

# Quantifying the Expected Utility of Communicating Constraint Information\*

Avi Rosenfeld<sup>1,2</sup>, Sarit Kraus<sup>2</sup> and Charlie Ortiz<sup>3</sup>

<sup>1</sup>Department of Industrial Engineering  
Jerusalem College of Technology, Jerusalem, Israel 91160

<sup>2</sup>Department of Computer Science  
Bar-Ilan University, Ramat-Gan, Israel 92500

<sup>3</sup>SRI International, 333 Ravenswood Avenue  
Menlo Park, CA 94025-3493, USA

Email: {rosenfa, sarit}@cs.biu.ac.il, ortiz@ai.sri.com

November 30, 2007

## Abstract

In this paper we investigate methods for analyzing the expected value of communicating information in multi-agent planning and scheduling problems. As these 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 distributed agents estimate their problem *tightness*, or how constrained their local sub-problem is. Agents can then immediately identify many problems which are not constrained, and will not benefit from sending or receiving further information. Finally, 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 c-TAEMS scheduling domain and found that this approach was overall effective.

**Keywords:** Multiagent Scheduling, Adaptive Coordination, Localized Decisions

---

\*This research was supported by the DARPA Coordinators Program under Air Force Research Laboratory Contract FA8750-05-C-0033. Sarit Kraus is also affiliated with UMIACS.

## 1 Introduction

The ability to effectively harnessing the relative strengths of mixed human agent groups can be critical in 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 [10]. Agents can quickly compute possible group compositions and assess group behavior even in dynamic and time sensitive environments [5, 6, 15]. This ability can be invaluable in focusing a person’s attention to the most critical decisions in these environments [10].

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, 6]. 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 [10]. As has been previously observed, coordinating decisions involving incomplete information and communication costs significantly increases the problem’s complexity [12].

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 [10]. 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 [10]. Similar issues have arisen in the Information Gain measure that has been put forward by the machine learning community [7]. 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 [9].

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 [10]. We accomplish this through three stages. First, agents send a limited amount of information regarding the general structure of their constraints. Specifically, we assume constraints are ordered through a Hierarchical Task Network (HTN) usually used in scheduling and planning problems [13]. This allows agents to easily assess if higher levels of the HTN tree could potentially impact their local task. Agents then use this information to identify which tasks can be locally solved through a general, non domain-specific constraint "tightness" measure we present. A constraint tightness of less than one indicates that a task is underconstrained and will not benefit from any additional information. Agents can then 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. Finally, this paper also describes a solution where agents use offline apply machine learning techniques to identify which of the remaining tasks are likely to have the highest expected utility from information.

The next section provides the background and motivation for this work. In Section 3 we briefly describe the c-TAEMS 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. [11] 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) [5], and a variety of algorithms have been proposed for solving these problems [5, 6, 15]. In these problems, inter-agent constraints must be coordinated to find the solution that satisfies as many of these constraints as possible. However, these problems are known to be NP-complete, even if agents freely share all information at their disposal [6, 12]. 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 [10, 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, 8]. 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 [8]. Based on that body of work, we attempt to locally identify tasks which are underconstrained, or represent “easy” problem instances that can be locally solved without any additional information, as well as the “hard” instances 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 that constitute these problem instances [1, 12]. In the theoretical graph coloring problems previously studied, phase transitions were found that were associated with the ratio of constraint clauses per variable [8]. 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 c-TAEMS 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**

The CA system we propose takes as its input the system’s distributed constraints, formulated in c-TAEMS, and outputs the constraints a person should focus on. c-TAEMS is a robust, task independent language used by many researchers for studying multiagent task scheduling problems [10] based on the TAEMS language standard [4].

The c-TAEMS language is composed of methods, tasks and subtasks that define a coordination problem where groups of agents interact in a Hierarchical Task Network (HTN) structure. Methods represent the most basic action one 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) will be for each possible result associated with the execution of that method. Uncertainty can be modeled through defining a probabilistic distribution for the values of Q, D, and C. 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. 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 assumed to be 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. Hard constraints between tasks or methods can be modeled in terms of what are referred to as non-local effect (NLE) constraints between tasks, using the primitives *enable* or *disable*. For example, assuming one task must occur before another, one could represent this constraint in terms of an enables relation between those two tasks. c-TAEMS can model that two tasks (or subtasks) cannot both be performed through a disables relation between the two tasks. Finally, soft constraints are modeled through *facilitates* or *hinders* relationships. For example, if the group prefers, but does not require, to execute one task before another, it would define a *facilitates* relationship whereby extra quality is awarded for scheduling this preference.

Figure 1 presents an example of a scheduling problem instance, described in c-TAEMS. In this example, 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

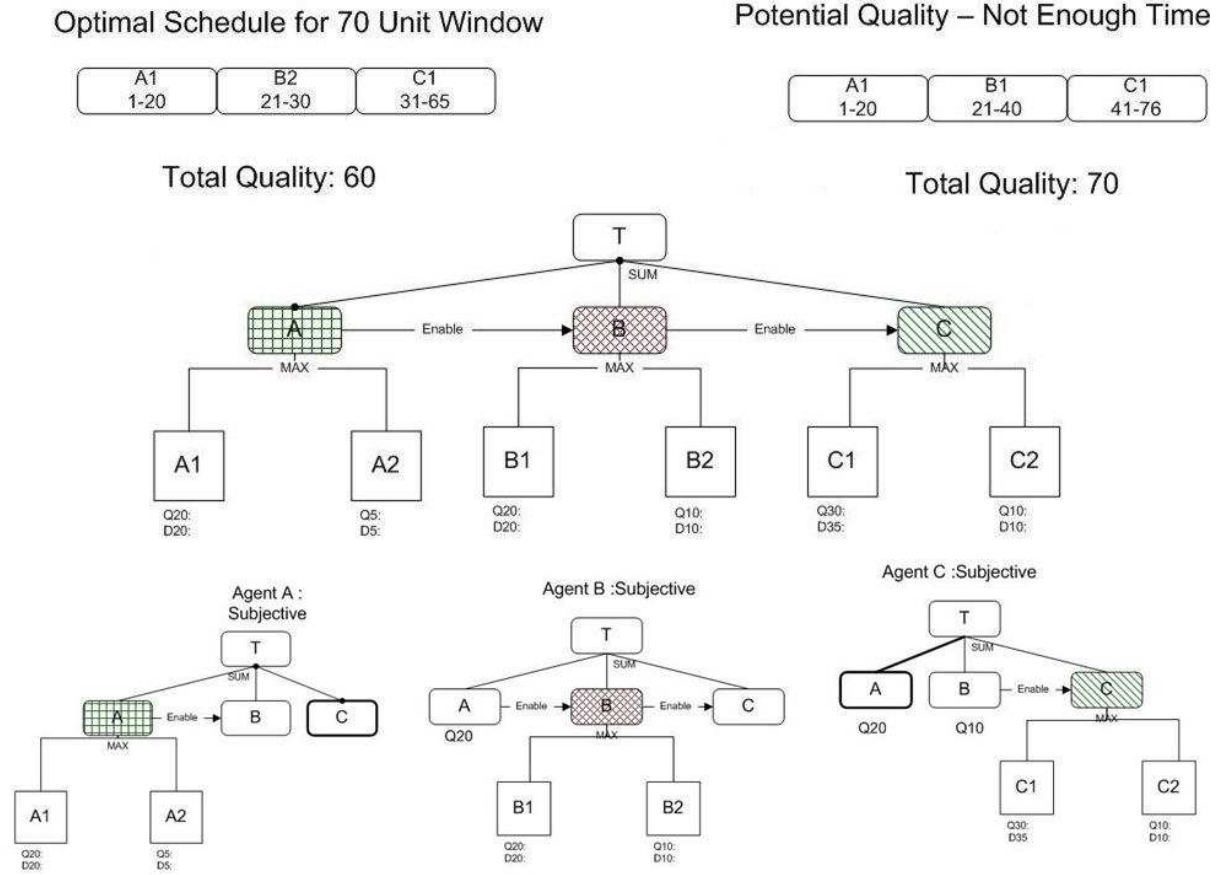


Figure 1: A sample c-TAEMS Scheduling Problem (global view, middle) with 3 agents (3 subjective views on bottom).

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 that methods A1 and C1 can be scheduled.

This paper focuses on the challenging problem of identifying constraints that can be safely resolved by local agents, and which constraints can most benefit from additional input. We particularly focus on the impact uncertainty in task quality and duration will have on decisions a specific agent may take. 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.

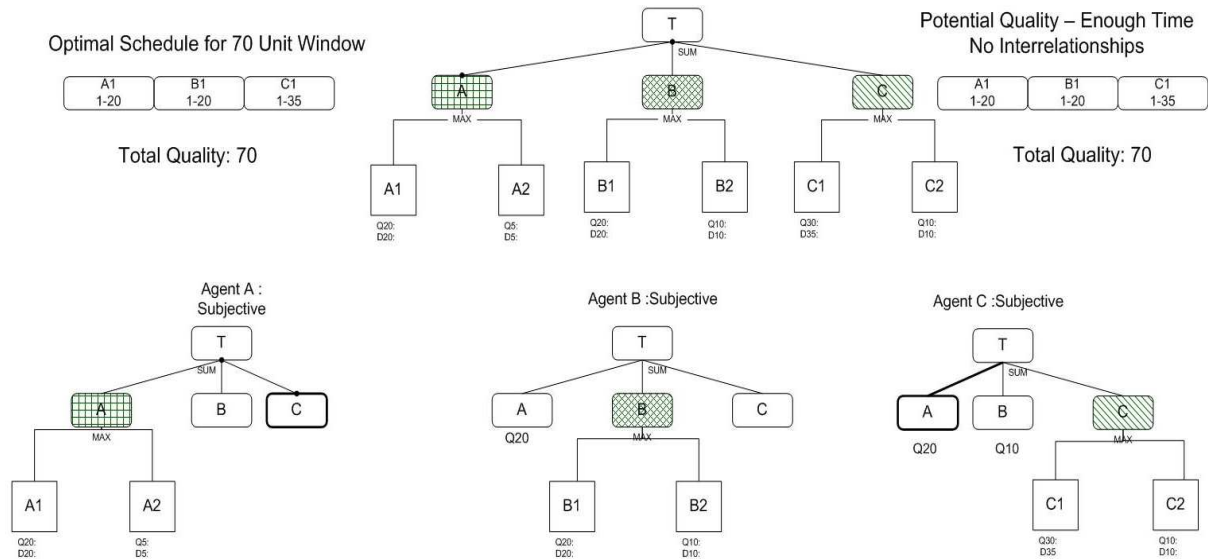


Figure 2: Modified c-TAEMS Scheduling Problem with NLE relationships removed. Note agents are now free to schedule tasks independently.

The impact from adding information in cases with NLE relationships can often be easily quantified. As these constraints must be explicitly formulated, their potential impact can often be easily detected. Conversely, the removal of these constraints may signify that agents can safely make local decisions without impact on other agents. For example, Figure 2 contains the same basic c-TAEMS structure from Figure 1, but with the NLE enable statements removed. Note that all agents can independently schedule their methods independently, and every agent can greedily choose the subtask with the highest quality (A1, B1, and C1) at the beginning of the time window.

Adding information can be equally significant in cases where task relationships are constrained based on their QAF relationships, especially in cases where uncertainty exists. For example, Figure 3 depicts a scheduling problem from one agent's local view. In this example, we assume no explicit constraints (NLE's) exist, and this agent can choose the best method(s) to schedule from among options A, B, and C within a 40 time until window. However, assuming uncertainty exists in the duration of these methods, this one agent may not have enough time to schedule all of these tasks. For example, assuming method C lasts for 30 time units (option C1), only 10 additional time units to potentially execute methods A and B. Furthermore, if either of these methods requires 20 time units (options

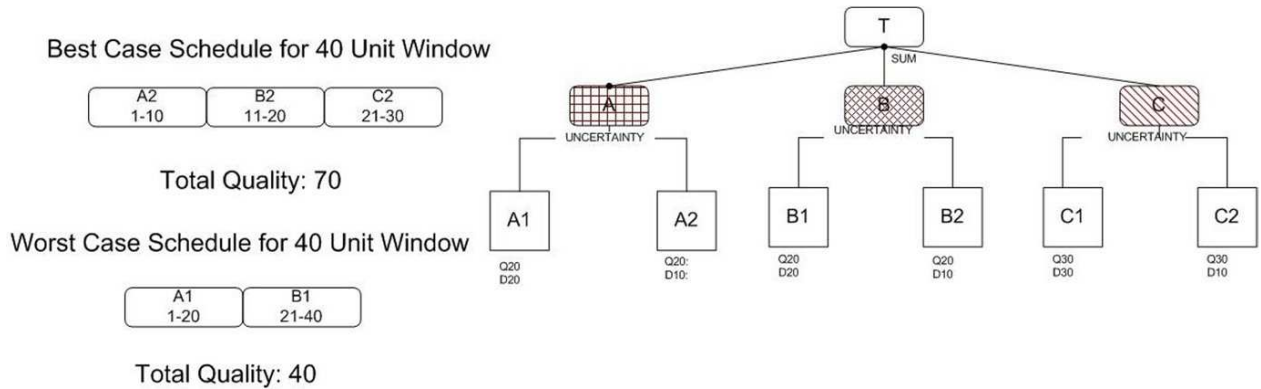


Figure 3: The impact of QAF constraints in the HTN c-TAEMS structure (uncertainty within only one agent).

A1 and B1), they cannot be scheduled. In cases of to this type of uncertainty, we assume agents schedule for the worst case, and the agent must only schedule methods A1 and B1. However, a human operator could provide information that tasks will take the shorter time, and potentially all three methods could be scheduled. For example, a person may have knowledge that tasks B and C will take the shorter times (options B2 and C2), and allowing the agent to achieve a quality of 70 from executing all three methods. Thus, in this and similar problems the CA system must identify the importance of querying the human user for additional information.

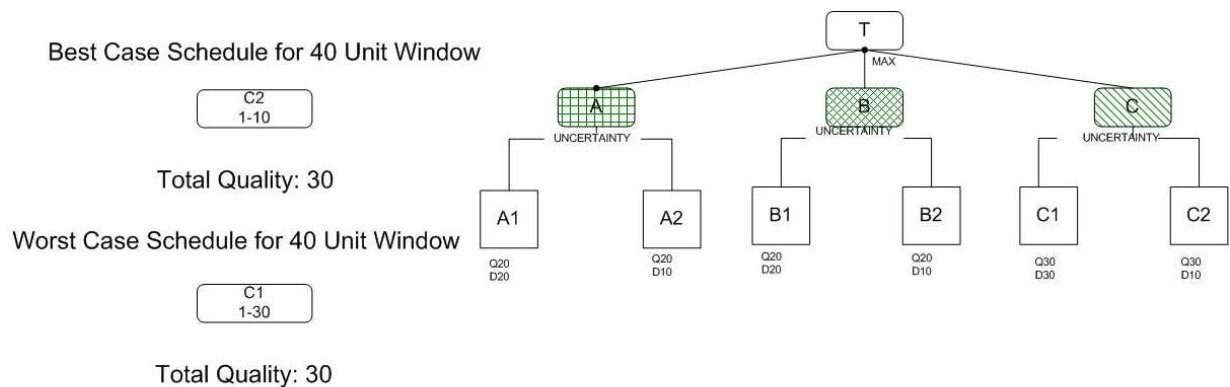


Figure 4: A modified c-TAEMS structure from Figure 3 where QAF constraints are relaxed.

Again, modifying the c-Taems structure, even slightly, can change the importance of these types of task uncertainty. Figure 4 contains the same c-Taems problem, with a max QAF at the problem root instead of a sum. Here, the agent

cannot execute all three methods A, B, and C, and is free to greedily choose between the options. In this example, even assuming task C will take its maximal duration (30 units) sufficient time exists to schedule this method within the task’s 40 unit time window. As a result, in this case the CA system should identify that there is no potential benefit from additional information, allowing the human operator to focus on other tasks.

## 4 Locally Quantifying Scheduling Constraints

In general, novel mechanisms are required to generally find which types of constraints are most worthy of further information. While we used the c-TAEMS 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, 8]. However, novel measures of interaction difficulty are needed to help quantify interactions so phase shifts can be discovered.

In this section we present a general “tightness” measure to quantify how much overlap exists between constraints. In referring to these constraints, let  $G = \{A_1, A_2 \dots, A_N\}$  be a group of  $N$  agents trying to maximize their group’s collective scheduling utility. Each agent has a set of  $m$  tasks,  $T = \{T_1, \dots, T_m\}$  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  $T_i$ ,  $\{Q_{i1}, Q_{i2}, \dots, Q_{ij}\}$  as the possible quality outcomes for  $T_i$ , and  $\{R_{i1}, R_{i2}, \dots, R_{ij}\}$  as the possible duration lengths of  $T_i$ .

Based on these definitions and assumptions, we model an agent’s quality *tightness* as:

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

where  $\text{Quality}_{\max}(T_i)$  returns the maximal quality from the possible values of  $T_i$ ,  $\{Q_{i1}, Q_{i2}, \dots, Q_{ij}\}$ , and  $\text{Quality}(\text{Window}(T_i))$  returns the minimal 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 must assume the worse case, and the lowest quality value is used 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. A1 has a quality distribution between 10 and 20, and A2 has 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,  $Tightness\text{-}Quality(A1) = 20/15 = 1.33$  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:

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

where  $Duration_{max}(T_i)$  returns the maximal duration from  $\{R_{i1}, R_{i2}, \dots, R_{ij}\}$ , and  $Duration(Window(T_i))$  refers to the time allotted for completing task  $T_i$ . Note that within the c-TAEMS 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  $40/25$  or 1.6. Without any additional information, we must assume that this sub-task 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 the important property that it 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.

Note that the previous tightness definitions assume that a given task’s window,  $Window(T_i)$ , can be quantified. Assuming no other constraints exist for a given task, this value can indeed be locally measured without any additional information. However, if these constraints do exist, the task’s quality or duration window must be extended to include the potential impact for these others tasks. For example, assuming a given subtask has a non-local event (NLE) and thus explicitly links several tasks, the concept of the task “window”,  $Window(T_i)$ , must be expanded to include all affected tasks. Similarly, tasks’ accumulation functions (QAFs) can also connect several tasks. Specifically, assuming a c-TAEMS sum QAF exists between several tasks, the task’s  $Window(T_i)$  must be expanded to include all connected tasks. As a result, agents can only safely apply the tightness measure we describe when their local tasks’ cannot be impacted by these constraints, allowing them to autonomously measure their window,  $Window(T_i)$ .

---

**Algorithm 1** Checking for Explicit and HTN Constraints (C-TAEMS File  $T$ , Task  $T_i$ )

---

```

1: constraint-bit  $\leftarrow$  unset
2: if Explicit( $T_i$ ) then
3:   constraint-bit  $\leftarrow$  set
4: else
5:   for  $j = Level_1$  to  $Level_{T_i}$  do
6:     if HTN-Constrained( $T_i$ ) then
7:       constraint-bit  $\leftarrow$  set
8:     end if
9:   end for
10: end if

```

---

We assume that agents set and check a constraint “bit” before measuring task  $T_i$ ’s tightness measure. We outline the procedure of setting this bit in algorithm 1. By definition, NLE’s or other explicit constraints can be easily observed (being they are explicitly defined), resulting in the agent immediately setting this bit (line 3). Otherwise, the agent must check for constraints higher up in the HTN constraint tree that may impact this task. This is done through having the agent iteratively check for any higher level QAF constraints starting from the root node ( $Level_1$ ) of the HTN structure through its current level ( $Level_{T_i}$ ). Assuming any such constraints are found (e.g. a c-TAEMS sum QAF), the agent again sets this constraint bit (line 7). Otherwise, the bit remains set to *unset*. Note that instead of simply setting this constraint bit in either of these cases, an agent could potentially learn more about its task window  $Window(T_i)$  through communicating

with other agents about these constraints. Potentially, this could allow the agent to definitively calculate the task window length even in these cases. However, in order to minimize communication, agents do not further explore the issue, and instead assume the worse-case, or this task window's tightness is more than one.

When an agent estimates the value of gaining additional information, it checks this constraint bit. In cases where the constraint bit is not set, and a tightness value of less than one is measured, agents can assume their task is underconstrained and cannot benefit from additional information. In cases where the constraint bit is set, or the tightness measure is greater than one, agents cannot definitively identify if additional information will be beneficial exclusively based on local information. In these cases, agents use machine learning approaches based on local problem attributes to predict the value of information. We present specifics of this approach in the next section.

## 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 benefitting 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 2. We used a problem generator created by Global Infotech Inc. (GITI) for the purpose of generating c-TAEMS problems within the framework of the COORDINATORS DARPA program<sup>1</sup>. We created two sets of 50 problems (the value for X in the algorithm) where c-TAEMS 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 many possible permutations – these 100 total c-TAEMS problems contained over 5000 subtasks where constraint information could potentially be communicated by the user.

The goal for creating these test cases was to study how the user’s information about quality and duration uncertainty affected the group’s utility. We used a previously tested c-TAEMS centralized scheduler [9] 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 possible quality

---

<sup>1</sup><http://www.darpa.mil/ipto/programs/coordinators/>

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 2 Training the Information Model(Problem Set of Size X)**

---

```

1: for  $k = 1$  to  $X$  do
2:   Without  $\leftarrow$  Utility( $Problem_k$ )
3:   for  $j = 1$  to Num(Subtasks( $Problem_k$ )) do
4:     With  $\leftarrow$  Utility( $Problem_{k,j}$ )
5:     Table[k][j]  $\leftarrow$  (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, in the quality problem set, in both the quality and duration problem sets we studied, less than 30% of the total entries within the training data benefited from any additional information. For example, in the duration problem set, there only 722 of the 2808 subtask benefited from any additional information. Similarly, in the quality problem set 711 of 2399 benefited from this 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 for so many queries.

## 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. For example, of the 1360 quality tasks with a tightness less than one, only 7 (about 0.5%) benefited from additional information.

Next, we used the Weka machine learning package [14] 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 learning algorithms 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.

We considered four different attribute sets for the learning task ranging from one that communicates all local information to one communicating no information. First, the *all information* model was based on an attribute set including all information agents have about their problem. This included the local tightness measure we presented, in addition to c-TAEMS information about tasks and methods, their potential durations and quality values, and general problem information such as the number of agents and the total number of tasks and methods to schedule. Next, we considered a model only involving the tightness measure and a binary value to indicate if there were any explicit or higher level HTN constraints for that task. As mentioned at the end of section 4, this binary constraint bit was true if any explicit (NLE) constraints existed, or if higher level sum HTN constraints, such as a sum QAF, existed that could impact that task. Note that this model required sending only a very limited amount of local information. Third, we considered the same model without this binary constraint information. Our hypothesis is that the tightness measure we present will be accurate only in cases where this binary value was false. As a result, we expected the model’s accuracy to drop once this binary value was omitted. Finally, we considered the baseline case where the majority case was always assumed, and information would help.

First, we trained and evaluated the regression based model. The results from this experiment are found in Table 1. The first column presents the results from the problem set with duration uncertainty, and the second column presents the results from the corresponding problem set with quality uncertainty. Note that this model yielded an average correlation of 0.60 and 0.73 when all local information was transferred. This value dropped to 0.55 and 0.59 with only the tightness and binary constraint information. As expected, the model’s accuracy dropped significantly, most noticeably in the duration category, once this binary information was removed. In all cases, the learned model produced significantly improved prediction accuracy. The first three attribute sets in both categories had accuracies well above the naive correlation of nearly zero (with optimal positive correlation being 1.0 or optimal negative correlation being -1.0) found within the no communication category. Thus, while these results do leave some room for improvement, they do show the success of the tightness measure even without

Table 1: Comparing the accuracy in regression trained model for duration and quality categories.

	Duration	Quality
All information	0.60	0.73
Tightness with Constraints	0.55	0.59
Tightness without Constraints	0.22	0.50
No Communication	-0.06	-0.04

other local information.

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 results here reflect the percentage of correctly classified instances, instead of the regression correlation found in Table 1. The larger class (information did not help) represents over 70% 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 model using all local information had the highest recall of the cases where information would help (0.61 in the duration set, and 0.70 in the quality set). Note that the tightness measure alone was sufficient to achieve similar results (and even had a slightly higher overall accuracy in the duration problem set) so long as minimal constraint information was also sent. As previously described in section 4, agents needed to identify if their localized task window has any other constraints that may need to be considered. Having even a binary bit (do these constraints exist or not) allows agents to definitively use the tightness measure in cases where these constraints do not exist. However, agents that lacked this information were not able to guarantee that their local tightness window was in fact underconstrained. Thus, the accuracy of the model, and the recall of the desired information dropped precipitously. Note that the model accuracy without this bit dropped from 79.95% to 70.36% in the duration problem set, and from 83.83% to 74.15% in the quality problem set. Similarly the system’s recall dropped from 0.54 to 0 (no cases found!) in the duration set, and from 0.56 to 0.34 in the quality set.

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.

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

	Duration Accuracy	Duration Recall	Quality Accuracy	Quality Recall
All information	79.16%	0.61	85.29%	0.70
Tightness with Bit	79.95%	0.54	83.83%	0.56
Tightness w/o Bit	70.36%	0	74.15%	0.34
No Communication	70.36%	0	74.29%	0

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

All information	Classified High	Classified Low	Classified Zero	Accuracy	Recall
High	135	34	50	81.37%	0.62
Low	34	250	219	81.37%	0.5
Zero	45	141	1900	81.37%	0.91
Tightness with Bit					
High	129	13	77	79.77%	0.59
Low	104	157	242	79.77%	0.31
Zero	66	66	1954	79.77%	0.94
Tightness w/o Bit					
No Communication					
High	0	0	219	74.29%	0
Low	0	0	503	74.29%	0
Zero	0	0	2086	74.29%	1

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 as Low, or a Low problem as High is likely to be less problematic than classifying a High problem as Zero. Note that within the quality experiments involving the tightness information with constraint bit (the middle portion of table 3), only 157 of 503 Low instances were classified as Low, but another 104 of these instances were classified as High. This distinction can be quite significant. The recall of the this category alone (Low classified as Low) is only 0.31, however, if we view classifying High either as High or Low as being acceptable, the recall jumps to 0.52. Similarly, in the duration experiments, the recall of the same category (Low classified as Low) was zero (none properly classified as Low). However, 113 of the cases were classified as High.

As the machine learning models never achieved a recall near 100%, we con-

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

All information	Classified High	Classified Low	Classified Zero	Accuracy	Recall
High	175	49	118	73.11%	0.512
Low	80	85	204	73.11%	0.23
Zero	92	102	1494	73.11%	0.885
Tightness with Bit					
High	222	4	116	74.57%	0.649
Low	113	0	256	74.57%	0
Zero	120	1	1567	74.57%	0.928
Tightness w/o Bit No Communication					
High	0	0	342	70.36%	0
Low	0	0	369	70.36%	0
Zero	0	0	1688	70.36%	1

sidered creating models which were biased towards categorizing a task as benefiting 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 may 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 system’s ability to find the tasks worthy of prompting the user for information, 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, and thus a lower precision value. Table 5 presents the results of this approach from the quality problem set with a two category classification model with agents shar-

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	Precision
1 to 1	85.29%	502	220	0.70	0.72
2 to 1	84.54%	538	184	0.75	0.68
5 to 1	82.66%	609	113	0.84	0.62
10 to 1	79.84%	655	67	0.91	0.57
50 to 1	70.26%	717	5	0.99	0.46

ing all local information (compare these results to the right side of the top line of Table 2) . For example, a 5 to 1 bias towards the Yes category found 609 of the 722 instances (or 0.84 recall), and a 50 to 1 bias found nearly all instances (0.99 recall). However the 50 to 1 bias yielded the lowest overall accuracy due to false positives (0.46 precision compared to 0.62 precision with a 5 to 1 bias and 0.72 with no bias). 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 lower model accuracy. Thus, a system designer will likely choose some cost bias, based on weighing the utility lost from a lower accuracy of the model (from the false positives), and the added utility gained from the user’s information (given the higher recall in the system).

## 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 [11] 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 peo-

ple 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 take 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 [9]. 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 [9]. 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 [10]. Towards achieving this goal, we defined and used a general *tightness* measure. We found that the non problem-specific tightness measure was extremely effective in finding where additional information about constraints would *not* be helpful in the c-TAEMS distributed scheduling domain. We also present an approach where locally observable c-TAEMS information can be used 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. Domain specific c-TAEMS information was moderately useful in identifying where information *would* be helpful. Finally, we presented several possible future directions 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] Rajiv T. Maheswaran, Milind Tambe, Emma Bowring, Jonathan P. Pearce, and Pradeep 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.
- [6] 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.
- [7] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [8] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic “phase transitions”. *Nature*, 400(6740):133–137, 1999.
- [9] Avi Rosenfeld. *A study of dynamic coordination mechanisms*. PhD thesis, Bar Ilan University, 2007.
- [10] D. Sarne and B. Grosz. Sharing experiences to learn user characteristics in dynamic environments with sparse data. In *AAMAS'07*, pages 202–209, 2007.

- [11] Paul Scerri, David V. Pynadath, and Milind Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research (JAIR)*, 17:171–228, 2002.
- [12] 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.
- [13] D. Smith, J. Frank, and A. Jonsson. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15(1):61–94, 2000.
- [14] 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.
- [15] 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.