executed by the different companies vary highly in their level of complexity. Many can be decomposed into such actions as choosing a firefighting strategy and managing the fire streams while coordinating the ventilation efforts, until the fire is suppressed. These actions might further decompose to subtasks which involve several companies (e.g., advancing hoselines and evaluating interior conditions) and can be performed in many ways, each associated with different outcomes. For example, the ventilation task, which releases heat and smoke, can be performed in the form of vertical ventilation or positive pressure ventilation. The first is labor intensive and is therefore likely to be time-consuming. The second is less labor intensive and should be quick to initiate, but it requires substantially more resources. Other tasks, e.g., stretching a fire hose, are not further divisible.

The success of the fire attack crew entering the building depends on the success of the ventilation effort. In some cases, only a successful ventilation of the structure will enable the entry of the fire attack crew; otherwise, a different firefighting strategy will need to be chosen. Similarly, a standpipe stretch from a stairway entrance to the floor on fire requires the availability of three firefighters using three lengths of 2½" hose. As these examples illustrate, the quality of performance of a task typically depends on the quality of performance of other tasks. Teams' plans constantly change. For example, the firefighters' current plan might be interior attack — inserting a team of firefighters into the burning structure, in an attempt to extinguish a blaze from inside the structure, minimizing property damage from fire, smoke and water. This approach requires a minimum of four fully-equipped firefighters: an entry team of at least two to enter the structure and fight the fire, and two standing by to rescue or relieve the entry team. If the entry team cannot extinguish the blaze, the plan may change to an "exterior attack", a method of extinguishing a fire which does not involve entering the structure.

As is typical in AI planning systems, we represent tasks in a hierarchical task structure, with each task decomposing into subtasks. Because there may be different ways to perform a task, the representation allows for different decompositions of a task. For presentation purposes, we will subsequently refer to the overall group task (e.g., getting control of the fire) as the *group activity*. We distinguish tasks that cannot be further decomposed from tasks that further decompose, using the term *atomic task* for the former and *complex task* for the latter. We note that atomic tasks may encompass efforts by multiple agents and hence are not "basic level" in the sense of executable at will by an individual. They are, however, not decomposed into constituents in the domain model. In the remainder of the paper, the unmodified term "task" will be used only to refer to atomic tasks, because the paper focuses mostly on reasoning about such tasks. In addition to decomposition, the task representation includes the inter-task relations "enables", "disables", "facilitates" and "hinders". For instance, it would use "enable" to represent the dependence of the success of the fire attack crew entering the building on the success of the ventilation effort.

The representation of atomic tasks includes several properties essential for ASA scheduling and performance calculations. In particular, the task model associates with each task temporal properties (in particular "duration") and "quality characteristics" that capture such aspects of performance as cost, resources consumed and benefit. In dynamic environments, the plan for a group activity or any of its constituent subtasks may or may not succeed, and the outcome of different ways of doing a particular task will vary in quality (e.g., consume more or fewer resources, produce a better product). For example, advancing a line from the stairwell might have different effectiveness and duration, depending on whether or not the hallway is smoke-filled.

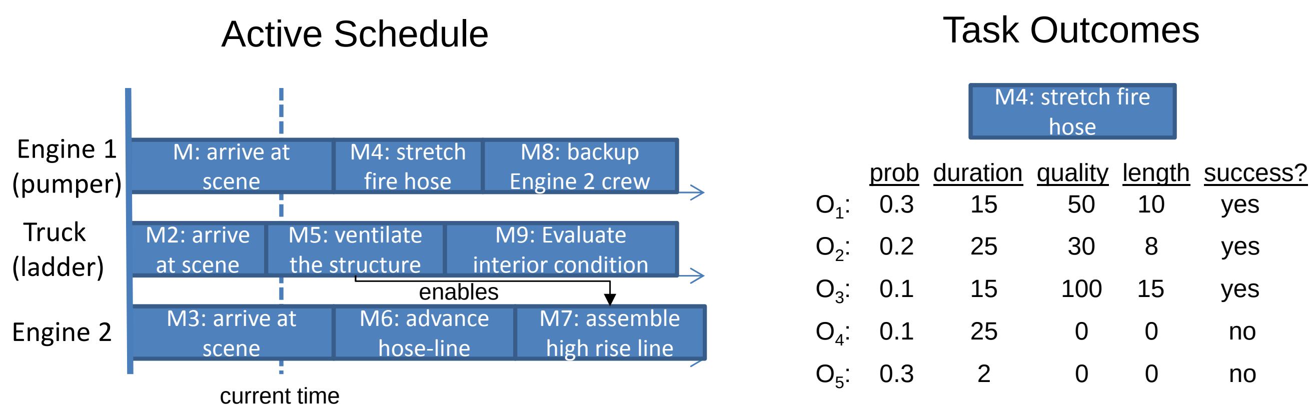
As is common in scheduling systems for dynamic environments [15, 43, 54, 5, 2], the ASA maintains an *active schedule* for the team which reflects their current plan. Figure 3 gives an example of an active schedule for the fire-fighting task.[2] When task outcomes occur that differ from those represented in the active schedule, ASAs must adjust the schedule to reflect this changed reality. To adjust schedules appropriately, ASAs need to know when tasks have been successfully carried out ("task completion") or not ("task failure") regardless of whether the task is directly assigned to their owner or not. They also need to know a range of outcome characteristics of task execution, including actual execution duration and any quality properties relevant to assessing team performance or affecting the execution of other tasks as part of the group activity. As ASAs have scheduling capabilities, but not sensors, and they are involved only in scheduling and not task execution, the only way they can come to know that a task has been completed (which is essential for them to be able to provide scheduling assistance) is for owners to provide this information. Thus, we assume the overall operating environment for ASAs is one in which owners recognize and agree to the requirement that they provide such task-completion information when a task is finished. (For instance, this reporting requirement is standard operating procedure in such domains as first responders, where each unit reports execution both for synchronization and control purposes.) Successful task completion is modeled by the issuing of an *execution completion (EC)* message. If an ASA does not receive such an execution completion indication immediately after one of the possible duration-outcome times has elapsed (which is an implicit indicator that this duration is no longer possible), then it must update the probability of the remaining potential outcomes of the task, increasing them proportionally.

During task execution, an ASA may also communicate with its owner to obtain new task-related information. By obtaining more accurate information about a task's potential outcomes prior to task completion, an ASA decreases its uncertainty; it is able to refine the probabilities associated with different possible task outcomes sooner. As in related work that deals with obtaining similar schedule-related information from users [49], this paper assumes that the information owners provide is reliable, e.g., that they provide the actual outcome. At the end of Section 3, we briefly describe the possibility of handling situations in which the owner can provide only partial information and is unable to provide the actual outcome.

We can now formally define a scheduling problem T.[3] T comprises a set of tasks applicable to

[2]The examples given in this section and the following one are simplified and given for illustrative purposes.

[3]For convenience, we have added a table at the end of the paper summarizing the notation used in this paper.



	prob	duration	quality	length	success?
O_1:	0.3	15	50	10	yes
O_2:	0.2	25	30	8	yes
O_3:	0.1	15	100	15	yes
O_4:	0.1	25	0	0	no
O_5:	0.3	2	0	0	no

Figure 3: Active schedule and possible task outcomes examples.

the problem domain, relationships among those tasks, outcome values for each task, quality accumulation methods and an active schedule. Any task $M \in T$ has multiple possible outcomes. Each possible outcome $o \in M$ has an a priori occurrence probability as estimated by the ASA, $P(o)$, where $\sum_{o_i \in M} P(o_i) = 1, \forall M \in T$. Outcomes are defined by the values they assign to a set of outcome characteristics, each representing a different aspect of task performance. The particular characteristics captured in a task's outcome, as well as their values, may vary by application and even within different tasks in the same application. For example, the right side of Figure 3 illustrates outcomes for the task of stretching a fire hose that is in the fire-fighting active schedule to the left. Each outcome is characterized by its duration, the depth of penetration into the building as achieved by the team with the hose (denoted by "length"), and by whether or not the task executed successfully or not. The outcome of the task of the Engine arriving at the scene, on the other hand, may be characterized by different parameters, such as its duration, the company size and the type of equipment brought with it.

The active schedule, denoted S, includes the tasks of T that are planned to be executed.[4] A task $M \in T$ that is not in the active schedule is an alternative that can be used if a scheduling revision is required. As shown in Figure 3, the active schedule includes tasks that have already been executed (e.g., arrival of the ladder Truck). They are included because these tasks impose scheduling constraints as a result of their relationships with other tasks (e.g., enable) and the way they contribute to overall task performance.

Schedule S uses only a hypothetical central value (usually the mean) for each outcome characteristic of a task, because schedulers cannot handle the scheduling dynamics of the exponentially large set of

[4]Even if the problem is over-constrained, the scheduler can potentially produce a schedule, which in the worst case will be associated with zero quality, with the expectation that the problem will evolve (e.g., it will receive some new information). In such cases, receiving the actual outcome of tasks ahead of time may be extremely useful.

possible outcomes combinations [56]. (Their architectures typically require a single distinct outcome for each task [15, 56, 35, 53].) During the performance of a group activity, the quality of its active schedule, henceforth "quality of the schedule", is derived from the quality properties of its constituent tasks (actual quality if the task has been executed, else hypothetical value) using quality accumulation functions. This schedule quality needs to be recalculated whenever the actual outcome of a task is revealed or the schedule is revised.

This formal setting enables us to state more succinctly the two key challenges addressed in this paper: Determining the value of knowing the actual outcome o_a of a task M, if this information is received at a time t that is prior to completion of M's execution, given a scheduling problem T, the set of possible outcomes O and the a priori outcome distribution $P(o_i)$; and designing efficient techniques for reasoning about that value in settings in which an external scheduler must be queried.

3 Calculating Information Value

This section describes the general framework for reasoning about the value of information that is known by an owner but not his/her ASA. It then describes principles which form the basis for a recursive algorithm that computes the value of such information. The algorithm follows the canonical information-theoretic approach according to which the value of a given piece of information is defined as the difference in expected value between the system performance with and without this piece of information [21]. As the previous two sections have explained, however, the problem environment in which the CI calculates the value of information (VOI) and the system context in which it operates have characteristics that lead to challenges not addressed in prior work. First, the algorithm must iterate through all possible values (e.g., all possible durations), because the CI does not know the actual value the owner will provide (e.g., that the action will take longer than some possible durations in the model). Second, to compute the system's performance without this piece of information requires, in essence, that the CI consider all possible futures allowed by the distribution of task outcomes, i.e, reason hypothetically about schedule changes that would ensue when it subsequently learns that some possible outcome is not the actual outcome. This process of reasoning about possible "eliminated outcomes" and other dynamically changing properties of the schedule could consume significant scheduler resources if done in the obvious, most straightforward manner.

The algorithm described in this section addresses these challenges, providing effective computation of the value of information through intelligently interleaving the calculation of possible schedules with and without information from the user in a single iterative sequence of queries. Within each iterative step, the value of all outcomes associated with the same duration is calculated, and the additional scheduling constraints that are required for calculations in the next step are generated. The algorithm thus avoids unnecessarily recalculating schedule segments that are in common for different potential outcomes. Furthermore, as described in Section 4, the two query-reduction methods we define, which work by identifying in advance outcomes associated with zero contribution to the value of information, significantly reduce the number of queries to the scheduler.

As discussed in Section 6, these techniques are novel both with respect to work in AI on interruption in human-computer interaction and adjustable autonomy and with respect to prior approaches to value-of-information calculation in a range of other settings. While the idea of improving the value-of-information computation by identifying redundant calculations is not new [63, 6], the improvements achieved in prior work depend on properties unique to the domain of application or to connections among data elements and do not address the temporal reasoning challenges that ASAs face.

3.1 The Value of Information from an External Source

The value of obtaining the actual outcome of task M at time t is the difference between the expected performance of the team if it continues to use the current active schedule and its expected performance if it revises the schedule at time t based on the actual outcome of M [21]. A schedule based on the actual outcome of M is less uncertain than one based only on a distribution of possibilities, making the schedule going forward with this new information of higher quality than the schedule without it. As a result, the expected value of obtaining the actual outcome is necessarily non-negative. Furthermore, the earlier the ASA knows the actual outcome of a task, the greater the usefulness of this information, as it provides more flexibility in scheduling revisions.

For instance, if an owner knows that the task currently executing will definitely fail, early notification of the ASA will allow it to immediately generate an alternative active schedule. In the example in Figure 3, if the Truck team owner estimates that efforts to ventilate the structure will fail and conveys this information to the ASA, the schedule can be revised to include an external attack rather than the planned interior attack. Furthermore, if this owner supplies this information before the start of execution of the task, then the ASA can change the schedule in a way that entirely avoids executing the task. This information may also be of benefit to other ASAs, if their owners' tasks either are part of a common complex task or are enabled by this task (e.g., Engine 2 in Figure 3). If the ASA notifies other owners' ASAs of this change, then the schedules of any owners whose tasks are affected by this task's outcome can also be changed earlier, rather than waiting for the failure to occur. Without such advance notice, the ASAs may lose flexibility in choosing alternative schedules as well as valuable task-execution time. The calculation of the expected value of external information therefore requires determining the influence of that information on the entire team's schedule.

The CI does not know what information the owner has, only that the owner has (or may have) relevant information. It thus must compute the value of information without knowing the actual value the owner will provide. For instance, in the firefighting example, the system might benefit from knowing the arrival time of a backup Engine, and it may know that the owner knows this information, but it does not know (for instance) that the owner knows it will arrive in five minutes. Unlike in prior work in AI, which has depended on having in-hand the piece of information Ψ which it is calculating the value of (e.g., Ψ is 3pm at the theater) [19, 22, 20, 61], the CI must iterate through all possible values of Ψ. This iterative process is complicated by the fact that for each possible outcome, multiple queries must be sent to the scheduler. While some tasks may be associated with a small set of outcomes (e.g., arrival time depends only on whether traffic is mild, heavy or jammed), others may have numerous possible outcomes. For example, the task of assembling a high-rise 2½" line (the standard hose line in most fire departments) can have numerous duration and quality outcomes due to delays encountered while carrying different sections, flaking the line or when connecting the sections, or inefficiencies of the team in attaching the nozzle and securing it.

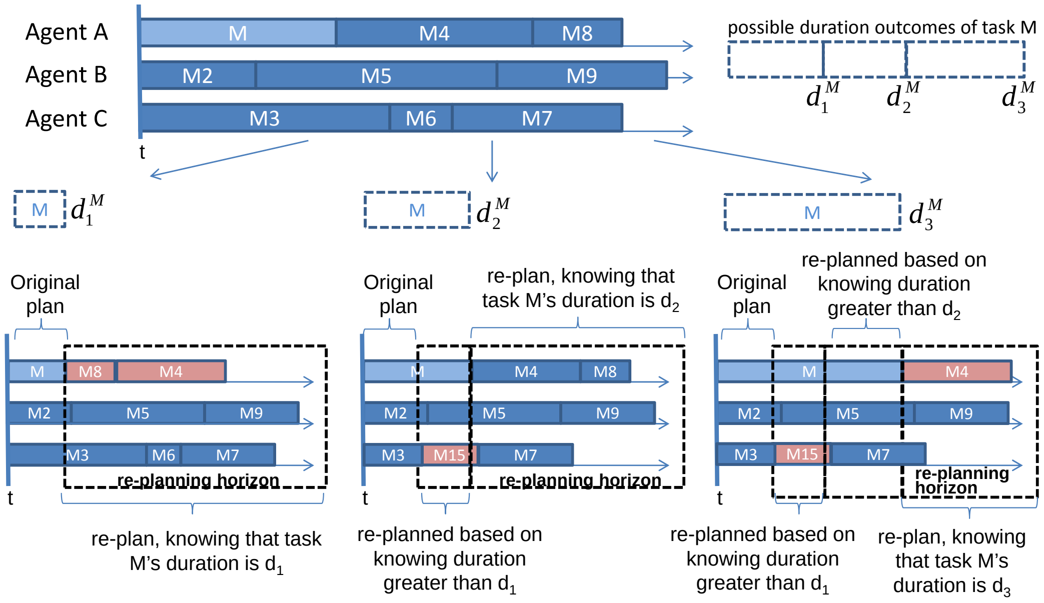

Agent A
Agent B
Agent C
M
M4
M8
M2
M5
M9
M3
M6
M7
t
possible duration outcomes of task M
d_1^M
d_2^M
d_3^M
Original plan
re-planning horizon
re-plan, knowing that task M's duration is d_1
re-plan, knowing that task M's duration is d_2
re-planned based on knowing duration greater than d_1
re-planned based on knowing duration greater than d_2
re-plan, knowing that task M's duration is d_3
M15

Figure 4: Execution-completion-based rescheduling.

The calculation is further complicated by the fact that the active schedule is continuously updated as a result of the receipt or absence of execution completion messages over time. These execution-completion-based reschedulings may significantly alter the actual team's performance in executing the group activity, and consequently calculating the benefit of receiving the information becomes more complex.

This process is demonstrated in Figure 4, which is based on the active schedule in Figure 3. The upper schedule represents the active schedule of three ASA owners (represented as agents A, B and C). The task for which information is required is M, the first task of agent A, which is currently executing. Task M has three possible duration values (denoted d_1^M, d_2^M and d_3^M). Currently, its duration in the active schedule is represented by a single value (e.g., the mean of the three possible durations). If the agents obtain, at time t, the actual duration outcome of method M, they can update their schedules. The potential scheduling changes include the rescheduling of tasks $M4 - M9$ and possibly terminating tasks M, $M2$ or $M3$, all of which are executing at time t. In the absence of information from an owner, rescheduling is based only on the receipt of execution completion messages, or the lack thereof. Several different reschedulings may occur, depending on the actual outcome of M, as shown in the bottom schedules.

If the actual duration outcome of M is d_1^M (bottom left schedule), then upon receiving the execution completion message at time d_1^M, the ASAs can generate a new schedule from that point onwards. The remaining tasks in agent A's schedule may be rescheduled (e.g., replacing the order of execution of tasks $M8$ and $M4$). Agents B and C gain little because $M2$ in agent B's schedule has already been executed and $M3$ in agent C's schedule has already progressed significantly. If the actual duration outcome of M is d_2^M (bottom middle schedule), then at time d_1^M the ASA can revise the schedule relying on the fact that no execution completion message has been received. The flexibility for rescheduling at this point includes all the tasks that have not started executing. The only flexibility for tasks that are already executing is that they can be dropped. In the example given in the figure, the rescheduling that takes place at time d_1^M (based on the fact that the duration of task M is either d_2^M or d_3^M) results in replacing $M6$ with task $M15$. The execution completion message will, in this case, be received at time d_2^M, and the ASAs can generate a new schedule from that point onwards. As a result, the remaining tasks in agent A's schedule may be rescheduled. However, $M2$ in agent B's schedule has already been executed, $M5$ in agent B's schedule has already started executing, $M3$ in agent C's schedule has already been executed and $M15$ in that agent's schedule has come a long way in its execution path. The agents have lost the flexibility to change their schedules until d_2^M, and the rescheduling that takes place at time d_2^M is constrained by the rescheduling that took place at time d_1^M and the tasks executed up to d_2^M. Similar rescheduling occurs if the actual duration of M is d_3^M.

The expected performance of the team if it does not receive information from the owner must therefore be calculated based on continuous revisions of the current schedule due to elimination of outcomes and the final revision upon the receipt of the actual outcome (as part of the execution completion message). We refer to the schedule produced based on updates associated exclusively with the receipt or absence of execution completion messages as the "execution-completion-based schedule (EC-based schedule)".

3.2 The Querying Algorithm

As a result of the complications described in the preceding section, the algorithm we define uses a Bayesian-decision-theoretic based approach for calculating the value of information an owner may have [41, 3, 8]. For each possible task outcome, it considers the consequences associated with getting this specific information before execution completion (or failure), and, alternatively, through execution-completion-based reasoning; it weighs each possibility's contribution by the probability it will occur. The algorithm queries the scheduler iteratively, while emulating the re-scheduling dynamics that would result at each stage, to determine the value of information an owner may have.

Before giving the algorithm itself, we illustrate the calculation for a task M, which has k potential outcomes, with $k' \leq k$ different duration values $\{d_1, ..., d_{k'}\}$ sorted by their length. If M is scheduled to start executing at the current time, the goal is to calculate the value of knowing the actual outcome of M at some time t which is during M's execution. For exposition purposes, we begin by assuming t is the current time, and we later explain the changes needed to calculate the value at any time during the execution of the task.

Since the CI does not know which of the task's outcomes is the outcome known to the owner, the expected value to the system of receiving the actual outcome of task M at time t, denoted $V(interact, t, M)$, is the sum of the values gained for all possible outcomes, weighted by each outcome's a priori probability. We use $S_t(T, I, Sched)$ to denote the schedule that the scheduler produces if it receives at time t a scheduling problem T associated with the active schedule $Sched$ and the new information I, which is a subset of O that includes all outcomes that are still valid. The information I can either be a single outcome (e.g., when the user specifies the actual outcome) or a set of several remaining outcomes (e.g., when some outcomes can be eliminated due to the time that has elapsed). $Sched$ encapsulates all the constraints imposed by former scheduling and re-schedulings. Denoting the quality of schedule $S_t(T, I, Sched)$ by $S_t(T, I, Sched).quality$ and the value of the duration characteristic of any outcome o_i by $o_i.dur$, the expected value of obtaining the actual outcome of task M at time t, is given by:

$$V(interact, t, M) = \sum_{i=1}^{k'} \sum_{o_j.dur=d_i} P(o_j)\Big(S_t(T, o_j, Sched).quality - \tag{1}$$
$$S_{t+d_i}(T, o_j, S_{t+d_{i-1}}(T, O - \{o_w | o_w.dur \leq d_{i-1}\}, ..., S_{t+d_1}(T, O - \{o_w | o_w.dur \leq d_1\}, Sched))...).quality\Big)$$

Equation 1 sums over each possible outcome of task M the difference between the quality of the schedule produced by the scheduler at time t if it knows that outcome o_i is the actual outcome and the expected quality of the schedule produced if it does not have this information at time t. This difference is the difference in value that results from obtaining this specific information from the owner. The second term in the summation corresponds to the case where task execution is completed at $t + o_j.dur$, at which point this information becomes available to the ASA anyway. The quality $S_{t+d_i}(T, o_j, S_{t+d_{i-1}}(...)).quality$ is the quality of the schedule produced by the scheduler at time ($t + o_j.dur$), where the input used for generating this schedule is based on the scheduling updates that will have taken place up to time $(t + o_j.dur)$. As illustrated in Figure 4, by time $t + o_j.dur$, the scheduler has lost some flexibility in scheduling, because none of the tasks that started execution between time t and time $t + o_j.dur$ can be re-scheduled.

If t is not the current time, the same analysis holds with minor revisions. For instance, consider the case where task M, which is already executing at time t, is the task for which the CI needs to calculate the value of knowing its actual outcome. If task M started executing at time $t_{start} < t$, then any outcome associated with a duration shorter than the time elapsed from t_{start} (i.e., any outcome o_j for which $o_j.dur < t - t_{start}$) is no longer valid. Any other outcome $o_i \in \{o | o.dur \geq t - t_{start}\}$ thus has a revised occurrence probability,

$$P(o_i)' = \frac{P(o_i)}{\sum_{o_i \in \{o | o.dur > t - t_{start}\}} P(o_i)} \tag{2}$$

This probability-update process is managed by the ASA based on the absence of an execution completion message before time t. Therefore, the same principles described above, taking the revised probabilities for each outcome of M, hold for calculating the value of obtaining at time t the actual outcome of a task M that is currently executing.

The calculation of the improvement in quality that would result from learning that a possible outcome o_a is the actual outcome therefore requires two types of queries to the scheduler: (1) a query that assigns a probability of 1 (complete certainty) to outcome o_a initially; and (2) a constrained query that incorporates the results of re-scheduling processes that occur up to time $t + o_a.dur$ and then assigns a probability of 1 to outcome o_a.

Algorithm 1, given in pseudo-code, embodies the principles described in Section 3.1, and captured in Equation 1, for calculating the value of obtaining the actual outcome of a task before the task completes execution. Once the CI module calculates this quality value gain, it can multiply it by the probability that the owner really does have information about the actual outcome [45], and, if the result is greater than the estimated cost associated with obtaining it, initiate an interaction.

The core of the Algorithm 1 computation is: (a) the calculation of the differences in the expected quality with and without information about the outcomes associated with the shortest duration; and (b) if execution exceeds the shortest possible duration outcome, recursive execution of the algorithm on the new schedule to obtain the quality of the revised group schedule for the reduced problem. The input for the algorithm is a scheduling problem, T (with an active schedule, denoted $T.Sched$), a task M (with a new outcome distribution) and the time t when M's actual outcome can potentially be obtained from the external source. It is with the recursive Step 13 that the algorithm emulates the re-scheduling process used by the ASA's EC-based scheduling.

Algorithm 1 $GetValue(T, M, t)$ - Calculating the value (in group quality terms) of obtaining the actual outcome of a task.

```
Input: $T$ - a scheduling problem; $M$ - A task from $T$ that we are working on; $t$ - the time at which $M$'s actual outcome is
    obtained from the owner;
Output: $V$ - the expected improvement in group activity quality if the owner provides the actual outcome at time $t$.
 1: Set $Value = 0$;
 2: if $M$ has no outcomes then
 3:     return 0
 4: end if
 5: Set $O' = \{o_i | o_i \in M \wedge o_i.dur = min\{o.dur | o \in M\}\}$
 6: for any $o_i \in O'$ do
 7:     Set $Value = Value + P(o_i)(S_t(T, o_i, T.Sched).quality - S_{t+o_i.dur}(T, o_i, T.Sched).quality)$
 8: end for
 9: Set $T.Sched = S_{t+min\{o.dur|o \in M\}}(T, \{o_i | o_i \in M\} - O', T.Sched)$
10: Set $P_{remain} = 1 - \sum_{o_i \in O'} P(o_i)$
11: Set $P(o_i) = P(o_i)/P_{remain}\ \forall o_i \in M - O'$
12: Remove from $M$ the set of outcomes $O'$
13: return $Value + P_{remain} * GetValue(T, M, t)$
```

The algorithm stores the value accumulated through its execution in the variable *Value*. It identifies outcomes associated with the minimal duration and stores them in O' (Step 5). The value contributed by each outcome in O' is calculated directly using the calls $S_t(T, o_i, T.Sched)$ and $S_{t+o_i.dur}(T, o_i, T.Sched)$ (Step 7). The first call obtains the outcome o_i at time t and thus does not impose any scheduling constraints beyond that time. The second call obtains the outcome o_i at time $t + o_i.dur$; thus the schedule produced is constrained by any tasks scheduled between time t and $t + o_i.dur$.

Upon completing this calculation, the algorithm generates the EC-based schedule resulting from the elimination of all outcomes associated with the minimum duration outcome (i.e., those stored in O') (Step 9). This schedule is used in the recursive calls to generate EC-based scheduling constraints. The probability of the remaining outcomes is updated (Steps 10-12) and the outcomes in O' are removed from the task (Step 12) in preparation for the recursive call in Step 13. The input to the recursive call thus includes an active schedule (as part of T) with all the scheduling revisions that would have occurred due to the elimination of the outcomes with the minimum duration outcome. Therefore, any call to compute $S_{t+o_i.dur}(T, o_i, T.Sched)$ in the recursive execution will be constrained by the EC-based schedules produced by recursive calls for time intervals (t,$t + o_i.dur$). Calls to $S_t(T, o_i, T.Sched)$, on the other hand, will not be affected by scheduling revisions made in the active schedule since these all relate to times later than t. In the remainder of the paper, we refer to $S_t(T, o_i, T.Sched)$ as "the non-constrained schedule" and to $S_{t+o_i.dur}(T, o_i, T.Sched)$ as "the constrained schedule". The stopping rule for the recursive execution is when no more outcomes remain in M (Steps 2-4).

The total number of queries sent to the scheduler through the recursive executions of this algorithm is $2 * |O| + |D|$, where $|O|$ is the number of possible outcomes and $|D|$ is number of possible duration outcomes of the task. This total is derived as follows: (a) two queries (constrained and non-constrained) for each outcome (making a total of $2 * |O|$), (see Step 7); and (b) one query for each duration outcome, to generate the schedule that realizes the elimination of that duration outcome as time elapses (Step 9). We note that the size of O is potentially exponential in the number of outcome characteristics (where duration is one of them), and that the ratio between $|O|$ and $|D|$ is mostly domain-dependent.

The adaptation of Algorithm 1 for the case in which the owner is only able to provide a revised distribution of values (or eliminate a subset of the outcomes), requires several rather straightforward changes and engenders one modeling challenge. The main change is in the generation of the non-constrained schedules, i.e., those schedules that are currently generated using the call $S_t(T, o_i, T.Sched).quality$ in Step 7. These schedules will need to be generated separately, as each different distribution of outcomes that the owner might supply will now result in a different sequence of EC-based schedule updates during task execution. The generation of such EC-based schedules follows the logic in Step 9 of the algorithm. This change adds $|D - 1|$ queries for calculating the value of each revised probability distribution that the owner might supply. It also requires that the EC-based schedule generation process (currently executed in Step 9) be executed separately, as a preliminary step rather than being interleaved as part of the regular execution. This latter change does not engender an increase in the number of queries required, as compared to the current algorithm design. The difference between the quality of the two schedules needs to be weighted according to the probability of receiving a specific revised distribution of outcomes. The challenge, which we describe in Section 7, is the need to model the probabilities of the possible revised distributions.

4 Improving Efficiency

Algorithm 1 requires repeated querying of the scheduler, which is a resource-intensive task. As a result, techniques that decrease the number of queries that the CI issues for reasoning about the value of information can significantly improve ASA performance. In this section, we introduce two complementary types of heuristic methods, each of which significantly reduces the number of queries sent to the scheduler. These methods take advantage of a "time-critical change" concept, which is defined in this section, to reduce the number of outcomes for which the scheduler needs to be queried. They do so without compromising accuracy (i.e., they either provide the exact value of information or reliably determine if that value is above or below a pre-defined threshold); they are heuristic in the sense that we cannot guarantee the amount of improvement they will provide. Methods of the first type are useful for situations in which the CI needs to compute the exact value of information; it reduces the number of queries by intelligently selecting the sequence of outcomes for which queries are generated and re-evaluating the effectiveness of further queries based on the results obtained from the scheduler. Methods of the second type are useful when the CI needs to determine only whether the value of information necessarily exceeds (or necessarily is less than) a given threshold value (e.g., the cost of obtaining the information from the owner).

4.1 Value Calculations

As described in earlier sections, new owner-provided information improves scheduling quality by allowing scheduling modifications that are likely to result in better performance given the actual world state. Information obtained at time t has value only if it leads some agents to change their task schedule in a way that they could not have done based only on an EC-based schedule. If no ASA changes its owner's schedule in comparison to an EC-based schedule, then there is no value in receiving that specific outcome ahead of time and its contribution to the overall weighted sum in Equation 1 is zero. Given a possible task outcome o, we may classify changes in the active schedule that result from knowing the actual outcome of the task is o into "time-critical" changes and "non-time-critical" changes. A *time-critical change* is a change in the active schedule that cannot be made in the active schedule only from an EC message or a lack thereof. The existence of a time-critical change indicates that the ASA's learning about a task's actual outcome o from an owner will yield a positive value for the ASA. A *non-time-critical change* is a change that can be made in the schedule even if the information regarding the actual outcome o is received after the task finishes executing. Formally, the definition of a time-critical change is as follows:

Definition 1 *For an initial problem T, its current schedule $Sched$, and a specific actual outcome o of task $M \in T$: Any scheduling differences between the schedules $S_t(T, o, Sched)$ and $S_{t+o.dur}(T, \{o_i | o_i \in M\}, Sched)$ in the interval $(t, t + o.dur)$ are time-critical changes.*

For the scheduling situation illustrated in Figure 4, the only possible time-critical changes for outcomes of duration d_1^M are for those tasks that are executing or are planned to be executed prior to time $t + d_1^M$. Therefore, only changes in the execution of tasks M, $M2$ and $M3$ before $t + d_1^M$ that result from receiving the information that the task will have duration d_1^M are time-critical changes. If the schedules $S_t(T, o_i, T.Sched)$ and $S_{t+o_i.dur}(T, o_i, T.Sched)$ are identical during $(t, t + d_1^M)$, then whatever the plan is in the non-constrained schedule for time $t + d_1^M$ and later, the same schedule can be devised at time $t + d_1^M$ as a result of obtaining the execution completion message at that time. Similarly, only changes between the constrained and non-constrained schedules in the intervals $(t, t + d_2^M)$ and $(t, t + d_3^M)$ are time-critical changes for duration outcomes d_2^M and d_3^M, respectively.

Time-critical changes affect the value of obtaining the actual outcome o_i at time t, while non-time-critical ones do not. If only N ($N \leq |O|$) outcomes are actually associated with time-critical changes then the theoretical minimum number of queries to the scheduler required to calculate the value of information is $2 * N + |D|$, where the $|D|$ additional constrained queries are required for the re-planning procedure as before and the $2 * N$ queries (N constrained and N non-constrained ones) are for calculating the quality differences for those outcomes that yield time-critical changes.

The first query-reduction method we present, the "Time-critical scanner", uses the fact that time-critical changes are all that matter for identifying non-zero-value differences in the summation in Equation 1. The difference in quality between the constrained and non-constrained schedules is necessarily positive if there are time-critical changes between the two at any time prior to $t + o_i.dur$. Otherwise, the difference is necessarily zero.

To reduce the number of unnecessary queries to the scheduler, the Time-critical scanner orders scheduler queries as follows: First it requests the EC-based schedule for each possible duration outcome of the task. Then it sends the scheduler all of the non-constrained queries (based on $S_t(T, o_i, T.Sched)$). Finally, it initiates a constrained query ($S_{t+o_i.dur}(T, o_i, T.Sched)$-based) only for those outcomes o_i, associated with a time-critical change in the schedule that results from the non-constrained query. The method takes advantage of the fact that the identification of time-critical changes can be reduced to comparing the non-constrained schedule of each outcome with the EC-based schedule correlated with that outcome's duration.[5] The validity of such a comparison derives from the fact that all of the EC-based constrained schedules that are produced for outcomes associated with the same duration outcome d are similar. The number of queries generated overall when using the Time-critical scanner method is $|O| + |D| + N$, which may be derived as follows: (a) $|D|$ queries for generating the EC-based schedule as in Algorithm 1; (b) $|O|$ queries for obtaining the non-constrained schedules of the different outcomes; and (c) N constrained queries to calculate the quality differences for those outcomes associated with time-critical changes. For instance, in the example given in Figure 3, if only two out of the five outcomes are associated with time-critical changes (i.e., $|D| = 3$, $|O| = 5$ and $N = 2$) then instead of executing 13 queries, the CI will need to execute only 10 queries.

This approach decreases the number of queries required for calculating the value of information, but it is still far from the theoretical minimum required number of queries $(2 * N + |D|)$. A further significant

[5]The comparison applies only to tasks scheduled prior to $t + o_i.dur$.

reduction in the number of queries may be obtained by utilizing a regularity that often exists. The occurrence of time-critical changes in the schedule resulting from specific outcomes bears a relationship to their relative position in the outcomes space. The regularity reflects the fact that if two outcomes o and o' have identical values for all of their outcome characteristics except for one characteristic j (formally, $o'.v_i = o.v_i$,$\forall i \neq j$, and $o'.v_j = o.v_j$, where $o.v_i$ denotes the value of the i-th outcome characteristic of outcome o of Task M) and the added value of knowing each of these outcomes a priori was calculated to be zero, then the value of knowing a priori any other outcome o'' placed between the two along the j axis in the outcome space (i.e., satisfying $o.v_i = o'.v_i = o''.v_i, \forall i \neq j$ and $min(o.v_j, o'.v_j) < o''.v_j < max(o.v_j, o'.v_j)$) is also zero.

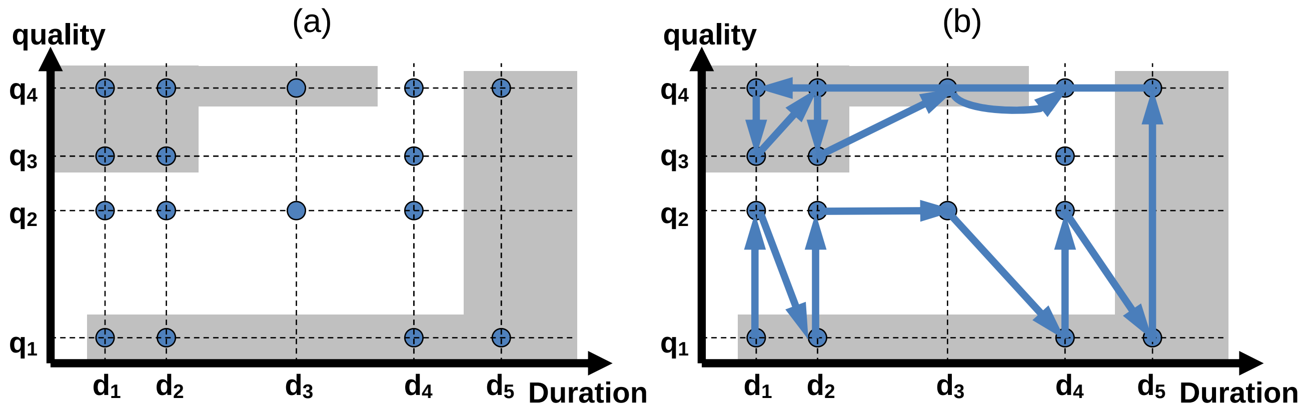


Figure 5: A hypothetical task-outcome space where each outcome is defined by a (duration, quality) pair. The gray area represents outcomes associated with time-critical changes. The arrows in the right-hand graph represent the sequence of queries generated using the Outcome-space scanner.

To better understand the regularity, consider Figure 5(a) which portrays an outcome space in which an outcome is defined merely by duration and quality characteristics for a task M with 5 distinct possible duration outcomes, 4 distinct quality outcomes and 16 outcomes overall.[6] Now, consider an arbitrary outcome $o = (d_i, q_j)$ that is not associated with time-critical changes. Although this outcome is not associated with time-critical changes, its neighboring outcomes in the outcomes space, (d_i, q_{j+k}) or (d_{i+k}, q_j), $k > 0$, may be associated with time-critical schedule changes (e.g., (d_4, q_4) is not associated with time-critical schedule changes while (d_4, q_1) and (d_5, q_4) are). Table 1 illustrates some ways in which an increase or a decrease in the duration or quality results in time-critical schedule changes (both in the executing ASA's and in other ASAs' schedules). Now consider an outcome o', which is different

[6]The decision of whether or not all possible combinations of different values result in a valid outcome or not is domain-dependent. In the case of more than two outcome characteristics, the outcome space representation is a multi-dimensional space.

	Changes in the executing ASA's own schedule	Changes in other ASAs' schedules
$duration \uparrow$	Task M exceeds deadline and is thus replaced by task $\overline{M'}$	Task M_1 (enabled by M) cannot start on time, and is thus replaced with $\overline{M_2}$
$duration \downarrow$	The execution of task M can now be delayed and a new task $\overline{M'}$ can be scheduled before it	Task $\overline{M_1}$, enabled by M, is added to the schedule, and consequently task $\overline{M_2}$ that enables $\overline{M_1}$ needs to be added
$quality \uparrow$	Will never result in a time-critical change in the ASA's schedule	Task $\overline{M_1}$, enabled by the success of M, is added to the schedule, and consequently task $\overline{M_2}$ that enables $\overline{M_1}$ needs to be added
$quality \downarrow$	Task M is replaced by task $\overline{M'}$ (which has a better quality)	A task M_1 which is likely to fail due to the poor performance of M is replaced by task $\overline{M_2}$

Table 1: Time-critical schedule changes due to new information about task M's quality and duration ($\overline{M_j}$ denotes a task that is currently non-scheduled).

from outcome o only in the value of the duration outcome. If o' is associated with time-critical changes, then any other outcome o'' differing from o only in its duration outcome, for which $o''.dur > o'.dur$ (if $o'.dur > o.dur$) or $o''.dur < o'.dur$ (if $o'.dur < o.dur$) is also necessarily associated with time-critical changes. The same holds for changes only in the values of the quality characteristic.

This ripple effect happens if an increase or a decrease in the value of characteristic j consistently increases or consistently decreases the schedule quality. It is explained by the fact that the rationale used for changing the schedule when realizing that the actual value of an outcome characteristic i is v_i', rather than v_i remains valid also in the transition from v_i to v_i'' (keeping all the other outcome characteristics values fixed). For example, if the scheduler determines that a reduction in the duration of a task, in comparison to the hypothetical value, will change the schedule in a way that some of the changes are time-critical changes, then necessarily knowing that the duration of the task is going to be even shorter results in time-critical changes.

The regularity suggests that the space of outcomes for which time-critical schedule changes occur in the outcome space (marked by a gray background in Figure 5(a)) typically wraps around the space representing the outcomes for which no time-critical schedule changes occur. This regularity can be exploited for reducing the number of queries made to the scheduler by considering the value-of-information calculation as a "game" defined as follows:[7] The player is given a multi-dimensional matrix representing the space of possible outcomes. Each element in the matrix may be associated either with time-critical changes or not. At each game stage, the player needs to determine which element of the matrix to query next. For each element queried, an indication is received of whether that element is associated with

[7]The use of the term "game" is due to the resemblance to the famous guessing game, "Battleship".

time-critical schedule changes. If two matrix elements along the same axis are found not to be associated with time-critical changes, then all outcomes between them (i.e., differing only in having a value of that outcome characteristic that is between the values of that characteristic in the two outcomes) are also not associated with time-critical schedule changes. The goal is to identify the matrix elements that are associated with time-critical changes (marked in gray in Figure 5) while querying as few elements as possible that are not associated with time-critical changes. A pre-processing stage for the system before playing this game is to generate the $|D|$ constrained schedules, based on which the time-critical changes will be identified for each outcome queried during the game.

Thus, instead of using the mechanism described in the former subsection for identifying outcomes in which time-critical schedule changes occur, we can construct a heuristic method for solving the game described above. We call this method "Outcome-space scanner". Several simple efficient algorithms can be used for playing this game. Algorithm 2 is one of them. For exposition purposes, we use C to denote the set of outcome characteristics and C_i to denote the set of exclusive values that the i-th outcome characteristic obtains. The algorithm stores the accumulated value in the variable $Value$. The array $flag[]$ is used to mark the outcomes for which the value of knowing that outcome in advance has already been calculated or inferred based on the regularity which is the basis for this approach. The array $marginal[]$ is used to store the value associated with each outcome.

The algorithm works on a two-dimensional matrix defined by the possible values of the last two outcome characteristics, $C_{|C|-1}$ and $C_{|C|}$. Each element of the matrix potentially represents an outcome of the task. If the task has more than two outcome characteristics, the values of the remaining $|C - 2|$ outcome characteristics are picked according to the selection rule specified in Step 6. Here, the representation $o_1 \prec o_2$ is used for defining the case where o_2 has a value greater than that of o_1 for the first outcome characteristic for which the two differ in their value (i.e., having a greater value for the i-th outcome characteristic overrides having a lower value in any of the j-th outcome characteristics, $j > i$). Once the two-dimensional matrix is defined by the selection of o, the algorithm iterates over it, attempting to uncover the boundaries of the area representing the outcomes for which no time-critical schedule changes occur.

We demonstrate the algorithm using the example given in Figure 5(b). The iterative process is managed with the variables $HrzDirection$, $VrtDirection$, Hrz and Vrt. The first two reflect the horizontal and vertical direction of movement in the matrix and the last two determine the actual position. The direction of movement is determined by the data structure $Corners$ and the variable $Corner$. The first holds a set (x, y, h, v) which determines an initial position (x, y) in the matrix (i.e., a corner) and the direction of movement horizontally and vertically (h, v). We begin with the outcome associated with the lowest values of the two outcome characteristics defining the matrix by setting $Corner$ to zero (Step 7) and

Algorithm 2 Outcome-space scanner - Calculating the expected value (in group quality terms) of obtaining the actual outcome of a task.

Input: T - a scheduling problem; M - The task we are working on; t - the time to obtain M's actual outcome from owner;
Output: V - the expected improvement in group activity quality if receiving the actual outcome from the owner at time t.

```
 1: Set $Value = 0$; // Initialize supporting variables
 2: For all $1 \leq i \leq |O|$ Set $flag[i] = false$; $marginal[i] = -1$;
 3: Set $Corners = \{(0,0,1,1), (|C|,0,-1,1), (|C|,|C|,-1,-1), (0,|C|,1,-1)\}$;
 4: Set $AdvanceCorner = false$; $AdvanceColumn = false$;
    // As long as some outcome values are not determined, pick a bi-dimentional matrix of outcomes
 5: while $\exists\ flag[i] == false$ do
 6:     Set $o = \{o | o \prec o_i \forall (o_i \in O) \&\& flag[i] == false\}$;
 7:     Set $Corner = 0$; $Hrz = 0$; $Vrt = 0$; $HrzDirection = 1$; $VrtDirection = 1$;
        // Iterate over outcomes starting from all four corners of matrix
 8:     while $Corner < 4$ do
 9:         if $\exists o_i = (o.v_1, ..., o.v_{|C|-2}, C_{|C|-1}[Hrz], C_{|C|-2}[Vrt]) \in O$ then
10:             Set $marginal[i] = GetDifference(T, o_i)$; $flag[i] = true$;
11:             Set $Value = Value + marginal[i] * P(o_i)$
                // Identify all outcomes that comply with the regulatory mechanism
12:             if $marginal[i] == 0$ then
13:                 while $\exists(j,l,w)$ satisfying $(marginal[j] == 0)\&\&\ (o_j.v_m == o_i.v_m, \forall m \neq l)\&\&$
                    $(o_w.v_m == o_i.v_m, \forall m \neq l)\&\&(min(o_i.v_l, o_j.v_l) \leq o_w.v_l \leq max(o_i.v_l, o_j.v_l))$ do
14:                     Set $flag[w] = true$; $marginal[w] = 0$;
15:                 end while
                    // Switch to the next corner/column
16:                 if $Vrt == Corners[Corner][1]$ then
17:                     Set $AdvanceCorner = true$;
18:                 else
19:                     Set $AdvanceColumn = true$;
20:                 end if
21:             end if
22:             Set $Vrt = Vrt + VrtDirection$;
                // If needed, advance to next column
23:             if $(Vrt \in \{-1, |C|\}) || (AdvanceColumn)$ then
24:                 Set $Vrt = Corners[Corner][1]$; $Hrz = Hrz + HrzDirection$; $AdvanceColumn = false$;
25:             end if
                // If needed, advance to next corner
26:             if $(Hrz \in \{-1, |C|\}) || (AdvanceCorner)$ then
27:                 Set $Corner = Corner + 1$; $Hrz = Corners[Corener][0]$; $Vrt = Corners[Corener][1]$;
                    $HrzDirection = Corners[Corener][2]$; $VrtDirection = Corners[Corener][3]$;
                    $AdvanceCorner = false$;
28:             end if
29:         end if
30:     end while
31: end while
32: return $Value$;
```

calculate the value of receiving a priori that outcome (Step 10). The calculation uses the function *GetDifference*, which extracts the value based on the constrained and non-constrained queries (implementing Step 7 of Algorithm 1), but it first checks the existence of time-critical changes in the schedule resulting from knowing the outcome and avoids sending the second query if one exists. To avoid the need to regenerate the EC-based schedule used in $S_{t+o_i.dur}(T, o_i, T.Sched)$ each time, the entire EC-based schedule can be produced once, at the beginning of the process, requiring $|D|$ queries to the scheduler (using a method similar to the one introduced in Algorithm 1). The calculated value, multiplied by the a-priori probability of that outcome, is added to $Value$ (Step 11). If the value stored in $marginal$ is zero (i.e., the algorithm reaches an outcome that does not involve time-critical schedule changes), the algorithm attempts to identify all outcomes bounded by the selected outcome and other outcomes in the outcome space that have already been identified to have a zero value (Steps 12-15). The value of any outcome of the latter type is set to zero and the outcome is marked as processed (in $flag$). Also, if the value of an outcome is zero, then the values of the binary variables $AdvanceColumn$ and $AdvanceCorner$ are set, in order to indicate that a change in the column or corner to continue with is required (Steps 16-20).

The remainder of the algorithm (Steps 22-28) handles the advancement within the column. In particular it handles column switch and corner switch for effectively iterating around the area that wraps around the one representing the outcomes for which no time-critical schedule changes occur. A column switch occurs when reaching an outcome that does not involve time-critical schedule changes or scanning all elements of a column. A corner switch occurs when reaching an outcome that does not involve time-critical schedule changes which is placed on the first or last rows of the matrix or scanning all columns in a given direction). [8]

The execution terminates when the added-value of all outcomes has been incorporated in the calculation. An alternative stopping rule can be based on a maximum number of queries allocated for the calculation by the ASA or once the accumulated value exceeds the cost of obtaining the information. Such a stopping rule can be implemented simply by adding such constraints to the condition checked in Step 10.

The Outcome-space scanner is guaranteed to send at most the same number of queries to the scheduler as the Time-critical scanner method does ($|O| + |D| + N$). In the worst case, when there are no outcomes associated with time-critical changes, and thus none that can be discarded from the value calculation, it requires sending $2|O| + D$ queries to the scheduler, similar to Algorithm 1 and to the Time-critical scanner method. However, as the number of queries that are not associated with time-critical changes increases, the Outcome-space scanner reliably tags more outcomes associated with a zero value, resulting in a two-fold benefit. First, the scheduler does not need to be sent the non-constrained query for these outcomes. Second, it increases the probability of identifying other outcomes as associated with a zero value without sending the scheduler any further queries related to these outcomes. The method's best case performance is when all the outcomes not associated with time-critical changes are aligned

[8]Depending on the dimensions of the problem, it may be better to scan rows instead of columns, but that process is analogous.

along the same axis in the outcome space, in which case it will generate only $2N + |D| + 2$ queries to the scheduler, which is only two more than the theoretical optimal bound discussed above. This latter number is derived as follows: (a) $|D|$ queries for generating the EC-based schedule as in Algorithm 1; (b) N queries for obtaining the non-constrained (and N for the constrained) schedules of all outcomes associated with time-critical changes; and (c) two single queries to learn that the two outmost outcomes of the sequence of outcomes not associated with time-critical changes are such.

4.2 Value Accumulation

The ultimate goal of calculating the value of information is deciding whether this value exceeds the cost of obtaining such information. Therefore, an alternative to calculating the exact value of information is determining merely that this value is greater than that cost. By determining that the value of information exceeds a pre-defined threshold (cost) with only a partial set of queries, a CI can make a decision using fewer scheduler resources. For example, consider the scenario given in Table 2. For each of four possible outcomes of a task, the table depicts the a priori probability, the quality of the schedule produced if this outcome is known to be the actual outcome, the quality of the schedule without obtaining this information until EC time and the difference between the two. The first column numbers the paired queries to the scheduler required for producing the schedule value with and without knowing about the outcome, and the last column is the weighted sum accumulated to this point (i.e., the sum given in Equation 1 up to the current outcome). For example, the weighted accumulated value after executing queries 5-6 is 13, which is the sum of the product of the probability and difference values in the first three rows. If the cost of obtaining the information is less than 25, then the actual outcome should be obtained. In this case, if the cost of obtaining the information is in the interval (13-25), then deciding it is worthwhile to get the information requires executing all 8 queries. However if the cost is in the intervals (4-13) and (0-4), then realizing the usefulness of obtaining the information requires 6 and 4 queries, respectively. Now consider a different ordering of query execution according to which the CI first executes queries 7-8 and then queries 5-6 (and finally queries 3-4). In this case, realizing the usefulness of obtaining the actual outcome for costs of (0-12) requires only 2 queries, and for costs of (12-21) and (21-25) only 4 and 6 queries, respectively.

For any given cost of interruption, the new ordering allows a decision to be made with fewer queries. This alternative ordering has an additional advantage when the number of queries that can be executed is limited (due to time constraints or ASA computational resource limitations). For example, if the cost of interruption is 20 and the ASA is allowed to execute only 4 queries, then with the original ordering, the wrong conclusion is reached and the information is not requested from the user. In contrast, with the alternative ordering the usefulness of obtaining the actual outcome can be realized and the right decision

Queries	Outcome	Probability	Quality with information	Quality without information	Difference	Weighed accumulated added value
1-2	o_1	0.2	20	20	0	0
3-4	o_2	0.2	40	20	20	4
5-6	o_3	0.3	40	10	30	13
7-8	o_4	0.3	40	0	40	25

Table 2: A possible order of executing the queries associated with the outcomes of hypothetical task M.

is made.

As this example shows, the order in which a CI sends queries to the scheduler affects its ability to minimize the number of queries that need to be sent to the scheduler in order to make a decision. Ideally, the CI would send queries in such a way that the weighted accumulated added value (last column in Table 2) at each step of the process is maximized. Doing so would enable it to reach a decision with the fewest number of queries to the scheduler. The ordering of the pairs of queries according to their weighted added value is a "gold standard", which, unfortunately, is only a theoretical ideal. Although this gold standard cannot be achieved, because the values of the different query pairs are not known a priori, it is a useful benchmark for evaluating methods aiming to produce an effective ordering of queries.

An effective ordering of queries also facilitates early identification (and elimination) of non-beneficial attempts to obtain external information. The sooner the CI is able to determine that information about a task outcome has an expected value lower than the cost of obtaining that information, the sooner the CI can "drop" the calculation. Realizing that the value of information is lower than the cost of obtaining it is facilitated by having an upper bound for the expected accumulated weighted value of that information. For example, if the gold standard ordering is used, then each subsequent pair of queries yields a smaller difference than those obtained for pairs preceding it. Denoting the value of the difference calculated as part of the summation used in Equation 1 for the j-th query pair by F_j yields: $F_j \geq F_{j+1} \forall 0 < j < |O|$, and $\sum_{j=1}^{|O|} F_j = V(interact, t, M)$. Thus, the accumulated expected value of the $|O| - j$ query pairs remaining after executing the j-th query pair is at most the sum of their probabilities multiplied by the difference that was calculated for the j-th pair. Formally, the upper bound for the expected value of interacting with the owner based on the calculation of j query pairs according to the gold standard sequence, denoted $\overline{V_j}(interact, t, M)$, is

$$\overline{V_j}(interact, t, M) = \sum_{i=1}^{j} F_i P(o_i) + F_j \sum_{i=j+1}^{|O|} P(o_i). \tag{3}$$

The upper bound given by Equation 3 is strictly decreasing as a function of j and converges to the value $V(interact, t, M)$ calculated using Equation 1. Figure 6 illustrates this process.[9]

[9]Since the gold standard is a theoretical sequence, the CI can only estimate the upper bound. The accuracy of the estimation may be improved by using exponential smoothing to estimate the maximum marginal value of the remaining query pairs.

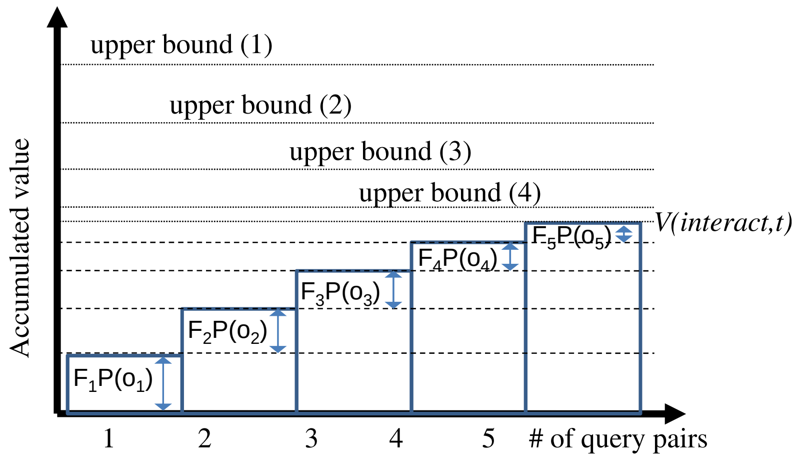


Figure 6: Depiction of upper bound convergence (Equation 3). The upper bound gets tighter upon the execution of any query pair until finally reaches the exact value of information.

Methods may be designed to effectively re-order the queries sent to the scheduler in a way that generally reduces the number of queries needed in order to reach any intermediate accumulated weighted value. We present two new heuristic methods for effectively sequencing the queries for value accumulation purposes. These methods, denoted *Duration scanner* and *Potential-impact scanner*, aim to prioritize outcomes that are more likely to result in substantial values at each stage of the querying process.

Duration scanner: This method starts by executing a query pair for the outcome $o \in M$ having the highest probability of occurrence ($P(o) \geq P(o') \forall o' \in M$). It then considers the set O' of outcomes with the same values as o for the full set of outcome characteristics, except for having a greater duration outcome (i.e., $O' = \{o' | o'.v_i = o.v_i \forall v_i \neq dur \wedge o'.dur > o.dur\}$). The querying process continues for all outcomes in the set, sequentially, according to the duration characteristic value, in ascending order, until either the difference between the group activity's quality of the two queries forming the pair is zero or the set O' is empty.[10] It recurs by choosing the outcome with the highest probability of occurrence among those that have not yet been queried. The idea is that since the F_i values are weighted in Equation 1 according to the probability of the outcome, moving between outcomes with greater probabilities may be beneficial.

Figure 7(a) illustrates a possible execution of the Duration scanner, based on the hypothetical task-outcome space that is given in Figure 5. In this example, the (d_4, q_1) outcome is the one associated with the highest probability of occurrence and therefore is the first to be queried. Outcome (d_5, q_1) is queried next as it is the subsequent one in terms of duration value. In the absence of any more outcomes with the

[10]The decision to "crawl" along the duration axis is based on the idea that the duration characteristic is inherent in the scheduling domain and is usually the characteristic that most constrains scheduling.

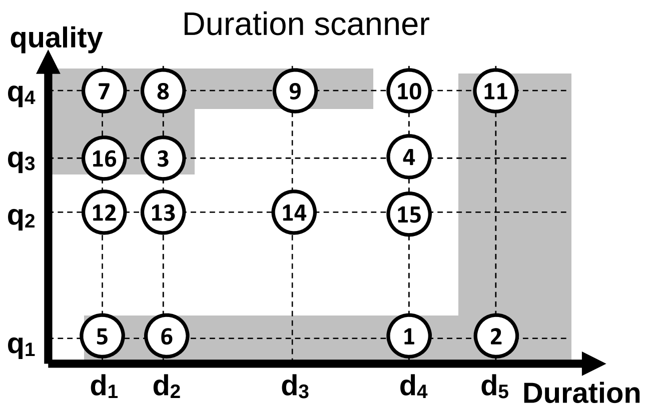


Figure 7: Value of information calculation with (a) Duration scanner (left-hand-side) and (b) Potential-impact scanner (right-hand-side), based on the hypothetical task-outcome space that is given in Figure 5.

same quality outcome and greater duration outcome, however, the next outcome to be queried is (d_2, q_3) which is associated with the highest probability among those that have not been queried. Consequently (d_4, q_3), which is the next in line in terms of its duration outcome value (and with the same quality outcome value), is queried. This continues, in the same way, until all outcomes have been queried.

Potential-impact scanner: This method makes use of the time-critical change concept, identifying zero-valued outcomes in the context of the multi-dimensional outcome space. It attempts to predict the relative added-value of different outcomes according to their place in the outcome space and uses this prediction to order queries. The distance in the outcome space of an outcome from the hypothetical outcome that is used by the scheduler for constructing the active schedule (either max, min or mean) is used as a measure of the magnitude of the added-value of knowing that this outcome is the actual outcome of the task. The idea is that outcomes with extreme characteristic values in comparison to the hypothetical outcome used by the scheduler are more likely to account for substantial added-value. After each query and scheduler response, the Potential-impact scanner attempts to determine if there are additional outcomes that can now be associated with zero added-value, based on the regularity discussed in Section 4.1.

Algorithm 3 specifies the pseudo-code of the Potential-impact scanner. The algorithm stores the accumulated value in the variable $Value$. It first calculates the distance in the outcome space of each outcome from the hypothetical outcome (denoted $hypothetic$) used by the scheduler (Step 2). The distance of each outcome characteristic from the hypothetical value is normalized according to the range of possible values to this outcome characteristic and stored in the array $dist$. The greater this value, the more likely that receiving this outcome a priori will result in substantial scheduling revision. The array $flag$ is used for marking outcomes that have not been processed yet. At each step, the outcome

Algorithm 3 Potential-impact scanner - Accumulating value (in group quality terms) of obtaining the actual outcome of a task.

```
Input: $T$ - a scheduling problem; $M$ - The task we are working on; $t$ - the time to obtain $M$'s actual outcome from the owner; $hypothetic$: the hypothetical outcome that is currently used by the ASA;
Output: $V$ - the expected improvement in group activity quality if receiving the actual outcome from the owner at time $t$.
 1: Set $Value = 0$;
 2: Set $dist[i] = \sqrt{\sum_{j=1}^{N}((o_i.v_j - hypothetic.v_j)/(max(o_1.v_j, ..., o_{|O|}.v_j) - min(o_1.v_j, ..., o_{|O|}.v_j)))^2}, \forall o_i \in M$
 3: For all $1 \leq i \leq |O|$ Set $flag[i] = false$; $marginal[i] = -1$
 4: while $\exists$ $flag[i] == false$ do
 5:     Set $i = ArgMax_i(dist[i] * P(o_i))$;
 6:     Set $marginal[i] = GetDifference(T, o_i)$; $flag[i] = true$;
 7:     $Value = Value + marginal[i] * P(o_i)$
 8:     if $marginal[i] == 0$ then
 9:         For any $(j, l, w)$ satisfying $(marginal[j] == 0)\&\&(o_j.v_m == o_i.v_m, \forall m \neq l)\&\&$
            $(o_w.v_m == o_i.v_m, \forall m \neq l)\&\&(min(o_i.v_l, o_j.v_l) \leq o_w.v_l \leq max(o_i.v_l, o_j.v_l))$,
                Set $flag[w] = true$; $marginal[w] = 0$;
10:     end if
11: end while
12: return $Value$
```

with the highest product of its normalized distance from the hypothetical outcome and its probability is chosen (Step 5), and the value of receiving that outcome a priori is calculated (Step 6). The calculation uses the function *GetDifference*, similar to its use in Algorithm 2, to extract the value associated with each outcome based on the constrained and non-constrained queries. Also, as in Algorithm 2, to avoid the need to regenerate the EC-based schedule used in $S_{t+o_i.dur}(T, o_i, T.Sched)$ each time, the entire EC-based schedule can be produced once, at the beginning of the process, requiring only $|D|$ queries to the scheduler (using a method similar to the one introduced in Algorithm 1). The value attributed to the selected outcome is stored in the array $marginal$.

If the value stored in $marginal$ is zero, then the algorithm attempts to identify all outcomes bounded by the selected outcome and other outcomes in the outcome space that have already been identified to have a zero value (see Steps 8-10), in a way similar to the one used in Algorithm 2. The value of any outcome of the latter type is set to zero and the outcome is marked as processed (in $flag$). The execution terminates when the added-value of all outcomes has been incorporated in the calculation.

Figure 7(b) illustrates a possible execution of the Potential-impact scanner, based on the hypothetical task-outcome space that is given in Figure 5. The black circle represents the hypothetical outcome used by the schedule for scheduling purposes. In this example, the first outcome to be queried is (d_4, q_2) as the product of its normalized distance from the hypothetical outcome and its probability is the highest. Then (d_1, q_2) is queried, for the same reasons. At this point, the value of (d_2, q_2) and (d_3, q_2) can be determined as zero even without querying based on the outcome-space regularity discussed above. The next outcome to be queried is (d_2, q_4) and so on.

The Duration scanner and the Potential-impact scanner have several properties worth highlighting. First, they reflect the goal of finding out as soon as possible whether information should be obtained from the user. Second, the use of these methods does not change the correctness of the information-value calculation and the resulting decision regarding obtaining it. As described above, if necessary, all query pairs will be used and the same value reached as the calculation without the methods. Third, they execute in polynomial time (linear in the number of possible outcomes). Thus, if either of these methods substantially reduces the number of queries that need to be executed, its computational cost will be negligible, because query execution time is several factors of magnitude greater than the polynomial time cost of these methods.

5 Empirical Investigation

The algorithms and heuristic methods described in this paper were designed and tested within the context of a large multi-agent planning and scheduling system, the Coordinators Project [58, 5, 53, 30, 4]. This project aimed to construct intelligent cognitive software agents able to assist fielded units in adapting their mission plans and schedules as their situations changed. Typical sources of potential owner-provided information in this application are coordination meetings (e.g., used for reporting status of task execution), open communications that the owner overhears (e.g., if a radio is left open, the owner may hear messages associated with other teams in the area) and direct communication with other owners participating in a joint task (through which an individual often learns informally about the status of actions being executed by others). An owner's prior experience in similar situations could also be a source of task-outcome related information [47, 43]. The scheduling problem that arises in Coordinators is known to be difficult (NP-hard even in its deterministic version) and not easily modeled and solved by traditional solvers for planning and scheduling [56]. The dynamic nature of the environment makes it even harder since as the scale of the problem increases, it becomes infeasible to calculate and store an optimal set of policies corresponding to the different states the system can be in as the different uncertainties are resolved [31].

In the first subsection, we briefly describe the Coordinators Project, which provided the implementation context and testbed environment for the approach to calculating value as described in this paper. Subsequent subsections establish the usefulness of calculating information value, and the improvement in efficiency yielded by the query-reduction methods both for calculating the value of information and value accumulation. While we would have liked to compare our results to alternative approaches to coordination autonomy in the Coordinators project, the other teams that focused on CI addressed different problems (e.g., the cost of interruption) and were not comparable.

5.1 Systems Context

Although a variety of approaches were taken in the design of ASAs for the Coordinators Project [39, 35, 42, 53], they all included four key modules: a single-agent scheduler; a component (called "Negotiation") for communicating and coordinating scheduling with the ASAs of other owners; a module that determined how to allocate an ASA's reasoning resources ("Metacognition"); and a coordination interface ("coordination autonomy"). Planning and scheduling problems were represented in Coordinators using cTAEMS structures [27, 35], which can be used to define multi-agent hierarchical tasks with probabilistic expectations on their outcomes.[11] cTAEMS is an instance of the general hierarchical task model representation described in Section 2, and we detail here only the specializations that are important for the description of the experiments and their results. The main elements of cTAEMS structures are (atomic, single-agent) tasks, inter-action dependency links, and functions for percolating task-performance measures up the hierarchy as tasks are executed. The multiple possible discrete outcomes of a task represent its possible durations and quality characteristics. In general, the quality of a task represents the contribution of performance of that action to the overall team effort and is execution-dependent. It contributes to the quality of its parent task by means of a quality accumulation function (QAF). If a task violates its temporal constraints, it is considered to have failed and yields a zero quality. Task interactions are represented by non-local effects links (NLEs) such as enablement and facilitation.

The algorithms and methods described in this paper were developed for the coordination interface of the SRI cMatrix architecture [13, 53], illustrated in Figure 8, and tested in this context. Figure 8 shows the connections among cMatrix modules. As illustrated, the architecture separates the scheduler and CI modules, and numerous modules interact with the scheduler. The CI module is responsible for deciding intelligently when and how to interact with the owner. The scheduler is the key resource used by the CI module. The Metacognition module also affects the CI as it controls the computational resources allocated to different ASA modules. The methods developed in this paper address the challenges raised by the need for the CI to take into account the resource constraints of the scheduler and Metacognition allocation.

The basic CI architecture is given in Figure 9. The "owner profiling" and "owner state monitoring" components track the owner's interruptibility preferences and costs as well as managing interactions with the owner [45, 47]. The "Event Handler" manages interactions with the Metacognition module and determines (future) times at which there is a high probability the owner will have new information that may affect system performance. For example, an owner is more likely to know the probability of success of a (future) task shortly prior to the task's execution than further from its execution time.

[11]This language is described in the paper for presentation purposes only. The algorithm and methods developed in the paper could be used with other representations without any adaptation.

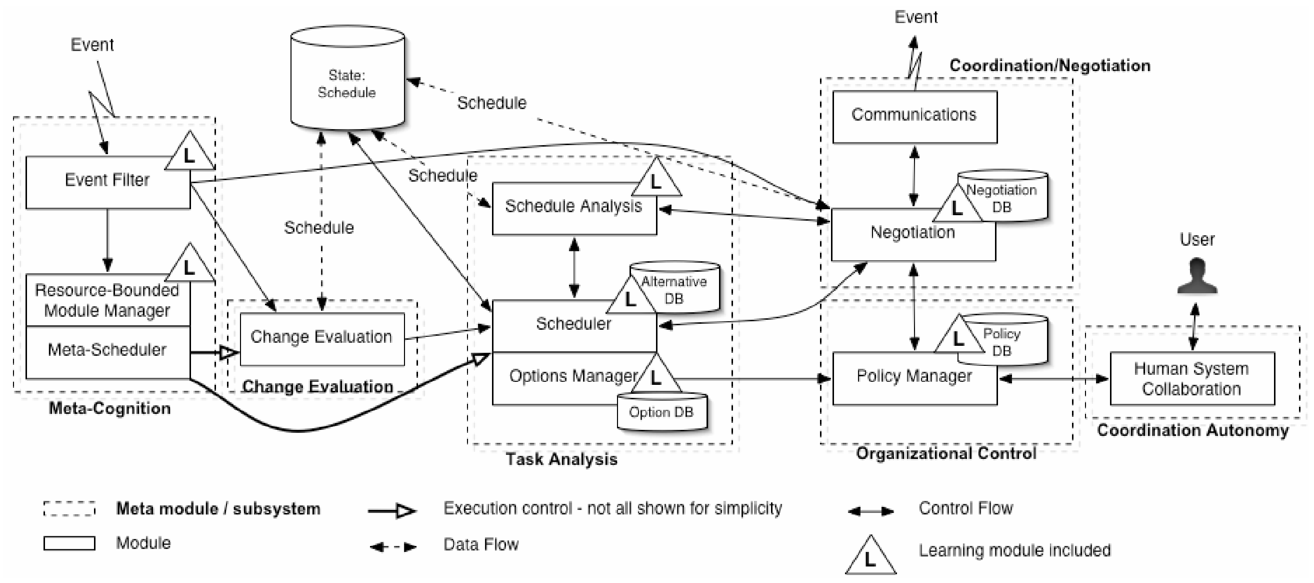


Figure 8: The cMatrix Coordinators architecture. The Coordination Autonomy is the CI.

The "Interaction Evaluator" evaluates the benefit of interacting with the owner at a particular time. If the expected benefit is positive, then an interaction with the owner is initiated and monitored by the "Interaction Handler". The mechanism and algorithms presented above enable the Interaction evaluator to activate the scheduler efficiently in order to estimate the value of a specific piece of task or environment information.

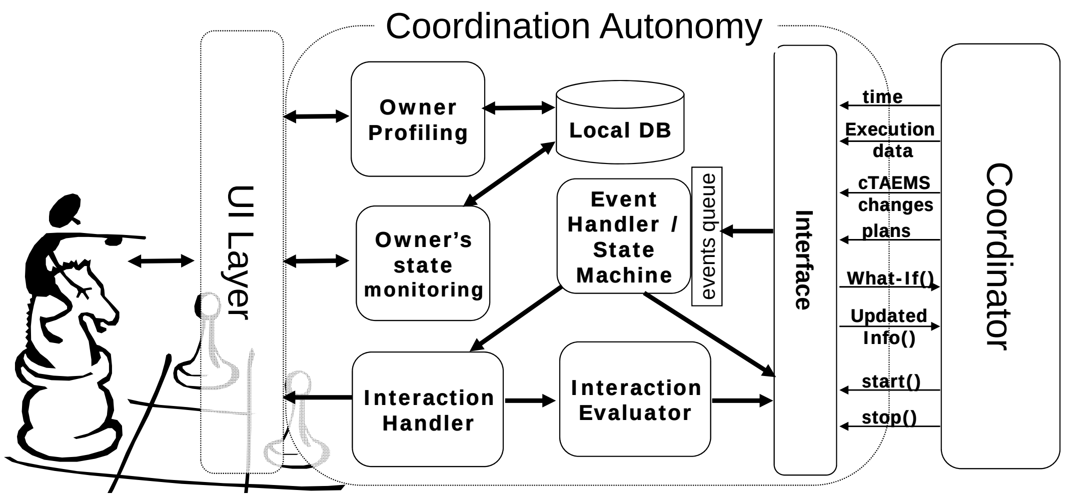


Figure 9: The CI architecture.

5.2 Validating the Usefulness of Information Value Calculation

Our CI module implementation was successfully integrated into the cMatrix Coordinators ASA and evaluated in a real-time simulation environment created for the Coordinators project. Figure 10 depicts the number of times the CI attempted to obtain the actual outcome of a task and the average improvement in team quality as a function of the interaction cost. These results are based on an initial test of the CI module that was done as part of its being integrated into a full ASA system. The test used 30 typical cTAEMS problems. The goal of this test was to demonstrate the benefit of obtaining the actual outcome from the user. The CI was not limited by the number of queries it could send the scheduler, because this test was not concerned with reducing the number of queries made to the schedule. Likewise, the owner interruption "cost" was taken to be known to the system. Each problem was tested with different cost values, in order to verify the expected effect of this variable on the number of interactions initiated and the benefit obtained. There was no attempt to model the cost of interrupting the user.

The results demonstrate that the improvement in the average quality of the group activity from learning the actual outcome early (from an owner) can be significant. While in many cases there was no schedule change, the improvement in schedule quality was substantial in those cases for which the schedule was updated as a result of the new information. Similarly, when the cost of interaction is relatively high, the CI did not find interaction with the owner to be worthwhile in any of the test scenarios. The improvement in quality when the interruption cost is zero provides an upper bound for the improvement that can be achieved. The improvement in quality decreases as the interaction cost increases, as the potential increase in schedule quality to be gained by interacting with the user is outweighed by the associated costs.

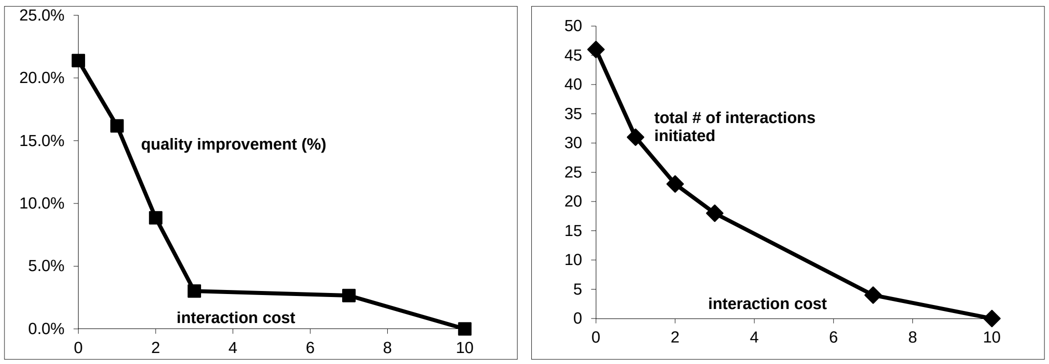


Figure 10: Average schedule quality improvement and number of interactions initiated as a function of the interruption ("interaction") cost.

5.3 Query Reduction

To evaluate the query reduction and value accumulation methods, we used the Coordinators' constraint-based scheduler [56] which enabled the execution of extensive testing without the overhead of the full Coordinators simulation system. These experiments used the test suite that the Coordinators project used to evaluate different ASA architectures.

For each test case, the CI picked a random time in the schedule and activated the value-of-information calculation mechanism for the specific task that was scheduled to be executed by this ASA's owner at that time. The use of a randomly selected time was meant to overcome the possible effect of timing on the results. Tasks executed at final stages of the schedule are likely to have a small number of time-critical changes for any of their outcomes. As a result, both the Time-critical scanner and Outcome-space scanner will produce substantial improvement in comparison to Algorithm 1. In tasks scheduled early in the process, there is much room for improvement in the schedule. As a result, more outcomes are likely to be associated with time-critical changes, and consequently there is a more modest improvement in the number of queries that need to be sent to the scheduler.

The following techniques from Sections 3 and 4 were compared: (a) Order scanner: evaluating all outcomes using Algorithm 1; (b) Time-critical scanner: sending a non-constrained query for each outcome and then sending constrained queries as necessary; and (c) Outcome-space scanner: using the game-based algorithm described in Section 4. In addition, we calculated the theoretical minimum number of queries needed ($2 * N + |D|$) for each problem, denoted "Theoretical" in describing results and in the results graphs.

The results are based on the complete Coordinators test suite of 2093 problems. In reporting results we use the Coordinators project division into classes based on such parameters as the scale of the scheduling task, the number of agents, and the number and type of interdependencies (NLEs) between tasks. Each class represents different problem characteristics and complexity. While the magnitude of the improvement varies with problem-class, the ordering of the performance of the different methods holds uniformly across all classes of problems (i.e., does not depend on the Coordinators problem class). The classes are:

- Class 1 (299 problems) - minimal dependency between tasks (no "enables" nor "facilitates" NLEs).
- Class 2 (360 problems) - substantial dependency between tasks, mostly in terms of "facilitates" NLEs.
- Class 3 (408 problems) - substantial dependency between tasks, mostly in terms of "enables" NLEs.

- Class 4 (309 problems) - problems characterized by temporal tightness; Large range of number of methods used in each problem.
- Class 5 (361 problems) - missions with synchronization points.
- Class 6 (356 problems) - missions with synchronization points and substantially large numbers of "facilitates" and "enables" NLEs.

Figure 11(a) depicts the average number of queries used by the CI for calculating the value of information as a percentage of the number of queries used by Algorithm 1, according to problem class. As shown in the graph, the Outcome-space scanner provides the greatest improvement and is very close to the theoretical minimum. For problems of classes 5 and 6, both the theoretical minimum and the Outcome-space scanner produce very few queries. This result is explained by the fact that in many problems in these classes a feasible solution did not exist, due to an over-constrained problem. As a result, many of the tasks' outcomes were associated with zero values, and they were easy for the Outcome scanner to find.

Figure 11(b) depicts the additional overhead, in percentages, of each method in comparison to the theoretical minimum number of queries, with the number of possible quality outcomes as the controlled variable, for all 2093 problems. This graph shows that while the overhead of the Order scanner and Time-critical scanner methods increases (though modestly) as the number of possible outcomes increases, the Outcome-space scanner algorithm exhibits constant improvement as a function of the number of outcomes. The intuitive explanation for this phenomena is that when there are more outcomes, the task is more likely to have sequences of outcomes associated with non-time-critical changes of the type the Outcome-space scanner method detects (and thus eliminates the need to send queries for). Furthermore, the number of outcomes in each such sequence, for which no queries are sent, will be greater. Therefore, even though an increased number of queries is sent when there are more outcomes, the relative additional overhead actually decreases. For 9 quality outcomes, the Outcome-space scanner has only a 23% average overhead compared to the theoretical minimum number of queries.

5.4 Value Accumulation

To evaluate the methods developed for improving value accumulation, we used the same 2093 cTAEMS problems and the same division into classes. For each test case, the CI again chose a random time in the schedule and activated the value-of-information calculation mechanism for the specific task that was scheduled to be executed by the ASA's owner. The value calculation process was repeated several times, each time using a sequence of queries generated by a different method (Order scanner, Time-critical

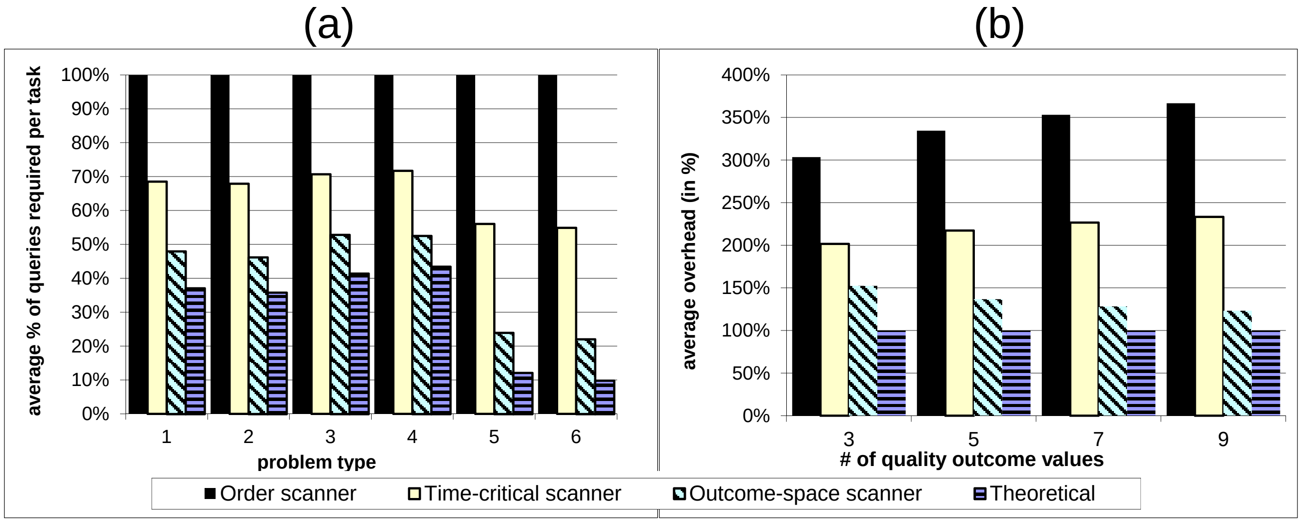


Figure 11: # of interactions initiated for calculating the value of information according to the different methods.

scanner, Outcome-space scanner, Duration scanner and Potential-impact scanner). In addition, the gold standard sequence was extracted and its value was computed.

One difficulty that needed to be resolved in this case is measuring the effectiveness of a sequencing method. Figure 12(a) illustrates the challenge of defining good measures of performance for sequencing methods. It gives hypothetical curves of the accumulated value (vertical axis) of different possible methods for ordering a sequence of $|O|$ query pairs needed to calculate the value of information about the actual outcome of a task.[12] Each accumulated value curve is a non-decreasing function of the number of query pairs executed, because each query contributes a non-negative value to the sum. Furthermore, all of the curves eventually reach the same value because the accumulated value always reaches the actual value of obtaining the information, $V(interact, t, M)$, after all $2|O|$ queries are executed.

In terms of accumulated weighted value, a sequence X of query pairs dominates a sequence Y of query pairs if $\sum_{i=1}^{j} F_i P(o_i^M)$ according to sequence X is greater than or equal to the equivalent sum according to sequence Y $\forall j \leq |O|$. The difficulty of finding a good metric for performance is that while some sequences are always better or always worse than others, other sequences satisfy partial dominance. For example, sequence A in Figure 12(a) dominates all other sequences, sequence B is dominated by all other sequences and sequence E dominates sequence C. In contrast, curve C dominates D if the interruption cost is smaller than c_1, but D dominates C otherwise.

[12]The figure contains $|O|$ queries overall, because the $|D|$ constrained queries required for reproducing the EC-based schedule are executed regardless of the ordering.

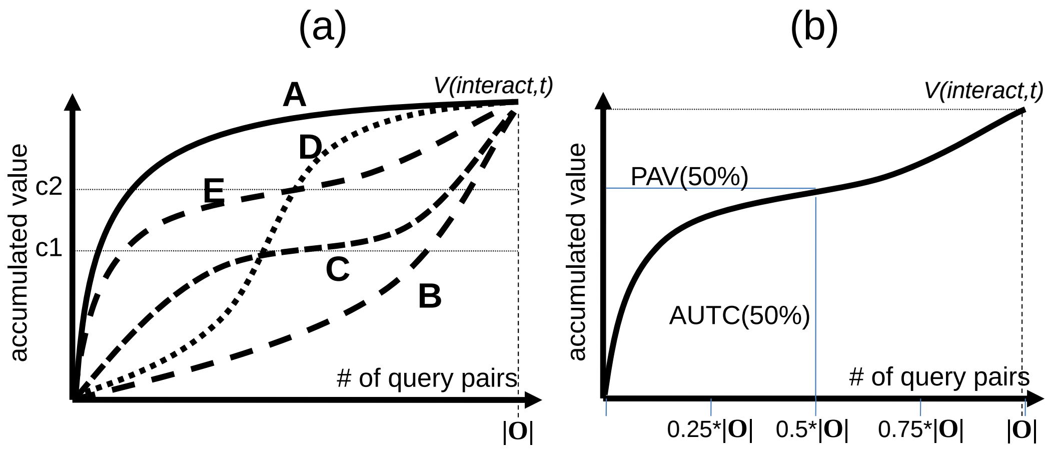


Figure 12: (a) Value accumulated as a function of the number of query pairs sent; (b) $PAV\alpha$ and $AUTC\alpha$ metrics.

The only sequence that always dominates all other sequences is the unrealizable gold standard sequence (sequence A in the example illustrated in Figure 12(a)). To compare the effectiveness of different sequencing methods, performance measures should reflect how close the sequences produced using each method are to the gold standard. For the ASA problem, they should also place more emphasis on the weighted value accumulated by queries early in the sequence, because the sooner the interruption decision is made, the greater the saving of ASA and scheduler resources in the value of information calculation process.

We used three types of metrics to evaluate a method's effectiveness in ordering the queries sent to the scheduler. The first metric, which we call First Moment (FM), is calculated by Equation 4:

$$FM = \sum_{i=1}^{|O|}(|O| - i)\sum_{j=1}^{i} F_j P(o_i^M) \tag{4}$$

In Equation 4, the weighting of the summed differences decreases as the number of query pairs executed for obtaining them increases. This metric assesses the desirability of giving more emphasis to values accumulated early in the sequence.

The other two metrics, which are illustrated in Figure 12(b), are percentile-based. For each percentile α of the queries in a sequence, they calculate the Percentage of Accumulated Value ($PAV\alpha$) achieved (out of the total expected value, $V(interact, t, M)$) and the Area Under the Truncated Curve ($AUTC\alpha$) that was obtained. The $PAV\alpha$ measure provides a snapshot of how much value has been accumulated after α percent of the total number of queries were executed; it thus measures how close a sequence gets to the gold standard after α percent of the queries needed to calculate $V(interact, t, M)$ are made.

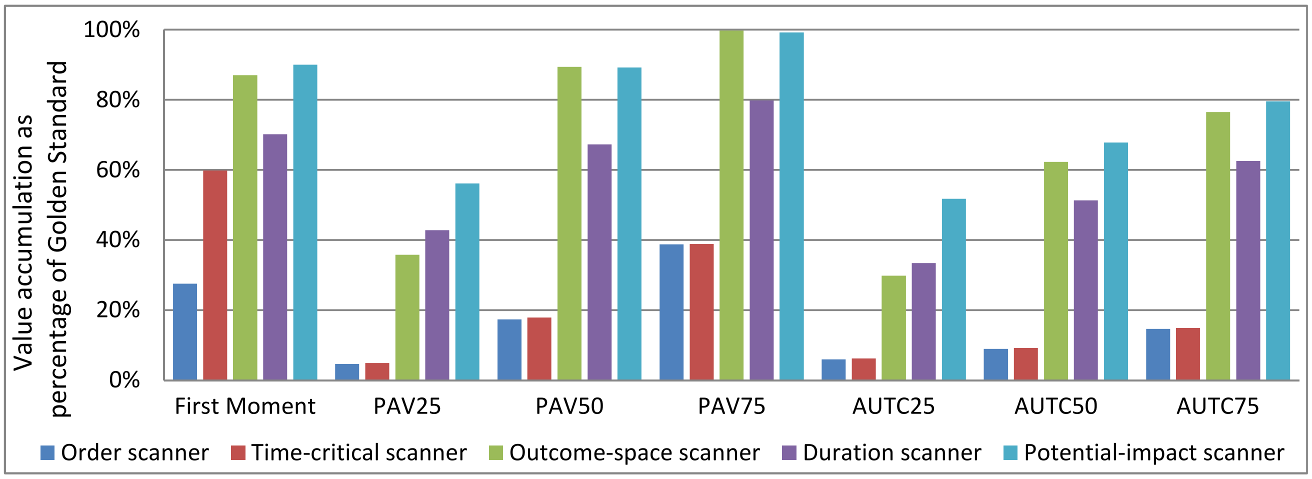


Figure 13: Performance of the different value accumulation methods using different metrics.

The greater the $PAV\alpha$ value, the better the performance of the method. The $AUTC\alpha$ measure gives some indication of the shape of the curve up to the α percentile of queries. In particular, it indicates the rapidness of its asymptotic behavior (concavity of the curve), which is a good indicator of the portion of the value achieved during initial queries. The $AUTC\alpha$ value integrates into one measure the ability of the sequence to accumulate substantial values at the beginning of the process and the value obtained after a specified number of queries.

These three metrics complement each other, as they emphasize different desirable characteristics of value accumulation: performance in time, focus on initial stages and time to reach a percentage of the overall value. They enable a comprehensive evaluation when used together. For relatively small α values, the $PAV\alpha$ and $AUTC\alpha$ metrics give the best indication of a method's ability to efficiently accumulate value early in the calculation. Furthermore, they relate directly to the initial values accumulated. Therefore, they are ideal candidates for assessing the efficiency of a method for early elimination of non-fruitful interactions (by constructing an upper bound for the value of information as illustrated in Figure 6).[13]

The greater the value achieved according to these metrics, the more accurate a decision to terminate early the calculation of the value of obtaining information about a task. Ideally, we would want a method that would perform best for all the metrics, but such a method is theoretically infeasible since for every measure we can find different sequences that will favor different methods.

[13]Alternatively, we could have considered a class of FM-like metrics that emphasize the early queries to different degrees. The more emphasis on the early queries, the more like $PAV\alpha$ and $AUTC\alpha$ for small values of α. To provide a more diverse set of metrics, we made the conservative choice of a linear FM-metric.

Figure 13 shows the performance of each method using the three metrics: FM, $PAV\alpha$ and $AUTC\alpha$. For $PAV\alpha$ and $AUTC\alpha$ we used $\alpha = 25, 50$ and 75. Since the magnitude of the accumulated value is problem-dependent, a mechanism for normalizing the performance of each method had to be constructed. We used the value accumulated using the gold standard sequence for that purpose. It is the vertical axis in Figure 13.

As reflected in Figure 13, the Potential-impact scanner dominates all other methods according to the FM and the $AUTC$ metrics. It also dominates the other methods for the PAV metric with the 25th percentile. The dominance of the method is statistically significant ($p < 0.01$). There is no statistical significance to the difference between the performance achieved by the Potential-impact scanner and the Outcome-space scanner for the $PAV50$ and $PAV75$ metrics. Thus, the Potential-impact scanner can be considered to weakly dominate the Outcome-space scanner (i.e., it guarantees at least as good a result). The dominance of the Potential-impact scanner over the other methods when using the $PAV25$ metric is especially important since this result relates to the stage at which the termination of the process will have the most impact in terms of the number of queries to the scheduler that can be avoided. Furthermore, it is notable that the Potential-impact scanner manages to accumulate more than 50 percent of the value accumulated by the gold standard by the 25th percentile of queries required for calculating the value of information. Overall, the results suggest that the Potential-impact scanner is the best choice for constructing the value of information in the general setting this paper considers.

The results from the experiments using the constraint-based scheduler provide additional comparative assessments of the methods developed for calculating the value of information (Order scanner, Time-critical scanner and Outcome-space scanner). (a) Of these three methods, the Outcome-space scanner is the most efficient in terms of value accumulation (cross-metric). (b) Surprisingly, the Time-critical scanner does not improve the value accumulation rate in comparison to Order scanner alone, despite its advantage in identifying zero-valued outcomes. The one exception is with the FM metric. (c) In the important initial stages of the process (PAV25 and AUTC25), the simple Duration scanner performs better than the Outcome-space scanner.

To further examine the relative performance of the Outcome-space scanner and the Potential-impact scanner methods at early stages, we analyzed how results varied by the six classes of problems used in analyzing the value-of-information experiments. Figure 14 shows the performance of these two methods for each problem group, using the PAV and $AUTC$ measures for the 25th percentile. As the graphs show, the dominance of Potential-impact scanner does not depend on any specific problem characteristics, with one exception: For group 6, when the $PAV25$ measure is used, the Outcome-space scanner performs slightly better on average. This anomaly may be explained by the unique characteristics of the problems in group 6: They had relatively many facilitation relationships, causing these problems to be

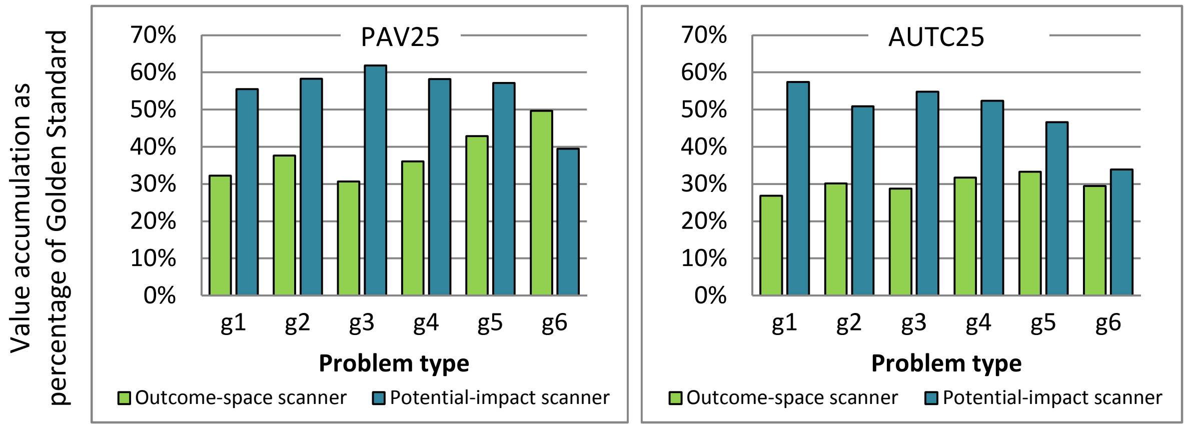


Figure 14: *Outcome-space scanner* and *Potential-impact scanner* value accumulation comparison.

associated with a small value of information in the first place. Also, as discussed above, in many of the problems of this class a feasible solution did not even exist for many of the outcomes. (A significantly small number of problems (13) in this group had a positive value to begin with.)

6 Related Work

Automated planning and scheduling is an active research area in AI, in particular in intelligent autonomous agents [54, 27]. A significant focus recently has been in the area of distributed scheduling systems, using various mechanisms aimed at optimizing the global quality of the system, including negotiation frameworks based on market economies [24], game theoretic algorithms and global or shared evaluation functions [34]. The essential difference between these efforts and the work described in this paper is that they focus on improving the efficiency of producing a schedule and the effectiveness of the schedule produced. They, therefore, relate more directly to the problem addressed by the ASA's scheduler module than to the CI.

Although most scheduling and planning systems research has overlooked issues of calculating the value-of-information, both in general and for architectures that require querying an external scheduler, several approaches to "mixed initiative planning" have considered related issues, including developing agents to assist groups carrying out a complex task in their scheduling and task allocation, eliciting guidance from people to direct planning, and providing appropriate information alerts. We discuss efforts in each area.

The research on agent-assisted scheduling of complex group activities most relevant to this paper was done within the framework of "Electric Elves". This project developed a system that coordinated day-to-day activities and was implemented and used by a research group at USC/ISI [49, 7]. The central research question addressed in this work was developing methods for determining when to transfer decision-making control from the electric elf to the user. Scerri et. al formalize choice of time to transfer control as an expected-utility maximization problem modeled in a Markov Decision Process (MDP). In this model, choices of which domain action to take are interleaved with choices to interact with the user (a different kind of action) and scheduling is handled only implicitly. While this solution was successful for Electric Elves, an MDP-based approach to ASA design is severely challenged by the complex plans and scheduling interactions in the domain. In particular, the Coordinators' ASA scheduling problem is potentially exponential in the tasks and outcomes, depending on intertask relations and QAFs. MDP-based approaches were taken on the Coordinators project. They focused on developing techniques for heuristically guiding the enumeration of a subspace of the full MDP, as for even a single-agent cTAEMS MDP, full enumeration was impractical [36, 2]. In addition, it is not clear that the integration of reasoning about scheduling with reasoning about which domain action or transfer-of-control action to take could be adapted to the ASA setting in which reasoning about the schedule is done by a separate module. We note that as the methods developed in this paper focus on reducing the number of calls made to the scheduler, they might be usefully deployed in constructing an MDP.

Other research on ASA-user interaction in planning domains has considered the possibility of eliciting advice from users to guide the system's planning efforts using this guidance to reduce the planner's (or scheduler's) solution space [30, 37]. This approach rests on the premise that people have good intuitions about how to solve certain planning (or scheduling) problems and that a planner working with a person can come up with better solutions than either operating alone. Several other mixed initiative approaches deploy collaborations between people and agents to produce better solutions for complex problems [38, 1]. This work is complementary to ours.

Other research has considered autonomous agents providing assistance to human planners by providing information alerts at appropriate times [40, 61, interalia]. These works, however, focus mainly on the human side of the human-computer interaction and consider, for instance, the cognitive load created by requests to provide information or alerting the person to important events.

Taking a broader view, we note that computing the value of information is an inherent part of decision-making under uncertainty and therefore is commonly found in mechanisms designed for meta-reasoning for search, in selecting measurements to make prior to choosing a course of action, and in managing the exploration vs. exploitation tradeoff [55]. It offers a systematic treatment to various problems, ranging from active learning and observation selection to user interaction [23, 25]. In particular, it has been used as a sensitivity analysis technique to rate the usefulness of various information sources and to decide whether pieces of evidence are worth acquisition before actually using them [29]. We therefore review related work on calculating the value of information in several intersecting domains, emphasizing the main differences both in terms of the problem for which the value of information needs to be calculated and the methods used.

One such area is “feature/observation selection” (also referred to as “set selection” and “prediction models for active learning” [64, 6]). In this problem setting, the decision maker needs to decide on the subset of expensive observations that will be made [29, 26, 25]. The value of acquiring a set of features involves comparing the system performance with and without the information. Similar to our case, the calculation requires taking an expectation over the possible values of the features in the set [6]. In our case, however, the calculation of this value is substantially complicated by the temporal aspect, especially the additional constraints that may be applied when information is not received early. In feature-selection based models the value calculation for a given set of values is much simpler because these models need to determine the state of the world only at the current time step. The majority of these methods assume that the information is evaluated “in isolation” and the value always derives from a single immediate act of the decision maker immediately after he receives the information. Even in models that consider sequential acquisition of observations [29, 25], all of the information collected is used for supporting a single decision (or determination of a world state). The temporal aspect comes into consideration only in the sense that probabilistic updates are used for evaluating the value in sampling additional sources of information. Therefore, the value of any given subset of values can be calculated simply as the value of the best selection (or decision) that can be made based on the information available. Consequently, these works tend to suggest solutions for dealing with the number of sets that need to be evaluated, which are exponential in the size of the feature set, rather than with the efficiency of the individual calculation of the value of each set. As a result, they usually suggest approximations, based on exploiting the statistical significance of large samples [14] using the central-limit theorem [29] or the strong junction tree framework [9].

Other work in this area focuses on improving the process of computing the value of information by identifying redundant calculations, or make use of the intermediate computation results. For example, Zhang et. al [63] present a method for incremental computation of the value of perfect information in stepwise-decomposable influence diagrams. Bilgic and Getoor [6] introduce two techniques for sharing computations between different subsets of features based on information caching and utilizing paths in the underlying Bayesian network. While the idea of identifying computation overlaps in the processes of evaluating the difference, thus avoiding those overlaps, is similar, the ability to identify such overlaps results from unique properties of influence diagrams, which are important representations for decision making but not used in scheduling. These methods thus cannot be applied in our domain.

Research on human-computer interaction has considered the value of supplying a user with information that is currently known only by the system. The primary goal of such work is to balance the expected value of information with the attention-related costs of disruption [52]. Work in this area differs from our work in three ways. First, in this work the agent has information needed by the person rather than the reverse, and there is thus no need to deal with uncertainty about the information (i.e., to consider a range of different possibilities). Second, their primary concern is when to interrupt users to provide help or information [19, 22, 60, 28]. These efforts focus almost entirely on modeling the user's attentional state, with the goal of intelligently choosing the best time to interrupt (minimize the resulting interference) [17, 19] and the ensuing frustration they cause [22]. Third, it commonly assumes that it is straightforward to calculate the value to the user of the information. For example, Horvitz et. al obtain the benefit, which they term "the value of message information", through user preferences encoded in the system (e.g., the value of getting to a meeting or not) [20]. Others assume that users will change their actions in a pre-determined manner when they get new information. The temporal aspect of the decision, which in our work is reflected by the rescheduling dynamics that occur in the absence of such information, is considered only in calculating interference costs and not in calculating benefits [19].

Other work that directly addresses the problem of estimating the value of information that the user can supply is in non-temporal domains, making the calculation of the value of information simpler. For example, Yakout et. al [62] introduce a framework for user-assisted data repair, that selectively acquires user feedback on suggested updates. The calculation in this case does not involve any temporal constraints, and the efficiency of the approach derives from understanding the mutual effects of the different observations in terms of the validity of the data that the system has. Fleming and Cohen [11, 12] use a utility-based quantitative approach to designing systems that are capable of making decisions about when to ask a user to provide additional information to improve problem solving. Their work focuses on modeling the cost of "bothering" the user repeatedly. Here again, despite focusing on schedule generation, the value-of-information calculation does not consider temporal dynamics or changes in schedule that would occur even if the user did not supply the information.

Continual computation is a research area in which a related value-of-information concept is used [51, 16]. In continual computation, the value of a particular computation is determined by whether its result is useful in solving the next problem. This value is affected primarily by the arrival of problem instances that could use such calculations. In contrast, we have the task schedule and and know the next task to be executed, and the main challenge is the efficient computation of the value of information.

Finally, work by Rosenfeld et al. [43], building on our earlier work [44, 46], estimates the expected utility gain from owner-provided information. This work takes a machine learning approach, correlating the value of information of different patterns and properties of the problem as a whole. It differs from the work described in this paper in focusing mainly on the value of new information about scheduling constraints rather than on information about outcomes. While the method was shown to be effective in determining when additional information about constraints would not be helpful (i.e., would produce zero value), the machine learning models aiming to identify when information would be helpful had a low rate of precision in predicting value and a high false positive rate, and were thus only moderately useful.

7 Conclusions and Future Directions

This paper addresses the problem of computing the value of information in uncertain dynamic environments in which people are supported in their work by an autonomous-agent scheduling system. As they carry out their tasks, these people have access to information that is not directly available to the agent, but is important for effective agent reasoning. The paper considers this problem as it arises in multi-agent coordinated scheduling-system architectures in which the expertise needed for reasoning about alternative schedules is contained within a separate scheduler module.

The methods presented in the paper are applicable to any scheduling-systems context in which a computer agent needs to reason about requesting task-outcome information from an external source, whether that source is a person or a computer system, when there is some significant cost to obtaining such information. In particular, there is no dependency of the algorithms, methods for reducing queries or results on a particular ASA-system design. They are useful for any ASA implementation that uses a separate scheduling module. The only constraint is that the scheduler be consistent (i.e., if it gets the same problem repeatedly, it supplies the same feasible solution). Furthermore, the basic approach is general and applies to any scheduling problem in which there are multiple, linked threads of activity, regardless of the number of agents involved in executing tasks. The greater the degree of parallelism and number of interdependencies among threads, the more likely information about one task will affect other parts of the schedule and the more threads may be affected.

The paper defines a decision-theoretic algorithm that reasons hypothetically about the impact of different task outcomes on schedule revisions, to determine the value of information. It importantly handles the complicating information conditions that the particular task outcome that obtains (or may obtain) is not known to the system, and that this actual outcome will be obtained at a later time that is less useful for replanning (Section 3). The algorithm handles the complex, recursive reasoning needed to accurately infer the schedule that would result if the system does not receive the information immediately (from the external source), but only later, when it receives notice that task execution has completed (or failed) or when some possible task-duration interval has been exceeded. Through the recursive use of prior calculations, the algorithm emulates these dynamics efficiently. To address the challenge of reducing demands on the scheduler, the paper presents two-types of query-reduction methods. One type, which is useful when the exact value of information must be computed, aims to eliminate as many extraneous scheduler queries as possible. The second, which is useful when the system needs to determine only whether the information has a value that necessarily exceeds the cost of obtaining it, orders queries so that information value is accumulated as rapidly as possible.

To evaluate the efficiency of these methods in decreasing the number of queries to the scheduler, we carried out a comprehensive set of empirical investigations using the extensive data set generated for the Coordinators project, which exemplifies environments in which schedules are dynamically, and often continuously, being revised. The empirical results, which are presented in Section 5, clearly establish the usefulness of the recursive decision-theoretic algorithm defined in Section 3 and the effectiveness of both types of query-reduction methods. In particular, both methods yielded significant reductions. The Time-critical scanner, which filters queries depending on whether there has been a change in relevant tasks, is applicable regardless of the structure of the scheduling problem. The Outcome-space scanner approach, which translates the problem of generating queries into a game and then attempts to minimize the score of that game, produced a greater reduction; it depends, however, on the scheduling problem exhibiting monotonicity in the effect of changes in task outcomes on the overall team effort.

The two value-accumulation methods, which are applicable when the system needs to determine only whether the value of information exceeds a certain threshold, were also shown to be efficient in accumulating most of the value associated with a task's outcome early in the querying process (Section 5). Both methods take a greedy approach to determining the order of queries, emphasizing different aspects of the outcome. While the Duration scanner emphasizes the duration of tasks, the Potential-impact scanner emphasizes the distance from the hypothetical outcome according to which the schedule was initially constructed. For this evaluation, we defined three new, complementary metrics for comparing the rate at which the heuristics accumulate value. The experimentation reveals that the Potential-impact scanner is the more efficient of the two.

One important extension of this work is the adaption of the proposed algorithm and methods to the situation in which owners have only partial information. For instance, they may be able only to eliminate some possible outcomes or to otherwise refine the ASA's outcome distribution model, but not to identify a single outcome. As described in Section 3, the changes required for augmenting Algorithm 1 for this case are quite straightforward, and similar issues arise in revising the query reduction and value accumulation methods. There is however one external modeling challenge that limits the applicability in such settings. In theory, there is an infinite number of probability distributions that can be assigned to the set of possible outcomes. Even if we consider the restricted case in which the owner can eliminate some outcomes, the number of possibilities is combinatorial in the number of outcomes. Domain modeling would require computing probabilities for each of these possibilities. Furthermore, the probability that the owner would provide any particular revised distribution (or set of discrete probabilities) depends not only on the task environment, but also on the owner. That is, one needs to model the likelihood that the owner will have various kinds of information, whereas such modeling is not required when assuming the owner has the actual outcome. For example, the owner might easily eliminate rain with fog and rain without fog outcomes if the actual outcome is sunny weather, but it is much less likely he could distinguish (predictively) between these two outcomes, if one of them is indeed the actual outcome.

Another set of extensions involves computing various owner-related characteristics, including the probability that an owner has the necessary information and the cost of an interruption for a particular owner and context. Such characteristics depend not only on the problem state, but also on the owner's state. Research in interruption management [17, 19] and initial work in the Coordinators' setting [45] provide a good foundation for these investigations.

The formulation of an outcome model requires domain expertise. In the Coordinators' project it was assumed that domain experts were available who could provide this information. More generally, the provision of an outcome model might require the deployment of techniques from machine learning.

Finally, there is the problem of discounting the value of information about tasks that are planned for further in the future. While the methods suggested in this paper remain valid for these tasks, the dynamic nature and long time-horizon of some scheduling problems engender uncertainty about whether a task will remain on the schedule by the time it is currently planned to be executed. Therefore, the value of knowing the outcome of such tasks may need to be discounted in some way.

8 Acknowledgments

The research reported in this paper was supported in part by contract number 55-000720, a subcontract to SRI International's DARPA Contract No. FA8750-05-C-0033. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the U.S. Government.

Preliminary versions of the research reported in the paper appear in two conference papers [44, 46]. We are grateful to Monira Sarne and Peter Owotoki for developing the experimental infrastructure, Willem-Jan van Hoeve for providing the constraint-based scheduler and support in its use, and Ben Lubin for insightful discussions and assistance in building an initial CI. Finally, we would like to thank the three anonymous reviewers for their valuable comments and suggestions to improve this paper.

References

[1] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J. Hsu, A. Jónsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, B. Chafin, W. Dias, and P. Maldague. MAPGEN: Mixed-initiative planning and scheduling for the Mars exploration rover mission. *IEEE Intelligent Systems*, 19(1):8–12, 2004.

[2] G. Alexander, A. Raja, and D. Musliner. Controlling deliberation in a markov decision process-based agent. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-08)*, pages 461–468, 2008.

[3] A. Ang and W. Tang, editors. *Probability concepts in engineering planning and design, Volume II - Decision, risk, and reliability*. John Wiley & Sons, New York, 1984.

[4] J. Atlas and K. Decker. Coordination for uncertain outcomes using distributed neighbor exchange. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, pages 1047–1054, 2010.

[5] L. Barbulescu, Z. Rubinstein, S. Smith, and T. Zimmerman. Distributed coordination of mobile agent teams: the advantage of planning ahead. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, pages 1331–1338, 2010.

[6] M. Bilgic and L. Getoor. Value of information lattice: Exploiting probabilistic independence for effective feature subset acquisition. *Journal of Artificial Intelligence Research*, 41:69–95, 2011.

[7] H. Chalupsky, Y. Gil, C. Knoblock, K. Lerman, J. Oh, D. Pynadath, T. Russ, and M. Tambe. Electric elves: Applying agent technology to support human organizations. In *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence*, pages 51–58, 2001.

[8] R. Clemen, editor. *Making hard decisions: An introduction to decision analysis*. Duxbury Press, Belmon, CA, 1991.

[9] S. Dittmer and F. Jensen. Myopic value of information in influence diagrams. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 142–149, 1997.

[10] M. Fleming and R. Cohen. User modeling in the design of interactive interface agents. In *Proceedings of the Seventh International Conference on User Modeling*, pages 67–76, 1999.

[11] M. Fleming and R. Cohen. A user modeling approach to determining system initiative in mixed-initiative AI systems. In *Proceedings of the Eighth International Conference on User Modeling*, pages 54–63, 2001.

[12] M. Fleming and R. Cohen. A decision procedure for autonomous agents to reason about interaction with humans. In *Proceedings of the AAAI Spring Symposium on Interaction between Humans and Autonomous Systems over Extended Operation*, pages 81–86, 2004.

[13] A. Gallagher. *Embracing conflicts: Exploiting inconsistencies in distributed schedules using simple temporal network representations*. PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, USA, 2009.

[14] D. Heckerman, E. Horvitz, and B. Middleton. An approximate nonmyopic computation for value of information. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-91)*, pages 135–141, 1991.

[15] L. Hiatt, T. Zimmerman, S. Smith, and R. Simmons. Strengthening schedules through uncertainty analysis agents. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 175–180, 2009.

[16] E. Horvitz. Principles and applications of continual computation. *Artificial Intelligence*, 126(1-2):159–196, 2001.

[17] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 256–265, 1998.

[18] E. Horvitz, J. Breese, and M. Henrion. Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning*, 2:247–302, 1988.

[19] E. Horvitz, C. Kadie, T. Paek, and D. Hovel. Models of attention in computing and communication: from principles to applications. *Communications of the ACM*, 46(3):52–59, 2003.

[20] E. Horvitz, P. Koch, C. Kadie, and A. Jacobs. Coordinate: Probabilistic forecasting of presence and availability. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 224–233, 2002.

[21] R. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, SSC-2:22–26, 1966.

[22] B. Hui and C. Boutilier. Who's asking for help?: A bayesian approach to intelligent assistance. In *Proceedings of the Eleventh International Conference on Intelligent User Interfaces (IUI-06)*, pages 186–193, 2006.

[23] A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 877–882, 2007.

[24] T. Kis, J. Vancza, and A. Markus. Controlling distributed manufacturing systems by a market mechanism. In *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 534–538, 1996.

[25] A. Krause and C. Guestrin. Optimal value of information in graphical models. *Journal of Artificial Intelligence Research*, 35:557–591, 2009.

[26] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *Proceedings of Information Processing in Sensor Networks (IPSN-08)*, pages 481–492. IEEE Computer Society, 2008.

[27] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. NagendraPrasad, A. Raja, R. Vincent, P. Xuan, and X. Zhang. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, 2004.

[28] X. Li and Q. Ji. Active affective state detection and user assistance with dynamic bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(1):93–105, 2005.

[29] W. Liao and Q. Ji. Efficient non-myopic value-of-information computation for influence diagrams. *International Journal of Approximate Reasoning*, 49:436–450, 2008.

[30] R. Maheswaran, C. Rogers, R. Sanchez, and P. Szekely. Human-agent collaborative optimization of real-time distributed dynamic multi-agent coordination. In *Proceedings of AAMAS Workshop on Optimization in Multiagent Systems*, pages 49–56, 2010.

[31] R. Maheswaran and P. Szekely. Criticality metrics for distributed plan and schedule management. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 14–18, 2008.

[32] R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, pages 310–317, 2004.

[33] W. McClure. Technology and command: Implications for military operations in the twenty-first century. Maxwell air force base, center for strategy and technology, 2000.

[34] P. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal*, 161(1-2):149–180, 2005.

[35] D. Musliner, E. Durfee, J. Wu, D. Dolgov, R. Goldman, and M. Boddy. Coordinated plan management using multiagent MDPs. In *Proceedings of the AAAI Spring Symposium on Distributed Plan and Schedule Management*, pages 73–80, 2006.

[36] D. Musliner, R. Goldman, E. Durfee, J. Wu, D. Dolgov, and M. Boddy. Coordination of Highly Contingent Plans. In *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 418–422, 2007.

[37] K. Myers. Strategic advice for hierarchical planners. In *Proceedings of the Fifth International Conference (KR-96)*, pages 112–123, 1996.

[38] K. Myers, P. Jarvis, M. Tyson, and M. Wolverton. A mixed-initiative framework for robust plan sketching. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS-03)*, pages 256–266, 2003.

[39] J. Phelps and J. Rye. GPGP: A domain-independent implementation. In *Proceedings of the AAAI Spring Symposium on Distributed Plan and Schedule Management*, pages 81–88, 2006.

[40] M. Pollack. Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. *AI Magazine*, 26(2):9–24, 2005.

[41] H. Raiffa and R. Schlaifer, editors. *Applied Statistical Decision Theory*. Harvard University Press, Cambridge, MA, 1961.

[42] A. Raja, G. Alexander, and V. Mappillai. Leveraging problem classification in online metacognition. In *Proceedings of the AAAI Spring Symposium on Distributed Plan and Schedule Management*, pages 97–104, 2006.

[43] A. Rosenfeld, S. Kraus, and C. Ortiz. Measuring the expected gain of communicating constraint information. *Multiagent and Grid Systems*, 5(4):427–449, 2009.

[44] D. Sarne and B. Grosz. Estimating information value in collaborative multi-agent planning systems. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-07)*, pages 1–8, 2007.

[45] D. Sarne and B. Grosz. Sharing experiences to learn user characteristics in dynamic environments with sparse data. In *Proceeding of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-07)*, pages 43–50, 2007.

[46] D. Sarne, B. Grosz, and P. Owotoki. Effective information value calculation for interruption management in multi-agent scheduling. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS-08)*, pages 313–321, 2008.

[47] D. Sarne and B. J. Grosz. Timing interruptions for better human-computer coordinated planning. In *Proceedings of the AAAI Spring Symposium on Distributed Plan and Schedule Management*, pages 161–162, 2006.

[48] P. Scerri, D. Pynadath, W. Johnson, P. Rosenbloom, M. Si, N. Schurr, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 433–440, 2003.

[49] P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

[50] N. Schurr, J. Marecki, J. Lewis, M. Tambe, and P. Scerri. The DEFACTO system: Training tool for incident commanders. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 1555–1562, 2005.

[51] D. Shahaf and E. Horvitz. Investigations of continual computation. In *Proceedings of the Twenty-First International Joint Conference on Artificial intelligence (IJCAI-09)*, pages 285–291, 2009.

[52] T. Shrot, A. Rosenfeld, and S. Kraus. Leveraging users for efficient interruption management in agent-user systems. In *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-09)*, pages 123–130, 2009.

[53] S. Smith, A. Gallagher, T. Zimmerman, L. Barbulescu, and Z. Rubinstein. Distributed management of flexible times schedules. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-07)*, pages 472–479, 2007.

[54] E. Sultanik, P. Modi, and W. Regli. On modeling multiagent task scheduling as a distributed constraint optimization problem. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 1531–1536, 2007.

[55] D. Tolpin and E. Shimony. Rational value of information estimation for measurement selection. In *Proceedings of the Twenty-Fifth Mini-EURO Conference on Uncertainty and Robustness in Planning and Decision Making (URPDM-10)*, 2010.

[56] W. van Hoeve, C. Gomes, B. Selman, and M. Lombardi. Optimal multi-agent scheduling with constraint programming. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*, pages 1813–1818, 2007.

[57] T. Wagner, V. Guralnik, and J. Phelps. A key-based coordination algorithm for dynamic readiness and repair service coordination. In *Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 757–764, 2003.

[58] T. Wagner, J. Phelps, V. Guralnik, and R. VanRiper. An application view of Coordinators: Coordination managers for first responders. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 908–915, 2004.

[59] T. Wagner, A. Raja, and V. Lesser. Modeling uncertainty and its implications to sophisticated control in TAEMS agents. *Autonomous Agents and Multi-Agent Systems*, 13(3):235–292, 2006.

[60] D. Wilkins, T. Lee, and P. Berry. Interactive execution monitoring of agent teams. *Journal of Artificial Intelligence Research*, 18:217–261, 2003.

[61] D. Wilkins, S. Smith, L. Kramer, T. Lee, and T. Rauenbusch. Execution monitoring and replanning with incremental and collaborative scheduling. In *Workshop on Multiagent Planning and Scheduling, The Fifteenth International Conference on Automated Planning & Scheduling*, pages 29–35, 2005.

[62] M. Yakout, A. Elmagarmid, J. Neville, M. Ouzzani, and I. Ilyas. Guided data repair. *Proceedings of the VLDB Endowment (PVLDB)*, 4(5):279–289, 2011.

[63] L. Zhang, R. Qi, and D. Poole. Incremental computation of the value of perfect information in stepwise-decomposable influence diagrams. In *Proceedings of the Ninth International Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pages 400–407, 1993.

[64] Y. Zhang. Multi-task active learning with output constraints. In *Proceedings of the Twenty-Forth National Conference on Artificial Intelligence (AAAI-10)*, 2010.

Notation	Meaning
T	A scheduling problem - consists of a set of tasks applicable to the problem domain, relationships among those tasks, outcome values for each task, quality accumulation methods and an active schedule.
M	A task - the basic scheduling entity with which ASAs work.
o	An outcome of a task - defined by the values it assigns to a set of outcome characteristics (e.g., duration, cost, performance level, resources consumed), each representing a different task performance quality aspect.
$P(o)$	The a priori probability of outcome o.
$o.dur$	The value of the duration characteristic of outcome o.
O	The set of possible outcomes of a task.
t	The time when the actual outcome of a task can be obtained from the external source.
k	The number of potential outcomes of a task.
k'	The number of distinct duration outcomes of a task ($k' = \|D\|$).
$S_t(T, I, Sched)$	The schedule that the scheduler produces if it receives at time t a scheduling problem T associated with the active schedule $Sched$ and the new information I.
$S_t(T, I, Sched).quality$	The quality of the schedule $S_t(T, I, Sched)$.
I	New information I which gives the actual outcome of task M.
$Sched$	The active schedule.
D	A vector of the possible duration outcomes of a task.
F_i	The value of the difference calculated as part of the summation used in Equation 1 for the j-th query pair.
Order scanner, *Time-critical scanner*, *Game*	Methods for calculating the value of obtaining the actual outcome of a task.
Duration scanner, *Potential-impact scanner*	Methods for efficient value accumulation as part of calculating the value of obtaining the actual outcome of a task.
FM, $PAV\alpha$, $AUTC\alpha$	Value accumulation measures.

Table 3: Summary of notation.