DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD

Effective Information Value Calculation for Interruption Management in Multi-Agent Scheduling.

(Article begins on next page)

Effective Information Value Calculation for Interruption Management in Multi-Agent Scheduling

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

Peter Owotoki
Institute for Computer Technology
TU Hamburg Harburg
Germany
owotoki@tu-harburg.de

Abstract

This paper addresses the problem of deciding effectively whether to interrupt a teammate who may have information that is valuable for solving a collaborative scheduling problem. Two characteristics of multi-agent scheduling complicate the determination of the value of the teammate's information, and hence whether it exceeds the costs of an interruption. First, in many scheduling contexts, task and scheduling knowledge reside in a scheduler module which is external to the agent, and the agent must query that module to estimate the value to the solution of knowing a specific piece of information. Second, the agent does not know the specific information its teammate has, resulting in the need for it to repeatedly query the scheduler. Choosing the right sequence of queries to the scheduler may enable the agent to make an interruption decision sooner, thus saving query time and computational load for both the agent and the external system. This paper defines two new sequencing heuristics which enhance the efficiency of the querying process. It also introduces three metrics for measuring the efficiency of a query sequence. It presents extensive simulation-based evidence that the new heuristics significantly outperform previously proposed methods for determining the value of information a teammate has.

Introduction

Several capabilities of autonomous-agents make them ideal candidates for managing schedules and plans, especially in highly dynamic environments and settings in which multiple agents' schedules are being coordinated [22, 21]. They can process a complex range of scheduling information, suggest alternative courses of action to team members, and negotiate scheduling conflicts. Consequently, the role of autonomous-agents in planning and scheduling systems has increased dramatically recently [14, 19]. By taking responsibility for changes in task schedules, agents can free people involved in doing tasks from being unnecessarily burdened, especially in times of crisis when plans need to be adapted to new circumstances [12].

We use the term "automated scheduling agent" (ASA) to refer to autonomous-agent systems that support multi-agent coordinated scheduling of people carrying out complex tasks in a dynamic environment. The quality of the schedules ASAs produce depends on their knowledge of the environment in which these tasks are being performed. In particular, ASAs must take into account various environmental facts that may influence scheduling choices significantly. Information beyond the system's limited knowledge of the environment may affect scheduling constraints, either relaxing them or imposing new ones that better reflect the actual situation [17]. In many cases, the user whose schedule the ASA manages may have more accurate information about the state of the environment than the ASA. For example, a human driver can observe a traffic jam as it forms while a GPS-based navigation system does not have the same immediate access to this information. The earlier such external information is available to the ASA, the greater its influence on system performance.

To obtain information from users, systems must interrupt them. Interruptions of people are inherently disruptive and result in degradation in task performance of the person being interrupted, by distracting the person and by consuming communication resources for the interaction [7, 9, 22]. Hence, Human-agent interactions for obtaining external information must be managed appropriately to avoid overburdening people [14, 19, 2].

The research in this paper supports the development of a collaborative interface (CI) for an ASA. To decide whether to interrupt, a CI must be able to efficiently and accurately calculate the value of information a user has and the cost of obtaining this information. Once these values are determined, the CI should interact with the user to obtain information only if the expected value of the information multiplied by the (estimated) probability a user has this information is greater than the cost incurred for obtaining the information from the user. This paper addresses the problem of calculating the value of information efficiently, complementing a prior paper that analyzed interruption costs [18].

Information is valuable to the extent it influences schedule changes, but that influence cannot be determined without task and scheduling knowledge. In many contexts in which ASAs operate, such task and scheduling knowledge resides

in a "scheduler" module which is external to the CI.[1] As a result, to determine whether the value of a piece of information exceeds the costs of an interruption, the CI must query the scheduler. Thus, for the CI to manage interruptions effectively, it must consider the resource demands it makes on the scheduler as well as the costs of interruptions to the person with whom it is interacting.

The information the CI needs from a user is the actual outcome of a particular task with which the ASA is currently concerned. By "outcome" we refer to the consequence of performing the task; outcome may be defined over single or multiple dimensions (e.g., execution time, quality of performance, resources consumed). The CI knows the set of possible outcomes for a scheduling task, but only has an a priori occurrence probability for each outcome. Getting the information from the user thus facilitates planning as it eliminates the ASA's uncertainty relating to the identity of the true outcome of the task. To calculate the expected value of obtaining the actual outcome from its user, the CI must compute the weighted sum of the values associated with all possible outcomes of the task, where the weighting is by the a priori probability of each specific outcome. Consequently, the CI needs to systematically query the scheduler to obtain the value derived from knowing each possible outcome for a task. For example, if one part of the task is for the user to go somewhere, the CI maybe able to ask the user when he will reach his destination and get back the actual time. In order to calculate the expected value of such an interaction prior to interrupting the user, the CI needs to repeatedly query the scheduler for each possible time the user might reach his destination.

This paper shows that the CI can determine whether to interrupt the user without completing the processing of the full set of queries required for calculating the expected value of information. For example, if the weighted sum of the values returned by the scheduler for a partial set of queries exceeds the cost of interrupting the user then the interruption is definitely favorable.[2] An early decision is beneficial, because it reduces the number of queries sent to the scheduler. This raises a "value accumulation" challenge for the CI: ideally the CI would first send to the scheduler those possible outcomes associated with the highest information value, so that the accumulation of value in the weighted sum would increase as rapidly as possible. The challenge is determining the right sequencing of the queries. In this paper, we evaluate the usefulness of different methods for sequencing queries within the context of value accumulation, where the goal of sequencing is early discovery (and early elimination) of useful (non-useful) interruption opportunities.

The contributions of the paper are three-fold. First, we show the importance of the sequence in which queries are executed to distributed collaborative scheduling. The con-

cept of enhancing value accumulation and the heuristics we define apply to any scheduling-related setting where an evaluating entity needs to rely on external scheduling expertise under resource constraints (e.g., DARPA's *Coordinators* project [13, 16] as well as many rescue and real-time scheduling environments). Second, the paper introduces two new query-sequencing heuristics for efficient value accumulation in early stages of the calculation. Such heuristics are needed, because it is impossible to know the results of queries prior to their execution, but for any sequencing method to dominate (i.e., always outperform other methods), it must, on each iteration, choose to send to the scheduler the outcome yielding the highest information value. Third, we provide an evaluation methodology which deploys three complementary metrics for comparing sequencing heuristics in terms of their value accumulation rate. The effectiveness of the new heuristics is demonstrated experimentally using a comprehensive data set generated for the DARPA *Coordinators* project, a widely used testbed for scheduling domains [20, 11, 17], which is described in more detail in the experimental evaluation section below.

Model and Analysis

To analyze the effect of query sequencing on the rate of value accumulation, we consider a task M about which the ASA needs information because M is scheduled for execution shortly. Task M has k possible outcomes $\{o_1^M, \ldots, o_k^M\}$, known to the ASA, and the a priori probability, as estimated by the ASA, for outcome o_i^M is $P(o_i^M)$, $\sum_{i=1}^N P(o_i^M) = 1$. An outcome is defined by a set of N values, $\{v_1^{o_i^M}, \ldots, v_N^{o_i^M}\}$, where each element of this set relates to a different aspect of the task execution. One inherent outcome characteristic associated with the execution of a task is its duration, the time it took to execute the task. Other outcome characteristics include quality, cost incurred while executing the task, and resources consumed. If the user knows the actual outcome of a task, the CI can learn this outcome of task M by asking the user. The information received from the user will decrease the ASA's uncertainty and enable it to produce a schedule based on the actual task outcome; the outcome the user supplies is then assigned a probability of 1 and all other possible outcomes a zero probability. Alternatively, if the CI does not interrupt the user, the ASA must rely on the a priori probability for each potential outcome. Therefore, the value of having the user provide the actual outcome o_i^M of task M, is the improvement in the quality of the schedule obtained using this information. The value of knowing a specific outcome o_i^M from the user is positive only if the system changes its schedule depending on o_i^M, but is necessarily non-negative as the system can only improve its performance as a result of the decreased uncertainty [17].

Since the ASA does not know which of the task's outcomes will be specified by the user, the expected value to the system of an interaction with the user at time t, denoted $V(interact, t)$, is the sum of the gains associated with each possible outcome weighted according to the a priori probability of this outcome. We use $S_t(T, o_i^M)$ to denote

---

the best schedule that the scheduler can produce at time t, given a scheduling problem T if receiving the information (at time t) that the true outcome of method M is o_i. We denote the value (quality) of a schedule $S_t(T, o_i^M)$ by $S_t(T, o_i^M).quality$. The expected value of interacting with the user at time t in order to learn the true outcome of method M that is scheduled to start at time t is thus

$$V(interact, t) = \sum_{i=1}^{k} \Big(S_t(T, o_i^M).quality -$$

$$S_{t+o_i^M.duration}(T, o_i^M).quality \Big) P(o_i^M) \quad (1)$$

where $o_i^M.duration$ is the value of the duration characteristic of outcome o_i^M, so $t + o_i^M.duration$ is the time that the task execution is completed and thus the information becomes available to the ASA anyway. The quality $S_{t+o_i^M.duration}(T, o_i^M).quality$ thus is the quality of the schedule produced by the scheduler at time ($t + o_i^M.duration$), where the input used for generating it is the schedule that was re-generated at time t based only on the a priori probability of all the different possible outcomes (i.e., $S_t(T)$). In this case, the scheduler loses some flexibility in scheduling, because all the tasks that started execution between time t and time $t + o_i^M.duration$ cannot be re-scheduled. 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 given this specific outcome o_i^M a priori (at time t) and the expected quality of the schedule produced without having this information a priori, i.e., the difference in value that results from obtaining this information from the user.

Therefore, the calculation of the improvement in quality that results from each possible outcome o_i requires two types of queries to the scheduler: (1) a query that assigns a probability 1 to outcome o_i initially, and (2) a constrained query that incorporates the results of re-scheduling processes that occur up to the time that the task completes its execution and then assigns a probability 1 to outcome o_i. Overall, the calculation given in Equation 1 requires sending k pairs of queries if there are k possible outcomes to the task being considered.

To reason about whether to interrupt its user, the ASA does not actually need the exact value of the information it obtains from the user. It only needs to know that this value is greater than the cost of the interruption. 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 to interrupt using fewer computational resources (i.e., fewer queries to the scheduler). For example, consider the scenario given in Table 1, which, for each of four possible outcomes of a task, depicts its a priori occurrence probability, the quality of the schedule produced if this outcome is known to be the true outcome, the quality of the schedule calculated without this information a priori 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

Que- ries	Out- come	Proba- bility	Quality with inter.	Quality w/o inter.	Diffe- rence	Weighed accumulated benefit value
1-2	o_1^M	0.2	20	20	0	0
3-4	o_2^M	0.2	30	40	20	4
5-6	o_3^M	0.3	10	40	30	13
7-8	o_4^M	0.3	0	40	40	25

Table 1: A possible order of executing the queries associated with the outcomes of method M.

last column is the weighted sum accumulated to this point (i.e., the sum given in Equation 1 up to the current pair). If the cost of interrupting the user was 20, an interruption would obviously be beneficial: the total expected benefit of getting this information from the user is greater than the cost of obtaining it. Nevertheless, if instead of executing the 8 queries required for the calculation in this case in the order given in Table 1, the ASA first executed queries 7-8 and then queries 5-6, the decision to interrupt the user could be made (based on accumulated weighted utility of 21) with half the number of queries.

As this example shows, the order in which a CI sends queries to the scheduler plays a significant role in its ability to minimize the number of queries that need to be sent to the scheduler. Ideally, the CI would send queries in such a way that the weighted accumulated expected benefit (last column in Table 1) at each step of the process is maximized. Doing so would enable it to reach a decision about interacting with the user with the fewest number of queries to the scheduler. The ordering of the pairs of queries according to their weighted benefit is a "gold standard", which, alas, is only a theoretical ideal. It cannot be achieved because the values of the different query pairs are not known a priori.

An effective ordering of queries also facilitates early identification of non-useful interactions with the user. The sooner the CI is able to determine that the user's information (about task outcome) has an expected value smaller than the cost of obtaining that information, the sooner the CI can "drop" the calculation. Realizing that the value of interaction with the user is smaller than the cost is facilitated by having an upper bound for the expected accumulated weighted value of the interaction. 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 we obtain: $F_j \geq F_{j+1} \forall 0 < j < k$, and $\sum_{j=1}^{k} F_j = V(interact, t)$. Thus, the accumulated expected value of the $k - j$ query pairs remaining after executing the j-th query pair is at most the sum of their probabilities multiplied by the difference calculated for the j-th pair. Formally, the upper bound for the expected value of interacting with the user based on the calculation of j query pairs according to the gold standard sequence, denoted $\overline{V_j}(interact, t)$, is

$$\overline{V_j}(interact, t) = \sum_{i=1}^{j} F_i P(o_i^M) + F_j \sum_{i=j+1}^{k} P(o_i^M). \quad (2)$$
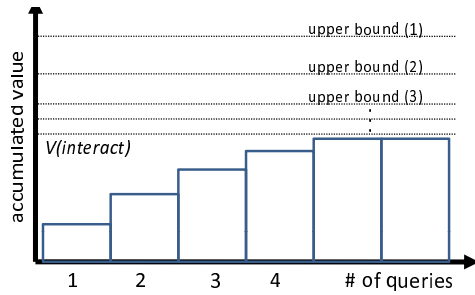
Figure 1: Depiction of upper bound convergence (Equation 2).

The upper bound given by Equation 2 is strictly decreasing as a function of j and converges to the value $V(interact, t)$ when calculated using Equation 1. Figure 1 illustrates this approximation.[3]

Heuristics may be designed to effectively re-order the queries that need to be sent to the scheduler in a way that generally reduces the number of queries needed. As discussed in the following section, the effectiveness of a sequencing heuristic is difficult to measure. Therefore, the paper presents three different metrics, each emphasizing a different aspect of efficiency. These metrics are then used for evaluating the effectiveness of previously proposed different heuristics and two new heuristics presented in this paper.

Metrics

Figure 2(a) illustrates the challenge of defining good measures of performance in order to compare different sequencing heuristics. It gives hypothetical curves of the accumulated value (vertical axis) of different possible methods for ordering the sequence of k query pairs needed to calculate the value of information about the actual outcome of a task. 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, the curves eventually reach the same value because the accumulated value always reaches the actual value of interacting with the user, $V(interact, t)$, after all potential outcome queries are made.

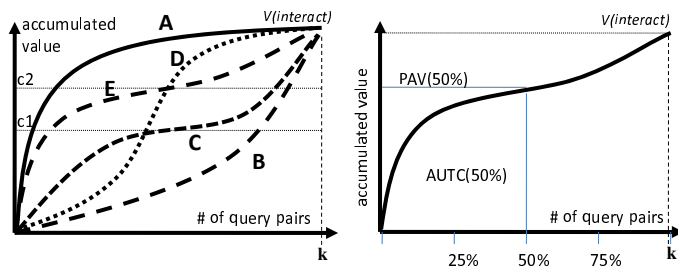Figure 2: (a) Value accumulated as a function of the number of queries sent; (b) $PAV\alpha$ and $AUTC\alpha$ metrics.

In terms of accumulated weighted value, a sequence X

---

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

of query pairs dominates a sequence Y of query pairs over the interval I 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 $Y \ \forall j \in I$. 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 2(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.

The only sequence that always dominates all other sequences is the unrealizable gold standard sequence. To compare the effectiveness of different sequencing heuristics performance measures should reflect how close the sequences produced using the heuristic are to the gold standard. For the ASA problem we consider, 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 this paper we use three types of metrics to evaluate a heuristic's effectiveness in ordering the queries sent to the scheduler. The first metric, which we call First Moment (FM), is calculated by Equation 3,

$$FM = \sum_{i=1}^{k} (k - i) \sum_{j=1}^{i} F_j P(o_i^M) \qquad (3)$$

In Equation 3, the weighting of the summed differences decreases as the number of query pairs executed for obtaining them increases. This metric assesses the requirement of giving more emphasis to values accumulated early in the sequence.

The other two metrics, which are illustrated in Figure 2(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 expected value of interacting with the user, $V(interact, t)$) 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 $V(interact, t)$ are made. The greater the $PAV\alpha$ value, the better the performance of the heuristic. The $AUTC\alpha$ measure gives some indication of the shape of the curve up to the α percentile of queries (and, in particular, the concavity, which is a primary indication for 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 along time, focus on initial stages and time to reach a percentile of the overall value. They thus 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 heuristic'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 heuristic for early elimination of non-fruitful interactions (by constructing an upper bound for the value of information as illustrated in Figure 1).[4]

Heuristics

Prior work [17] has investigated three heuristics for calculating the value of information in scheduling settings where the CI is separated from the scheduling expertise: *Duration Order*, *Relevant Change* and *Game*.

Duration Order: This heuristic generates queries in pairs in ascending order of the duration outcomes defined for the task.

Relevant Change: This heuristic generates the first query in the pair (based on knowing the outcome at time t). It goes over the schedule generated and identifies changes in schedule that cannot be applied otherwise due to the constraints imposed while executing tasks according to the schedule $S_t(T)$ during the interval $(t, t + o_i^M.duration)$. Then it sends the second query in the pair only for those outcomes associated with scheduling changes that (necessarily) cannot be made if the outcome is not received a priori.

Game: This heuristic considers the outcomes as placed in an N dimensional outcome space, where N is the number of characteristics defining an outcome. The problem is then modeled as a game, with distinct game and scoring rules. Then it attempts to solve the game optimally, based on isolating sequences of outcomes in the outcome space that contribute no value to the weighted sum.

The *Relevant Change* and *Game* heuristics have in common a focus on identifying outcomes associated with zero-valued contributions to the weighted sum in Equation 1. If such query pairs can be identified before they are sent then the execution of these queries can be saved, resulting in fewer computational resources being used to calculate $V(interact, t)$. However, their effectiveness in enhancing the value accumulation rate at early stages has never been tested.

This paper proposes two new heuristics aimed at enhancing value accumulation by prioritizing outcomes that are more likely to result with substantial values on each stage of the querying process:

Greedy Crawler: This heuristic also considers the outcomes as placed in an N dimensional space. It starts by executing

a query pair for the outcome having the highest probability of occurrence. It then crawls along the duration axis, querying the value of outcomes that are the consecutive over this axis (i.e., having same values for all other outcome characteristics) until either the difference between the expected quality of the two queries forming the pair is zero or the maximum possible duration along the duration axis is reached.[5] The idea is that since the F_i values are weighted in $V(interact, t)$ according to the probability of the outcome, moving between outcomes with greater probabilities may be beneficial.

Distance Crawler: This heuristic generalizes an approach described in earlier work for identifying zero-valued outcomes in the context of an N-dimensional outcome space [17]. For example, if knowing a priori that a task's duration is d_i does not affect the active schedule, and the same holds for duration d_j ($d_j > d_i$), then the same holds for any duration d_l ($d_i < d_l < d_j$). This phenomena is discussed in detail elsewhere and illustrated for settings where outcomes are characterized by duration and quality characteristics [17]. In general, if two outcomes have identical values for all their outcome characteristics except for one characteristic j (formally, $v_i^{o_1^M} = v_i^{o_2^M}, \forall i \neq j$) and the added value of knowing each of these outcomes a priori was calculated to be zero, then it is possible that the value of knowing a priori any other outcome o_l^M placed between the two along the j axis (i.e., satisfying $v_i^{o_1^M} = v_i^{o_l^M}, \forall i \neq j$ and $min(v_j^{o_1^M}, v_j^{o_2^M}) < v_j^{o_l^M} < max(v_j^{o_1^M}, v_j^{o_2^M})$)) is also zero. This will happen if the differentiating value (the value of characteristic j) is of a characteristic that has a consistent effect over the schedule quality for an increase or a decrease in its value.

The *Distance Crawler* attempts to predict the relative added-value magnitude of different outcomes according to their place in the outcome space and uses this prediction to order queries. It measures the magnitude of the added-value of knowing an outcome as the distance of the outcome from a hypothetical central outcome in the outcome space.[6] Outcomes with extreme characteristic values (in comparison to the hypothetical central outcome used by the scheduler) are more likely to account for substantial added-value. After each query and scheduler response, *Distance Crawler* attempts to determine if there are additional outcomes that can now be associated with zero added-value and removes them from the sequence. The following algorithm gives the overall flow of the Distance Crawler heuristic.

---

[4]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.

[5]The decision to crawl along the duration axis is based on the idea that the duration characteristic is a mandatory and inherent characteristic of tasks in the scheduling domain and is usually the most influential characteristic on scheduling constraints.

[6]The scheduler assigns a hypothetical central outcome any time a task has several possible outcomes, because scheduling tasks with several possible outcomes result in an exponentially large number of possible schedule qualities, which are difficult for schedulers to handle. In most cases, the values of this outcome's characteristics are set as the mean of the original outcome's possible values [20, 4].

Input: O: array with k possible outcomes, where $O[i].v^j$ is the value of characteristic j of outcome i; $Cost$: the cost of interrupting the user;

Output: V: the expected value of interacting with the user;

1: Set $V = 0$;
2: Set $O[i].val^7 = \sqrt{\sum_{j=1}^{N}((O[i].v^j - Avg_j)/(max_j - min_j))^2}$, $\forall 1 \leq i \leq k$;
3: Set $O[i].flag = false$, $\forall 1 \leq i \leq k$;
4: **while** $\exists\, O[i].flag == false$ **do**
5:    Set $i = ArgMax_i(O[i].val * O[i].prob)$;
6:    Set $O[i].marginal = Query(O[i])$; $O[i].flag = true$;
7:    $V = V + O[i].marginal * O[i].prob$
8:    **if** $V > Cost$ **then**
9:      **return** V
10:   **end if**
11:   **if** $O[i].marginal == 0$ **then**
12:     For all (j, l, k) satisfying $(O[j].flag == true)\&\&$ $(O[j].marginal == 0)\&\&$ $(O[k].v^m == O[i].v^m, \forall m \neq l)\&\&$ $(O[k].v^m == O[i].v^m, m \neq l)\&\&(min(O[i].v^l, O[j].v^l) \leq O[k].v^l \leq max(O[i].v^l, O[j].v^l))$, Set: $O[k].marginal = 0$; $O[k].flag = true$; $O[k].val = 0$
13:   **end if**
14: **end while**
15: **return** V

The algorithm stores the accumulated value in the variable V. It first calculates a weighted distance for each outcome from a central weighted point in the outcome space (Step 2). This value (stored in the field *val* for each outcome) is the main indication for the potential influence that knowing an outcome a priori will have on schedule quality. The distance is calculated based on the method used by the scheduler for tasks with several possible outcomes. In our case, the calculation is according to mean values as this is the most common method used by schedulers [4, 20], however if a different method is used (such as relying on the minimum or maximum values) then the distance calculation can be changed accordingly. At each step, the heuristic chooses the outcome with the highest *val* value (step 5) and calculates its value for receiving a priori this information (step 6). It then attempts to identify outcomes bounded by two zero-valued outcomes in the outcome space (Steps 11-13), in which case their value is zero, for the reasons given above. These latter outcomes are assigned the value zero (Step 12). The heuristic terminates when either: (a) the accumulated value exceeds the cost of interrupting the user (step 8); or (b) the added-value of all outcomes was incorporated in the calculation. This latter case is supported by setting the *flag* field of each outcome to true once the value of knowing this outcome is assigned to the *val* field.

The *Greedy Crawler* and the *Distance Crawler* heuristics have several properties worth highlighting. First, they reflect the goal of finding out as soon as possible whether an interaction should be initiated. Second, the use of these heuristics does not change the correctness of the information value calculation and the resulting interruption decision. As described above, if necessary all of the query pairs will be

---

^{7}Where: $Avg_j = \sum_{l=1}^{k} O[l].v^j O[l].prob$, $max_j = max(O[i].v^j | 1 \leq i \leq k)$ and $min_j = min(O[i].v^j | 1 \leq i \leq k)$.

used and the same value reached as the calculation without the heuristics. Third, they execute in polynomial time (linear in the number of possible outcomes). Thus, if either of these heuristics 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 heuristics.

Experimental Evaluation

The five heuristics described in the previous section were implemented and tested using the DARPA *Coordinators* test bed. The *Coordinators* project aims to construct intelligent cognitive software agents able to assist fielded military units to adapt their mission plans and schedules as their situations change [13, 16]. In the *Coordinators'* application domain [21], the ASAs, called "coordinators", operate in a rapidly changing environment and are intended to help maximize an overall team-objective. Each ASA operates on behalf of an *owner* whose schedule it manages. Examples of owners are a team leader of a first-response team or a unit commander. The actual tasks being scheduled are executed either by owners or by units they oversee, and the ASAs' responsibility is limited to scheduling these tasks. In *Coordinators* environments, owners may acquire relevant external information through various channels and methods [17].

In this domain, scheduling information and constraints are distributed. Each ASA has a different partial view of the tasks and structures that constitute the full multi-agent problem, and scheduling problems must be solved distributively. The ASA must reason about changes in the timings of tasks with regard to their interaction with other ASAs owners' tasks and to adapt its owner's schedule accordingly. Several architectures have been suggested for ASAs in these domain [16, 13, 20], all of which share a core set of modules. The modules for which the value accumulation problem is relevant are the CI module (named "coordination autonomy" (CA)) and the scheduler. The CI module is responsible for deciding intelligently when and how to interact with the owner to improve the ASA's scheduling. The scheduler is responsible for task analysis. Thus the scheduler is a resource used by the CI module whenever scheduling reasoning is required for evaluating the effect of changes in problem settings on the system's expected performance. Thus, *Coordinators* provides an example of a settings in which the scheduler is separated from the CI.

In *Coordinators*, the planning and scheduling problems the ASAs are engaged in are represented by structures (called cTAEMS) that define multi-agent hierarchical tasks with probabilistic expectations on their outcomes [13]. The atomic elements composing a task are called methods and represent actions executed by a single ASA. A method has multiple possible discrete outcomes that determine its possible durations and quality (i.e., a bi-dimensional outcome space).

In general, the quality characteristic of a method represents the contribution of performance of that action to the overall team effort and is execution-dependent. Methods are usually constrained by "release times" (earliest possible
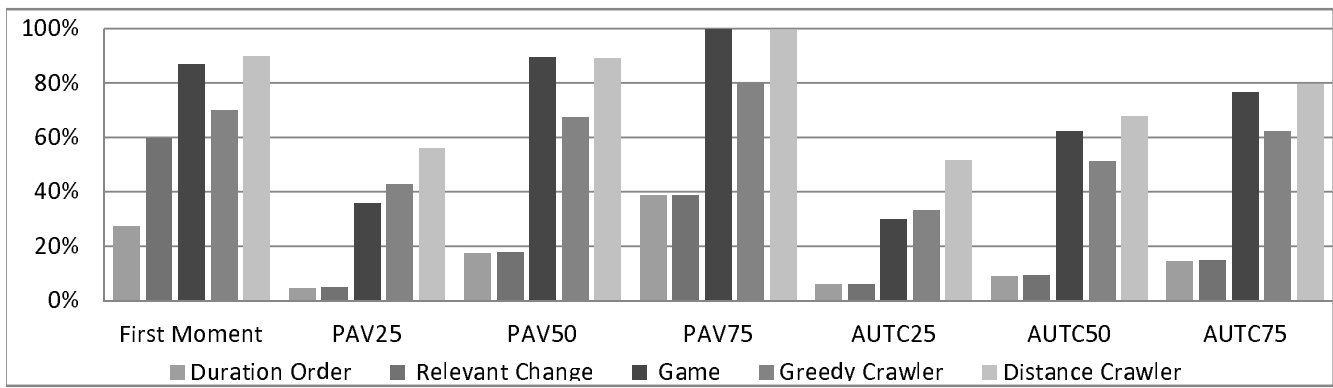
Figure 3: Performance using different metrics.

start) and deadlines that are set by tasks higher in the hierarchy to which a method belongs, and inherited hierarchically. Each ASA's schedule defines a set of methods it plans for its owner's unit to execute, where at any time, each ASA can schedule execution of at most one of its methods. The *Coordinators* domain and its use of cTAEMS structures have become important infrastructure in scheduling and coordination related research [1, 3, 21].

Our evaluation used 2093 cTAEMS problems that were originally used by DARPA as a test suite for evaluating different ASA architectures. These problems were divided into types based on parameters such as the scale of the required scheduling task, the number of agents, and the interdependencies between tasks. Each type represents different problem characteristics and complexity. Other than the advantages of using *Coordinators* problems as an easily generalized scheduling problems, our adoption of this evaluation methodology makes our experimental design consistent with evaluations testing prior heuristics [17].[8]

For each test case, the CI picked a random snapshot of the schedule being executed and activated its utility estimation mechanism for the method being executed at that time. The utility estimation process was repeated several times, each time using a sequence of queries generated by a different tested heuristic (*Duration Order*, *Relevant Change*, *Game*, *Greedy Crawler* and *Distance Crawler*). Then, the gold standard sequence was extracted and its value was computed for each metric.

Figure 3 depicts the performance of each heuristic using the three metrics: First Moment (FM), Percentage of Accumulated Value ($PAV\alpha$) and the Area Under the Truncated Curve ($AUTC\alpha$). 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 different heuristic had to be constructed. Therefore, performance of each heuristic in any given cTAEMS problem was calculated as the percentage it managed to achieve out of the value achieved when using the gold standard sequence. This latter value is used for the vertical axis in Figure 3 (i.e., the 100% is the upper performance level that can be obtained, in which case the value obtained is equal to the value of using the gold standard sequence).

As reflected in Figure 3, the *Distance Crawler* heuristic dominates all other methods according to the FM and the $AUTC$ metrics. It also dominates the other methods for the PAV metric when using the 25 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 *Distance Crawler* and the *Game* heuristic for the $PAV50$ and $PAV75$ metrics. Therefore the *Distance Crawler* heuristic can be considered to weakly dominate the *Game* heuristic. The dominance of the *Distance Crawler* heuristic over the other methods when using the $PAV25$ metric is especially important since this result relates to the initial and most important stages of the process, i.e., 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 spared. It is notable that the *Distance Crawler* heuristic manages to accumulate more than 50 percent of the value accumulated by the gold standard by the 25 percentile of queries. Overall, the results suggest that the *Distance Crawler* heuristic is the best choice for constructing the value of information in reasoning about the usefulness of interruption.

The results also provide the following three new comparative assessments of the heuristics proposed in prior work [17]: (a) Of the three earlier heuristics, the *Game* is the most efficient in terms of value accumulation (cross-metric); (b) Surprisingly, the *Relevant Change* heuristic does not improve the value accumulation rate in comparison to the "naive" Duration Order method, despite its advantage in identifying zero-valued outcomes. The one exception is with

---

[8]To fully match with the evaluation used in prior work [17], we used the same augmentation of the outcome distribution used there. This augmentation, which does not change the mean and extreme values of the different outcomes, is necessary for extending the outcome space above the small number of outcomes (3) typically found in the test suite.
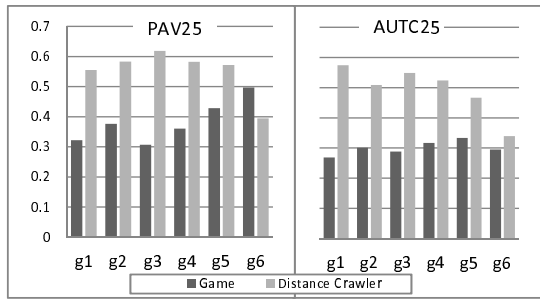
Figure 4: *Game* and *Distance Crawler* comparison.

the FM metric; (c) In the important initial stages of the process (PAV25 and AUTC25), the simple Greedy heuristic performs better than the *Game* heuristic.

To further examine the relative performance of the *Game* and the *Distance Crawler* heuristics, we used the groupings of problems in the original set as described above, resulting in six types of problems, each of which represents different problem characteristics and complexity. Figure 4 gives the performance differences between these two heuristics for each problem group, using the PAV and $AUTC$ measures for the 25 percentile. As the graphs show, the performance improvement achieved by the *Distance Crawler* heuristic does not depend on any specific problem characteristics, with one exception. For group 6 when the $AUTC$ measure is used, the *Game* heuristic performs slightly better on average. This anomaly may be explained by the unique characteristics of the problems in group 6: It has relatively many facilitation relationships, causing these problems to be associated with a small value of information in the first place. (A significantly small number of problems (13) in this group had a positive value to begin with.)

Related Work

Value of information (VoI) is one of the most useful notions in decision analysis [8, 15]. It is commonly defined as the value of a decision in a situation with perfect information minus the value of the decision made with only partial knowledge [10]. Most research on value-driven information gathering systems [5] is on the development of autonomous agents operating in an information rich domain under time and monetary resource restrictions. In these settings, the key question is what information to collect based on an explicit representation of the user's decision model and a database of information sources. The main idea in these systems is to select which information sources to query based on the marginal value of a query. This marginal value is usually derived from a utility function based on the user's preferences in combination with information provided by the expert who constructed the decision model. Unlike in these settings, for the problem we address, the system cannot predict a priori the marginal value of each query. Instead, the value can only be derived by applying complex scheduling reasoning for which the system must rely on an external scheduler.

The need to calculate the value of information has been widely addressed in interruption management research where agents are required to make interruption decisions in dynamic environments with incomplete information [6, 2]. Within the framework of the decision-theoretic techniques used in this related work, the value of information supplied to the user derives from the alternative actions a user will undertake as a result of receiving new information relevant to her work [7, 9]. The focus in these models is almost entirely on modeling the user's attentional state and avoiding unnecessary or unhelpful interruptions and the ensuing frustration they cause [9, inter alia]. Furthermore, this work assumes that users will change their actions in a pre-determined manner when they receive the new information and that it is straightforward to calculate the benefit from performing the alternative action. Finally, unlike the setting the current paper deals with, in prior work on interruption management there is no need to deal with uncertainty about the information (i.e., to consider different types of information the user has) because it is the agent that has information needed by the person rather than the reverse.

Research on autonomous agents providing assistance to human planners by providing information alerts at appropriate times [22, interalia] derives the value of alerting users with new information from its usefulness for their decision making. In these domains, the focus has been mainly on the person side of the human-computer interaction, rather than the efficiency of the process by which the value of information is calculated.

All this prior work differs from ours in three key ways: (1) the value of information either was assumed known or could be calculated simply; (2) the problem of efficiently obtaining the value of information is not an issue; and, (3) resource constraints are not considered.

Discussion and Conclusions

Generating queries in the appropriate sequence is crucial for reasoning about interrupting a user in collaborative fast-paced environments with computational resource constraints. A good ordering facilitates an intelligent decision early in the process, thus improving efficiency. The uniqueness of the approach presented in this paper is that it is fully targeted towards extracting an effective sequence of queries, which yields most value for any number of queries used, rather than just finding the minimal set of queries required for calculating the value of information. While the paper considers this problem in the context of a computer agent interrupting a person, the analysis is applicable to any scenario in which an agent needs to reason about interrupting a teammate, whether that teammate is a human or computer agent.

The paper introduces two new heuristics for situations in which reducing the computational load in computing the value accumulation matters. These are evaluated based on a comprehensive set of problems used to test ASAs in the *Coordinators* program. Both heuristics outperform heuristics proposed in prior work whenever it is important to accumulate value as early in the query sequence as possible (i.e., for $\alpha = 25$). Furthermore, the *Distance Crawler* heuristic

dominates all the heuristics we tested for all metrics. The uniqueness of this heuristic is in its ability to identify outcomes which are most likely to account for substantial value while eliminating outcomes that do not lead to any schedule changes and thus are not relevant for determining the value of information. The use of three types of metrics, each emphasizing different desirable aspects of the heuristics, in the evaluation provides a comprehensive evaluation methodology.

Acknowledgement

References

[1] G. Emami, J. Cheng, D. Cornwell, M. Feldhousen, C. Long, V. Malhotra, I. Starnes, L. Kerschberg, A. Brodsky, and X. Zhang. Active: agile coordinator testbed integrated virtual environment. In *AAMAS '06*, pages 1580–1587, 2006.

[2] M. Fleming and R. Cohen. User modeling in the design of interactive interface agents. In *UM'99*, pages 67–76, 1999.

[3] A. Raja G. Alexander and D. Musliner. Controlling deliberation in a markov decision process-based agent. In *AAMAS'08*, forthcoming, 2008.

[4] C. Gomes, W. van Hoeve, and B. Selman. Constraint programming for distributed planning and scheduling. In *AAAI Spring Symp. on Distributed Plan and Schedule Management*, 2006.

[5] J. Grass and S. Zilberstein. A value-driven system for autonomous information gathering. *J. Intell. Inf. Syst.*, 14(1):5–27, 2000.

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

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

[8] R. Howard and J. Matheson, editors. *Readings on the Principles and Applications of Decision Analysis*. Strategic Decision Group, Menlo Park, CA, 1984.

[9] B. Hui and C. Boutilier. Who's asking for help?: a bayesian approach to intelligent assistance. In *IUI '06*, 2006.

[10] H. Kuhn. Extensive games and the problem of information. In H.W. Kuhn and A.W. Tucker, editors, *Theory of Games II*, pages 193–216. 1953.

[11] R. Maheswaran, P. Szekely, M. Becker, S. Fitzpatrick, G. Gati, J. Jin, R. Neches, N. Noori, C. Rogers, R. Sanchez, K. Smyth, and C. VanBuskirk. Predictability & criticality metrics for coordination in complex environments. In *AAMAS'08*, forthcoming, 2008.

[12] W. McClure. Technology and command: Implications for military operations in the twenty-first century. Maxwell Air Force Base, Center for Strategy and Technology, 2000.

[13] D. Musliner, E. Durfee, J. Wu, D. Dolgov, R. Goldman, and M. Boddy. Coordinated plan management using multiagent mdps. In *AAAI Spring Symp. on Distributed Plan and Schedule Management*, 2006.

[14] K. Myers, P. Jarvis, and T. Lee. Active coordination of distributed human planners. In *AIPS'02*, 2002.

[15] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[16] J. Phelps and J. Rye. Gpgp a domain-independent implementation. In *AAAI Spring Symp. on Distributed Plan and Schedule Management*, 2006.

[17] D. Sarne and B. Grosz. Estimating information value in collaborative multi-agent planning systems. In *AAMAS'07*, pages 227–234, 2007.

[18] D. Sarne and B. Grosz. Sharing Experiences to Learn User Characteristics in Dynamic Environments with Sparse Data. In *AAMAS'07*, pages 202–209, 2007.

[19] 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 *AAMAS'03*, pages 433–440, 2003.

[20] S. Smith, A. Gallagher, T. Zimmerman, L. Barbulescu, and Z. Rubinstein. Distributed management of flexible times schedules. In *AAMAS'07*, pages 1–8, 2007.

[21] T. Wagner, J. Phelps, V. Guralnik, and R. VanRiper. An application view of coordinators: Coordination managers for first responders. In *AAAI*, pages 908–915, 2004.

[22] D. Wilkins, S. Smith, L. Kramer, T. Lee, and T. Rauenbusch. Airlift mission monitoring and dynamic rescheduling. *Engineering Applications of Artificial Intelligence*, 2007.