Quantifying the Expected Utility of Information in Multi-Agent Scheduling Tasks

Avi Rosenfeld[1,2], Sarit Kraus[2], and Charlie Ortiz[3]

[1] Department of Industrial Engineering
Jerusalem College of Technology, Jerusalem, Israel
[2] Department of Computer Science Bar Ilan University, Ramat Gan, Israel
[3] SRI International, 333 Ravenswood Avenue Menlo Park, CA 94025-3493, USA
Email: {rosenfa, sarit}@cs.biu.ac.il, ortiz@ai.sri.com

Abstract. In this paper we investigate methods for analyzing the expected value of adding information in distributed task scheduling problems. As scheduling problems are NP-complete, no polynomial algorithms exist for evaluating the impact a certain constraint, or relaxing the same constraint, will have on the global problem. We present a general approach where local agents can estimate their problem *tightness*, or how constrained their local subproblem is. This allows these agents to immediately identify many problems which are not constrained, and will not benefit from sending or receiving further information. Next, agents use traditional machine learning methods based on their specific local problem attributes to attempt to identify which of the constrained problems will most benefit from human attention. We evaluated this approach within a distributed cTAEMS scheduling domain and found this approach was overall quite effective.

1 Introduction

Effectively harnessing the relative strengths of mixed human agent groups can be critical for performing a variety of complex scheduling tasks. The importance of effective coordination has been demonstrated in scheduling and planning domains such as hazardous cleanup, emergency first-response, and military conflicts [12]. Agents can quickly compute possible group compositions and assess group behavior even in dynamic and time sensitive environments [5, 11, 14]. This ability can be invaluable in focusing a person's attention to the most critical decisions in these environments [12].

We present the challenge of how agents can best coordinate their support decisions with people as the "Coordination Autonomy" (CA) problem. While using computer agents in these tasks can be beneficial, they are subject to several key limitations. First, we assume agents have only a partial view of the global constraints and the utility that could potentially be achieved through fulfilling these tasks. Because of this, it is impossible for agents to compute the group's utility

exclusively based on their local information [5, 11]. Second, we also assume there is cost associated with sending or receiving constraint information. These costs may stem from agent communication costs, or costs associated with interrupting the human operator [12]. As has been previously observed, coordinating decisions involving incomplete information and communication costs significantly increases the problem's complexity [9].

In the CA system under consideration, human operators are assumed to have access to more complete knowledge about task uncertainty because they have updated information or expert knowledge. This information can be critical in improving the group's utility by relaxing agents' rigid constraint information [12]. Specifically, we focus on how this information may affect decisions where uncertainty exists in task quality and duration. For example, a human operator in an emergency response environment may have updated information about domain weather conditions, or acquired knowledge from years of experience. Without this information, agents must plan for the worse-case scenario, e.g., that tasks with uncertainty in quality will yield the lesser utility amount, and tasks with uncertainty in duration will take the longest time. However, the user may know what task outcomes will actually be, allowing for more or higher quality tasks to be scheduled. On the one hand, this information can be invaluable in increasing the group's productivity. On the other hand, a person's time is valuable, and thus the person should only be consulted if the agent believes that the current scheduling problem is such that additional information will help.

Effectively measuring the expected value from a teammates' information is critical to the success of these types of systems [12]. Similar issues have arisen in the Information Gain measure that has been put forward by the machine learning community [6]. Information gain is typically used to learn the effectiveness of a certain attribute in classifying data, or how much additional information is gained by partitioning examples according to a given attribute. Thus, one approach may be to query the system with and without a given piece of information and build the CA application based on the resulting Information Gain.

However, there are several reasons why basic machine learning measures such as Information Gain cannot be applied to this type of problem. First, we assume there is a cost involved with agents exchanging constraint information. As a result, the cost in computing the Information Gain for a given problem may outweigh its benefit. Second, since there are a potentially large number of tasks to schedule, the size of the learning space will be very large. Sending "what if" queries for each task to a local scheduler can become resource intensive, leading to delays. Finally, sending too much constraint information can result in a lower group quality: we have previously found that in highly constrained problems, sending additional information can prevent agents from finding the optimal solution [8].

Towards addressing this problem, this paper presents an approach where agents can estimate the value of information without the resource intensive queries used in other works [12]. Our first contribution is a general, non domain-specific constraint "tightness" measure in which agents can locally measure how

constrained a task is. A constraint tightness of less than one indicates that a task is underconstrained and will not benefit from any additional information. This allows agents to locally and immediately identify that the expected utility from added information in such cases will be zero. A tightness measure of greater than one indicates that the problem **may** be affected by other constraints, and thus information **may** be of importance. However, we found the problem of determining how constrained the problem is exclusively from local information to be quite challenging. In addressing this challenge, we present a solution where agents can offline apply machine learning techniques to identify which of the remaining tasks are likely to have the highest expected utility from information.

Our next section provides the background and motivation for this work. In Section 3 we briefly describe the cTAEMS language used in quantifying the distributed scheduling tasks we studied. In Section 4 we present the domain independent constraint "tightness" measure for locally assessing what the utility of added information will be. Section 5 present how we quantified this value of information through machine learning regression and decision tree models. Section 6 provides experimental results. In Section 7, we provide a discussion about the general applicability of these results, as well as provide several directions for future research. Section 8 concludes.

2 Related Work

The goal of this paper is to quantify the expected utility to be gained from added information in distributed scheduling problems. This information can then be used to help agents decide what constraints should be forwarded to a human user for further information. Thus, this paper is linked to previous research in the field of agent-human interactions, as well as previous distributed scheduling research.

The adjustable autonomy challenge as described by Scerri et al. [13] refers to how agents can vary their level of autonomy, particularly in dealing with other types of entities such as people. They proposed a framework where agents can explicitly reason about the potential costs and gains from interacting with other agents and people to coordinate decisions. A key issue to applying their model within new domains is effectively quantifying the utility of information the entities can provide – a challenge we address in this paper.

Distributed scheduling problems belong to a more general category of distributed constraint optimization problems (DCOP) [11], and a variety of algorithms have been proposed for solving these problems [5, 11, 14]. In these problems, inter-agent constraints must be coordinated to find the solution that satisfies as many of these constraints as possible. However, these problem are known to be NP-complete, even if agents freely share all information at their disposal [5, 9]. As such, no polynomial algorithms exist for checking how a scheduling problem's utility is affected by a given piece of information.

This paper presents a process through which agents are able to successfully quantify inter-agent interactions, such as the expected utility of sending or

receiving information, exclusively with the local agent's information. Previous approaches have considered all possible group interactions, potentially creating a need to generate very large numbers of "what if" queries to obtain this information [9, 12]. Our approach is significant in that agents locally estimate the utility of additional information without additional data, allowing them to reason about a limited number of interactions. This reduction allows for tractable solutions in estimating the value of information without using any queries during task execution.

In creating our approach, we draw upon previous studies that found that different categories of problem complexity exist, even within NP-complete problems [2, 7]. Many instances of NP-complete problems can still be quickly solved, while other similar instances of problems from the same domain cannot. These studies have introduced the concept of a phase transition to differentiate classes of "easy" and "hard" instances of a problem [7]. Based on that body of work, we attempt to locally identify tasks which are underconstrained, or represent "easy" interactions that can be locally solved without any additional information, as well as the "hard" interactions that can potentially benefit from additional information.

However, finding such phase transitions within real-world domains is far from trivial due to the varied types of possible agent interactions [1, 9]. In the theoretical graph coloring problems previously studied, phase transitions were found that were associated with the ratio of constraint clauses per variable [7]. Unfortunately, these graph coloring problems are relatively simple in that all constraints typically have equal weighting and every agent has equal numbers of constraints (edges). Thus, discovering phase transitions in experiments can be accomplished only through variation of a single parameter – the ratio of graph edges to nodes. In contrast, as we now describe, many real-world domains, such as the cTAEMS scheduling domain we have focused on, are far more complex. Novel measures are needed to quantify inter-agent actions in this and similar domains.

3 Domain Background and Description

cTAEMS is a robust, task independent language used by many researchers for studying multiagent task scheduling problems [12] based on the TAEMS language standard [4]. The cTAEMS language is composed of methods, tasks and subtasks that define a coordination problem in a Hierarchical Task Network (HTN) structure. Methods represent the most basic action an agent can perform and have associated with them a list of one or more potential outcomes. This outcome list describes what the quality (Q), duration (D) and cost (C) distributions will be for each possible result associated with the execution of that method. Tasks represent a higher level abstraction and describe the possible interrelationship between actions through what is referred to as a quality accumulation function (QAF), that indicates how the expected quality of subtasks will contribute to the overall quality of a (group) task. QAF's are of three forms: min, max or sum. In a min QAF, the total added quality is taken to be

the minimum of all subtasks – it could be thought of as a logical AND relation between tasks. In a max QAF, the quality is the maximum value (or the logical OR), whereas in a sum QAF the quality is the sum of the expected quality of all subtasks. These subtasks can then be further subdivided into additional levels of subtask children, each potentially with their own QAF relationship. Finally, hard constraints between tasks or methods can be modeled in terms of what are referred to as non-local effects (NLE's) constraints between tasks, using the primitives *enable* or *disable*. Soft constraints are modeled through *facilitates* or *hinders* relationships. For example, assuming one task must occur before another, one could represent this constraint in terms of an enables relation between those two tasks. Assuming two tasks (or subtasks) cannot both be performed can be modeled in terms of a disables relation between the two tasks. System dynamics are modeled through probabilistic values for quality and duration within the HTN's tasks and subtasks. For example, a given task could have a duration of 10 with 50% probability, a duration of 4 with 25% probability, and a duration of 20 with 25% probability.

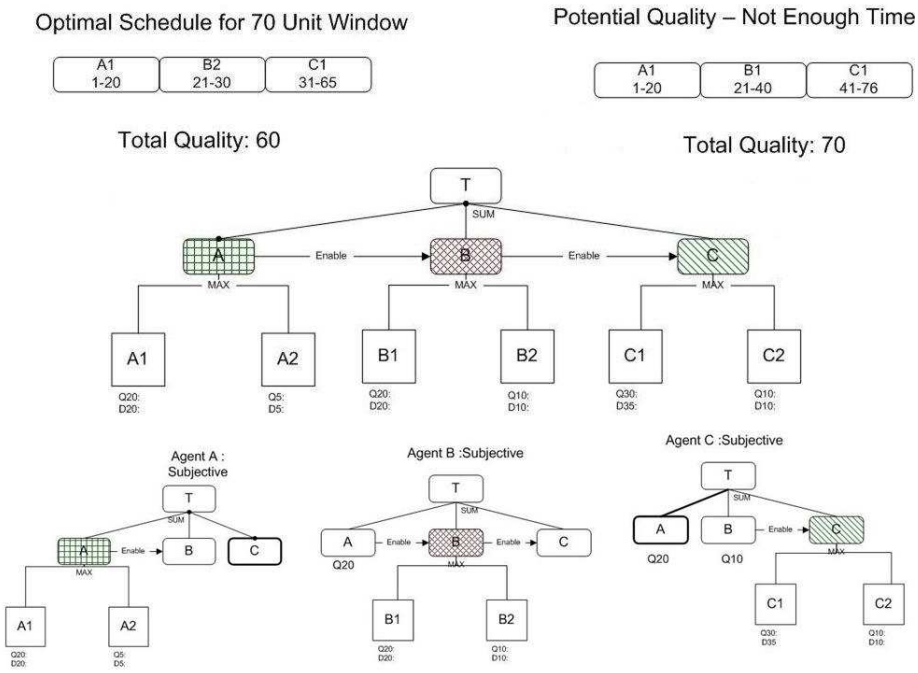

Fig. 1. A sample cTAEMS Scheduling Problem (global view middle) with 3 agents (3 subjective views on bottom).

The CA system we propose takes as its input the system's distributed constraints, formulated in cTAEMS, and outputs the constraints a person should focus on. For example, Figure 1 is an example of a scheduling problem instance,

described in cTAEMS. In this example, three agents, A, B, and C must coordinate their actions to find the optimal schedule for a global task T. Task T has three subtasks (A, B, and C) and these tasks are joined by a sum relationship. There are enable relationships between these tasks and thus they must be executed sequentially. In this example, an optimal schedule would be for A to schedule method A1, B to schedule B2, and C to schedule C1. However, assuming only 70 time units exist for all three tasks, there is insufficient time for A to schedule A1, for B to schedule method B1, and for C to schedule C1. As such, one of the agents must sacrifice scheduling its method with the highest quality so the group's quality will be maximized. The group will lose 15 units of quality if A does not schedule A1, 10 units of quality if B does not schedule B1, and 20 units of quality if C does not schedule C1. Thus, B chooses B2 so the A1 and C1 can be scheduled.

Changing the cTAEMS structure, even slightly, can greatly affect the inter-agent constraints. We particularly focus on the impact uncertainty in task quality and duration will have on decisions. If a given task has a distribution of possible durations, agents supporting the automated scheduling process assume the worst-case scenario must be planned for (e.g. we must assume the task will have the smallest possible quality, or the task will take the longest possible time). Adding more precise information, such as eliminating certain duration possibilities, or identifying which outcome will definitely occur, can also greatly impact inter-agent constraints. For example, if a human operator could provide information that task B1 will take less than 15 units of time (instead of the current 20) this task could be scheduled and the group's utility will be raised. As a result, the CA system should identify task B1 as being the first task worthy of the person's attention. Novel mechanisms are required to generally find which types of constraints are most worthy of further information.

4 Locally Quantifying Scheduling Constraints

While we used the cTAEMS language to quantify scheduling constraints, the solution we are developing is meant to be as general as possible. Towards this goal, we have developed a tightness measure which we use to identify which subtasks are not constrained, and therefore cannot possibly benefit from any additional information. Our hypothesis is that general types of interactions can be quantified, similar to the phase shifts found within simpler graph coloring optimization problems previously studied [2, 7]. However, novel measures of interaction difficulty are needed to help quantify interactions so phase shifts can be discovered.

Towards this goal, we present a "tightness" measure to quantify how much overlap exists between constraints. In referring to these constraints, let $G = \{A_1, A_2 \ldots, A_N\}$ be a group of N agents trying to maximize their group's collective scheduling utility. Each agent has a set of m tasks, $\mathrm{T} = \{T_1, \ldots, T_m\}$ that can be performed by that agent. Each Task, T_i, has a time **Window** within which the task can be performed, a **Quality** as the utility that the task will add to the

group upon its successful completion, and a **Duration** as the length of time the task requires to be completed. We assume that task quality and duration often have uncertainty, while task windows are typically based on the problem's structure. We model W_i as the fixed Window length for task T_i, $\{Q_{i1}, Q_{i2}, \ldots, Q_{ij}\}$ as the possible quality outcomes for T_i, and $\{R_{i1}, R_{i2}, \ldots, R_{ij}\}$ as the possible duration lengths of T_i.

Based on these definitions, we model an agent's quality *tightness* as:

Tightness-Quality$(T_i) = \frac{Quality_{max}(T_i)}{Quality(Window(T_i))}$

where $Quality_{max}(T_i)$ returns the maximal quality from $\{Q_{i1}, Q_{i2}, \ldots, Q_{ij}\}$, and $Quality(Window(T_i))$ returns the maximal expected quality of all **other** subtasks that share the task window of (T_i). Note that if quality uncertainty exists in these other tasks, we again assume the worse case, and the lowest quality value must be considered in computing the value of $Quality(Window(T_i))$. The measure of Tightness-Quality(T_i) can then be used to quantify what potential overlap exist between T_i and other local constraints within T_i's task window. For example, assume T_A can be fulfilled by methods A1 and A2 with A1 having a quality distribution between 10 and 20, and A2 having a quality distribution of 15 and 25. It is possible that asking the user for input about the actual quality of A1 is worthwhile as these quality distributions overlap and A1 may have a higher quality than A2 (say 20 for A1 and 15 for A2). In this example, Tightness-Quality(A1) = 1.33 indicating the quality distributions of these tasks overlap and information may be beneficial. However, a tightness under one would indicate no possible quality overlap, and thus no possible gain from information.

Similarly, we model an agent's duration *tightness* as:

Tightness-Duration$(T_i) = \frac{Duration_{max}(T_i)}{Duration(Window(T_i))}$

where $Duration_{max}(T_i)$ returns the maximal duration from $\{R_{i1}, R_{i2}, \ldots, R_{ij}\}$, and $Duration(Window(T_i))$ refers to the time allotted for completing task T_i. Note that within the cTAEMS problems we studied, no uncertainty existed within the time window for the tasks sharing a given window $(Window(T_i))$ so uncertainty in this value need not be considered.

As was the case within the quality tightness measure, a value of more than one indicates an overlap between T_i and other task constraints, while a value less than one indicates no possible overlap. For example, assume task T_A may last for either 10, 20, or 40 time units to be completed within a time window, $Window(T_A)$ of 25 units. According to the tightness definition, tightness Tightness(T_A) is 40/25 of 1.6. Without any additional information, we must assume that this subtask task will take the maximal time, potentially preventing that agent from performing other tasks. However, assuming information can be provided regarding the task's duration, say that T_A will only last for 10 or 20 units, the value of Tightness(T_A) drops below 1.

The tightness measures we present have two important properties: First, they are locally measurable. Agents can measure its quality or duration tightness without any additional input from other task agents or the human operators within the group. Second, they can effectively quantify the impact making a local decision will have. By definition, a tightness of 1.0 or less means that the

problem is not constrained, as no task option overlaps with others options sharing that task. In such cases, agents will not obtain additional utility from more information, and the agent should not prompt the human operator for additional data. After this measure exceeds 1.0, a phase shift occurs when information **may** be helpful. However, further solutions are still needed to quantify how much the group's utility is expected to increase if this constraint is relaxed.

5 Learning the Value of Information

As the tightness measure only addresses which tasks will definitely **not** benefit from additional information, the next step in our approach is a machine learning model to suggest which tasks **will** show an increase in utility as a result of additional information. In addressing this challenge, we built two machine learning models: a regression based model where agents predict a numeric value for added information from the human operator, and a classification model where agents classify a given task as potentially benefitting or not benefiting from additional information. Alternatively, within the classification model, qualitative categories can be created, such as High, Low, and Zero impact categories instead of binary Yes and No categories.

The human-agent interface within the Coordination Autonomy (CA) application we propose will be affected by the learning model chosen. Within the regression model, the CA front-end will present the human operator a numeric field for the expected value of information that can potentially be added through the user's attention. Assuming a classification model is trained, we propose that the CA's front-end color code various tasks to represent how much information can potentially help the group. For example, referring back to Figure 1, subtasks would be colored green if information was categorized as less important while subtasks of high importance would be colored red. We believe this interface can most effectively focus the user's attention.

The procedure we adopted for training the machine learning model is outlined in algorithm 1. We used a problem generator created by Global Infotech Inc. (GITI) for the purpose of generating cTAEMs problems within the framework of the COORDINATORS DARPA program[4]. We created two sets of 50 problems (the value for X in the algorithm) where cTAEMS parameters (such as the number of tasks to be scheduled, the hierarchical structure of the tasks, the number of agents able to perform each task, the number of NLE relationships between tasks, task duration, and average task quality) were randomly generated. In the first set of problems, we generated problems with uncertainty in quality distributions, while keeping other parameters deterministic. In the second set of problems, we created problems with uncertain durations, while keeping other parameters constant. Note that these 100 total problems represent a small fraction of the total number of the thousands of problem permutations the GITI scenario generator could create. Additionally, each of these base problems had

[4] http://www.darpa.mil/ipto/programs/coordinators/

many possible permutations – these 100 total cTAEMS problems contained over 5000 subtasks where constraint information could be added.

The goal for creating these test cases was to study how user information about quality and duration uncertainty affected the group's utility. We used a previously tested cTAEMS centralized scheduler [8] to first compute the group's utility under the assumption that uncertainty in problems would result in the lowest probabilistic outcome. Thus, in the first set of problems, we assumed tasks would have the lowest quality, and in the second problem set the tasks would take the maximal time (line 3). We then computed what the group's utility would be if we could relax that assumption, and the user could provide information that task would have the highest possible quality, or take the shortest time (line 4). Next, we stored this information into a table along with a vector of the problem's specific parameters (e.g. problem parameters such as tightness, local NLE's, maximal duration, quality, etc.) and entered this information into the table, Table (line 5). This problem information would then be used for offline training of the machine learning model.

In computing the value of added information, we adopted a highly optimistic approach for the value of Utility($Problem_k$,j) that makes several assumptions: a) the person being contacted actually has information about the subtask j, b) the person will have information that will help the group in the maximal possible way by informing the agents that the task will have the highest quality or take the shortest duration, and c) this information can be provided without cost. Despite this oversimplification, the approach was useful for identifying the maximal upper bound for the potential utility that could be gained through added information.

Algorithm 1 Training the Information Model(Problem Set of Size X)

```
1: for k = 1 to X do
2:    Without ⇐ Utility(Problem_k)
3:    for j = 1 to Num(Subtasks(Problem_k)) do
4:       With ⇐ Utility(Problem_k,j)
5:       Table[k][j] ⇐ (With) - (Without)
6:    end for
7: end for
```

Interestingly, we found that only a small percentage of subtasks had any benefit from adding this type of constraint information. While each problem, k, contained at least one subtask that did benefit from added information, only 1022 subtasks, or roughly 20% of the total entries within the training data, benefited from any additional information. Thus, finding these subtasks is akin to "finding a needle in a haystack". Clearly, naive methods that query every subtasks are not appropriate, especially if there is a cost associated with generating queries or if the human is not able or willing, for whatever reason, to provide information.

6 Experimental Results

We found strong support for the usefulness of the quality and duration tightness measures in identifying the cases where information definitely did **not** help. Of the 2490 cases where a tightness value was less than 1, only 38 (1.5%) cases benefited from additional information. Next, we used the Weka machine learning package [10] to train and evaluate what the value of information would be in the remaining cases. We present the results from training and evaluating three decision tree models: a regression model based on the M5P algorithm and C45 decision trees (J48 within the Weka implementation) to create classifier models based on 2 information categories (Yes / No impact of information) and a 3 category information classification task (High, Low, and Zero impact). Note that while we present results from decision trees learning approaches, other possibilities exist. We did, in fact, train models based on Bayes Networks, Neural Networks, and SVM models and found the results to be nearly identical with those we present. In all cases, we performed 10-fold cross validation to evaluate the results.

First, we trained and evaluated the regression based model. The results from this experiment are found in Table 1. The first two rows present the results from the problem set with quality uncertainty, and the last two rows present the results from the corresponding problem set with duration uncertainty. Note that this model yielded an average correlation of 0.56 and 0.46. In comparison, we present the Naive approach which assumes all instances belong to the majority class, and information adds zero quality. While these results are certainly significant (0.56 and 0.46 being much larger than the Null hypothesis of 0.05), they do leave room for improvement as even with the tightness measure these results are far from the optimal correlation of 1.0.

Table 1. Comparing the accuracy and Mean Absolute error in regression trained model.

	Learned Model	Naive (Majority)
Quality Correlation	0.56	-0.06
Quality Mean Absolute Error	1.27	2.61
Duration Correlation	0.46	-0.04
Duration Mean Absolute Error	1.67	2.00

Next, we trained a two category classification model (Yes/No categories) to find instances where information does or does not help. Table 2 presents the results. Note that the larger class (information did not help) represents over 80% of the subtasks in both problem sets, and thus even naively categorizing all subtasks within this category results in a relatively high accuracy of the model. However, as the goal is to effectively find all subtasks where information will help, this approach will find none of the desired instances. In both problem sets, the trained model had better accuracy than the naive baseline (both slightly over

84%) while still finding many of the instances where added information would help (55.20% of the instances in the quality set, 19.03% in the duration set).

Table 2. Comparing the overall accuracy and number of high information instances found within a 2 category decision tree model.

	Learned Model	Naive (Majority)
Accuracy (Quality)	85.04%	81.95%
Instances Found (Quality)	55.20%	0%
Accuracy (Duration)	84.13%	83.45%
Instances Found (Duration)	19.03%	0%

Finally, we studied the three category classification task. Here, we divided the training data into a High category where information helped 10 or more units, a Low category where information helped less than 10 units, and a Zero category where information did not help. The results of this experiments from the quality set are found in Table 3, and those from the duration experiments are in Table 4.

Within these tables, we present the classification confusion matrix, often presented in multi-category classification problems. We plot the number of instances found within a given category (the diagonal of table) as well as the number of instances misclassified per category. Not all misclassification errors are necessarily equally important. For example, misclassifying a High problem or Low, or a Low problem as High is likely to be less problematic that classifying a High problem as Zero. For example, within the quality experiments, only 22 of 66 High instances were classified as High, but another 22 of these instances were classified as Low. This distinction can be quite significant. The recall of the High category alone (High classified as High) is only 0.33, however, if we view classifying High either as High or Low as being acceptable, the recall jumps to 0.67.

Table 3. Comparing the accuracy and number of high information instances found in a 3 category decision tree model with **quality** uncertainty.

	Classified as High	Classified as Low	Classified As Zero	Recall	Ave. Accuracy
High	22	22	24	0.32	82.53%
Low	15	100	170	0.35	82.53%
Zero	18	86	1465	0.93	82.53%

Table 4. Comparing the accuracy and number of high information instances found in a 3 category decision tree model with **duration** uncertainty.

	Classified as High	Classified as Low	Classified As Zero	Recall	Ave. Accuracy
High	13	14	19	0.28	82.54%
Low	9	83	277	0.23	82.54%
Zero	10	109	1974	0.94	82.54%

As the machine learning models never achieved a recall near 100%, we considered creating models which were biased towards categorizing a task as benefitting from information. While this bias will result in a higher recall of this category, it will come at a cost of false positives that will lower the overall accuracy of the model. This type of approach would likely be useful if the human user is able to be prompted Q times to add information, when Q is greater than the actual number of tasks that can benefit from added information. Alternatively, this approach will also be useful if the known cost from interrupting the user is relatively low. For example, if the cost of prompting the user is 1 unit, we should be willing to ask several queries for information so additional High instances (each worth 10 units) can be found. To train this model, we followed the previous developed MetaCost approach [3] and refer the reader to their work for additional details in how the cost bias is created.

We did find that the MetaCost approach was extremely effective in increasing the recall of the desired system categories, albeit at a cost of false positives that reduced the overall accuracy. To explore this point, we used the MetaCost function to apply different weights for falsely classifying a subtask where information was useful (Yes) as belonging to the non-useful category (No). The base weights, or unbiased classification, will have equal weighting for these categories (1 to 1 weight). We found that as we increased these weights, we obtained progressively higher recall from the desired Yes category, but at an expense in overall reduction of model accuracy. We present the results of this approach from the quality experiment in a two category classification model in Table 5. For example, a 5 to 1 bias towards the Yes category found 566 of the 607 instances (or 0.93 recall). However it had a higher rate of false positives (0.32) and lower accuracy (72.61%) from the baseline (1 to 1 weights). We also applied this approach to the two category duration problem set, as well as the quality and duration 3 category classification models. As expected, in all cases the cost bias was effective in increasing the recall of the categories where information helped, albeit at a cost of more false positives.

Table 5. Exploring the tradeoff between higher recall of desired results (Yes instances found), and false positives and negatives within a 2 category decision tree model with quality uncertainty.

Weight	Total Accuracy	Found	Not Found	Recall	False Reject	False Accept
1 to 1	85.04%	335	272	0.55	0.45	0.08
2 to 1	82.46%	435	172	0.72	0.28	0.15
5 to 1	72.61%	566	41	0.93	0.07	0.32
10 to 1	70.95%	584	23	0.96	0.04	0.35

7 Discussion and Future Directions

In general, we found that the tightness measure was extremely effective in finding which subtasks would **not** benefit from adding information. By locally filtering

out which tasks were not constrained we were able to focus on determining whether adding information would help in the remaining subtasks. However, several key directions are possible to expand upon this work.

First, we found decision trees were overall very effective in quantifying the expected impact of adding information, and thus were helpful in recommending if the user should be contacted for additional information. In contrast, previous work on adjustable autonomy [13] found decision trees were ineffective in enabling agents to make autonomous decisions. It seems that the difference of results stems from the very different tasks considered. The previous work used the learned policy from decision trees to enable agents to act independently of people within their group. As a result, their scheduler system made several critical errors (such as canceling group meetings and volunteering people against their will for group activities) by overgeneralizing decision tree rules. In contrast, our support system never tries to make autonomous decisions, and instead took the support role of recommending what constraint(s) a person should focus on. This distinction may suggest the need to create different types of learning models for different agent-human tasks. We hope to further explore this point in the future.

Also, further work is necessary to identify general attributes where information definitively **does** add utility. Our hypothesis is that local information is sufficient for guaranteeing that a given problem is not constrained, and thus information will **not** help. However, the disadvantage to the exclusively local approach we present is that agents are less able to consider the full extent of all constraints within the problem. Because of this, we believe this approach was less affective in finding the cases where information would **definitely** help.

We have begun to study several of these directions in parallel to the work we present here. Along these lines we have studied how the tightness measure can guide agents if they should communicate all of their constraints to a centralized Constraint Optimization Problem (COP) solver [8]. The COP solver would then attempt to centrally solve the constraint problem after receiving all constraints from all agents. To address what agents should communicate, each agent viewed all of its constraints as belonging to only one task window, with all subtasks falling within this window. We found that the resulting tightness measure created three classic clusters of constraint interactions: under-constrained, constrained, and over-constrained with a clear communication policy emerging based on this measure. Under-constrained problems had low tightness and could locally be solved without sending any constraints to the COP solver. Constrained problems had a medium tightness value, and most benefited from having every agent send all of its constraints. Problems with the highest tightness value were the most constrained. In fact, these problems had so many constraints that agents sending all constraints flooded the COP solver, which was not able to find the optimal solution. In these problems, agents were again best selecting communication approaches that sent fewer constraints.

Finally, it is important to note that these research directions are complementary. We foresee applications where different tightness measures are applied to filter and predict different characteristics. Say, for example, a domain exists

where agents could send constraint information freely. As we have previously found, sending too much information can prevent centralized problem solvers from finding the optimal solution [8]. One solution might be to apply the local tightness measure we present here to filter cases where information definitely will not help, and then have agents send all remaining constraints. This more limited set of constraints might be most manageable than the original set. We are hopeful that this work will lead to additional advances in this challenging field.

8 Conclusion

In this paper we presented an approach to quantifying the expected utility change from adding information to agents within distributed scheduling problems. Agents exclusively used local information about their constraints to predict whether adding information will help the group increase its utility. The significance of this work is its ability to enable agents to find which constraints will most benefit from additional human information, without using resource intensive queries required in other approaches [12]. Towards achieving this goal, we defined and used a general **tightness** measure and domain specific information from the cTAEMS distributed scheduling domain to train a regression based learning model for numerically quantifying the value of this information, as well as classifier models to identify if a given subtask should be categorized as benefiting from information or not. In general, we found that the non problem-specific tightness measures was extremely effective in finding where addition information about constraints would **not** be helpful. Domain specific cTAEMS information was moderately useful in identifying where information **would** be helpful. Finally, we presented several possible future direction of study in this challenging problem.

References

1. Sven A. Brueckner and H. Van Dyke Parunak. Resource-aware exploration of the emergent dynamics of simulated systems. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 781–788, New York, NY, USA, 2003. ACM Press.
2. Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the Really Hard Problems Are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI-91, Sidney, Australia*, pages 331–337, 1991.
3. Pedro Domingos. Metacost: a general method for making classifiers cost-sensitive. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164, New York, NY, USA, 1999. ACM Press.
4. 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.Q. Zhang. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.

5. Roger Mailler and Victor Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 438–445, Washington, DC, USA, 2004. IEEE Computer Society.
6. Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
7. Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic "phase transitions". *Nature*, 400(6740):133–137, 1999.
8. Avi Rosenfeld. A study of dynamic coordination mechanisms. *Ph.D. Dissertation, Bar Ilan University*, 2007.
9. Jiaying Shen, Raphen Becker, and Victor Lesser. Agent Interaction in Distributed MDPs and its Implications on Complexity. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 529–536, Japan, 2006. ACM.
10. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, June 2005.
11. R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 310–317, Washington, DC, USA, 2004. IEEE Computer Society.
12. D. Sarne and B. Grosz. Estimating Information Value in Collaborative Multi-Agent Planning Systems. In *AAMAS'07*, (to appear), 2007.
13. Paul Scerri, David V. Pynadath, and Milind Tambe. Towards adjustable autonomy for the real world. In *JAIR* volume 17, pages 171–228, 2002.
14. Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.