A Quantitative Model of Capabilities in Multi-Agent Systems

Linli He and Thomas R. Ioerger
Department of Computer Science
Texas A&M University – College Station
{linli, ioerger@cs.tamu.edu}

Abstract

Reasoning about capabilities in multi-agent systems is crucial for many applications. There are two aspects of reasoning about the capabilities of an agent to achieve its goals. One is the symbolic, logical reasoning, which is based on whether the agent can determine a plan that can be used to achieve a goal (i.e. "know-how"). Another is quantitative reasoning about levels of skill and whether a set of tasks can be achieved within a specified time and with quality constraints. Capabilities in this sense are determined by the limits of internal processing capacity. Both artificial agents and human agents can be subject to processing capacity limits, whether due to CPU speed, or cognitive limits on attention, memory, etc. This could depend not only on the unique skill levels of each individual and their cognitive abilities, but also on their ability to handle those task demands in parallel, given intrinsic limits on internal cognitive resource capacities. In past work, researchers have focused primarily on the logical aspect of capability reasoning; less work has been done on modeling the quantitative aspects of reasoning about capabilities in agents. In this paper, we introduce a general mathematical model to define the capabilities of agents to achieve a set of tasks. Our definitions of capabilities are based on whether a feasible schedule exists to complete the tasks within the constraints, either in static environments or dynamic environments, for which we present two corresponding preliminary scheduling algorithms. We illustrate this model with two experiments to evaluate the algorithms. We conclude by discussing the potential applications of this model, and future work.

1. Introduction

It is becoming increasingly important in[1] multi-agent systems (MAS) for agents to be able to reason about each other's capabilities[1]. In large, complex MAS applications, agents need to interact, cooperate, negotiate, and solicit/accept/re-distribute tasks dynamically. This might require individuals to decide whether they can/should accept new tasks, or to whom it would be most effective or least disruptive to delegate them. For instance, in team-based multi-agent systems, agents work together toward common, shared goals [12]. Each agent might have unique skills and capabilities, and the team would like to balance its workload to be most effective [9].

To date, most previous research on modeling capabilities has focused on two aspects. First, there is the derivation of capability for complex combinations of actions, based on capability to perform the atomic steps involved. For example, one might start with associations of certain operators with certain agents, and then compute capability over sequences and more complex expressions by composition, using dynamic logic or other techniques related to programming languages semantics [4]. Typically, capability does not mean the agent has to be able to execute the action right away, but only that it can eventually force the situation to hold, regardless of the initial state or occurrence of non-deterministic events. A second aspects that has received attention within the AI community is the notion of ``know-how." [11] It is argued that it is not sufficient for an agent to be capable in principle of performing something, but that it must have the knowledge, i.e. a plan.

However, there are important cases in which these models of capability are inadequate. In some applications, there is an intrinsic limit on processing capacity. For example, a web server can only serve so many web pages per second, or a database so many queries. Workload limits stem from the fact that processing the task requires some finite amount of processing on fixed resource (such as a CPU). The existence of such limits induces a resource capacity, for which tasks must compete. This makes the notion of capability context-dependent: even if the agent could perform the task in principle, it might not be able to accomplish it in time, given its other current activities (or commitments). Meanwhile, agents can often process multiple tasks in parallel, such as by time-sharing or multi-tasking (provided the capacity is not exceeded). Interestingly, by scheduling task processing at different frequencies, the duration can be dynamically adjusted (expanded or contracted), with a compensatory change in required workload. This enables agents to adjust their processing of individual tasks dynamically in very flexible ways, e.g. to reduce workload on non-urgent tasks, or increase it on those that need to be finished for a quickly approaching deadline. So not only is capability context-dependent, but agents have the ability to manipulate the way they do things to accommodate new task demands.

Of course, there is also a sense of capability that has to do with quality. Quality is often inter-related with speed. For certain computations, e.g. search algorithms (Monte Carlo, simulated annealing, hill-climbing/gradient-descent, GAs), iterative algorithms (e.g. convergent algorithms for computing derivatives or solving large systems of equations, EM), or randomized algorithms, the more processing time they are given, the better. Hence there is a desire to run on faster CPUs, or at least ones that can compute an acceptable answer in the allotted time. Different approximation schemes can often be used that have different tradeoffs of CPU time required (i.e. "effort") versus accuracy of the result. An extreme case of this is anytime algorithms, which guarantee to give incrementally better results with increasing time [2].

[1] This work was supported in part by MURI grant #F49620-00-1-0326 from DoD and AFOSR.

Hence a more quantitative model of capabilities is needed to capture these tradeoffs among effort, speed, and quality, and to integrate all these together in the presence of intrinsic limits on workload or processing capacity, which ultimately makes reasoning about capabilities context-dependent.

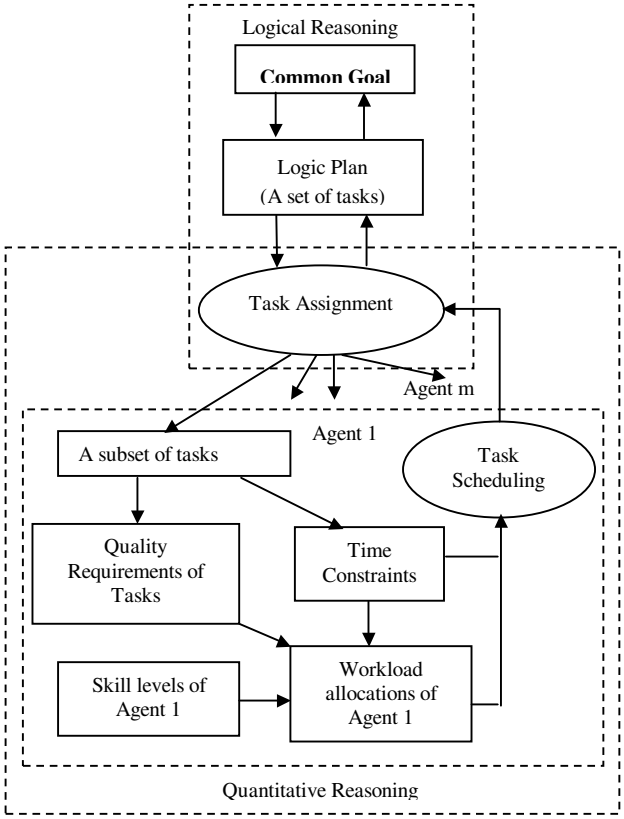

Figure 1: Capability-reasoning framework of an agent team

In this paper, we present a new definition of capability that agents can use for reasoning about their performance and that of others in the environment. Our model is situated within a quantitative framework that allows for capturing and expressing graded tradeoffs among time (duration), effort (workload), and quality with which tasks are performed. In this framework, whether or not an agent is capable of performing a task depends on what other tasks and deadlines it currently is trying to honor. We formalize this context-dependence by defining capability based on whether a feasible schedule can be found in which all tasks (current and new) can be completed successfully by their deadlines (and other constraints) without exceeding 100% of the agent's individual workload limit (or other resource capacities) at any given time. However, it is important to note that the definition is flexible enough to allow for the possibility that the agent might be able to re-arrange its schedule of processing current tasks to accommodate the new one, such as by expanding the duration of processing non-critical tasks to free up some processing resources. Hence this definition of capability is not a simple ``property" (e.g. a rule expressed in logic), but is tied to the notion of scheduling (i.e. determining whether a feasible schedule exists). We discuss application to teamwork modeling and agent interactions toward the end of this paper.

Figure 1 depicts a capability-reasoning framework for a team-based agent system. Through the logic reasoning capabilities, the team can build a plan composed of a set of tasks. Through a task assignment algorithm that may be part of the plan, each team agent gets a subset of tasks. By the quantitative reasoning, each agent can know whether executing the new tasks satisfies its workload limitation and the time constraints of each task. If any team member determines it is not capable of its tasks, it may cause either reassignment of tasks or revision of the entire team plan.

2. Capability

In this section, we will first introduce a mathematical model of single-agent capabilities. Then we will extend this model to define the capabilities of a group of agents.

2.1. Mathematical Model

"Workload" has been studied a great deal in cognitive science and industrial engineering. A succinct definition of mental workload by O'Donnell [7] is "The term workload refers to that portion of the operator's limited capacity actually required to perform a particular task." The theoretical assumption underlying this definition is that the human operator has limited processing capacity or resources. If the processing demand of a task or tasks exceeds available capacity, performance decrements result [13]. Furthermore, numerous studies show a substantial positive relationship between cognitive abilities and job performance [6]. In multi-agent systems, especially involved with human agents, there exist similar situations as well. The processing capacity of each agent is limited.

Generally, people assume workload a constant, namely, a task executed in its duration always has a constant workload demanding. But in the real world, the workload of a task may be changing with time. For instance, baking turkey is a high workload task at the beginning. After putting the turkey into the oven, it turns to a low workload task. Nobody will stay in front of the oven to watch the turkey in the entire process. The chef only needs to check it several times. Therefore, in our model, we delineate the workload allocated to a task is a function of time. We represent this function by equation 2-1.

$$w = \begin{cases} f(t) & \text{if } T_{start} \le t \le T_{end}; \\ 0 & \text{otherwise.} \end{cases} \qquad (2\text{-}1)$$

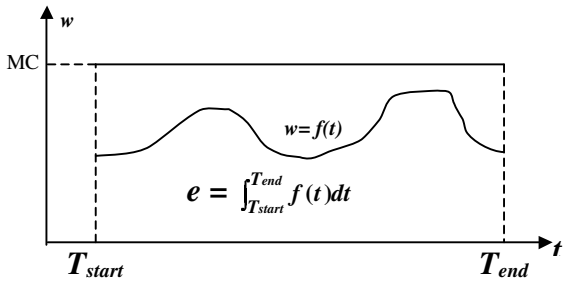

Figure 2: Workload and effort in duration $\{T_{start}, T_{end}\}$

Where, T_{start} and T_{end} denote the start and end times of the task, respectively, and w refers to workload allocation; the duration of the task is represented by $d = T_{end} - T_{start}$. We use MC to denote the maximum processing capacity that the agent has available at any time. In practice, it could be attention of a person or computational capacity of a processor.

In the sense of "Work harder, finish a task earlier", there exists a trade-off between the workload allocated and the time required for a task with given quality. In our model, we assume that the more workload, the less time required. We use the term "effort" which represents the relation between the workload allocation and the duration for a given task with a specific quality. According to the above function of workload and duration, we define the effort requirement of a task by equation 2-2, where e refers to effort. Figure2 depicts the workload allocation and the effort requirement function for a task in duration $\{T_{start}, T_{end}\}$.

$$e = \int_{T_{start}}^{T_{end}} f(t)dt \qquad (2\text{-}2)$$

We use the term "quality" to denote the performance requirement of a task. The quality of a task an agent can achieve depends on its workload allocation and the time required. We can translate this relation to the one between quality achievement and effort expended. In general, the more effort the agent exerts, the higher the quality it can achieve. Generally, it is a monotonic relation [15]. Equation 2-3 shows the quality achievement as a function of effort, where q refers to the corresponding quality by effort paid-off e. Figure 3 depicts an example of the functional relation between quality and effort.

$$q = g(e) \qquad (2\text{-}3)$$

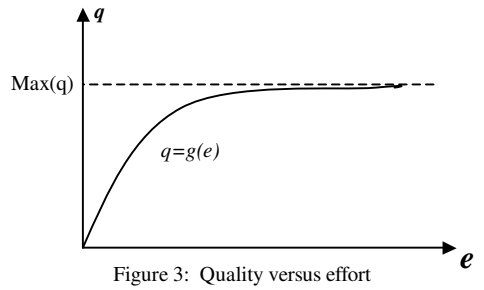

Figure 3: Quality versus effort

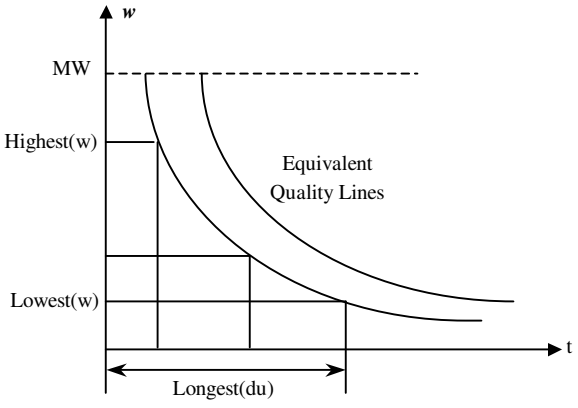

Figure 4: Equivalent Quality (or Effort) Line

The effort is a constant for a given quality requirement, no matter how the workload is allocated and the duration is assigned. In other words, we could adjust the workload and duration combination to satisfy the effort requirement for a given quality. In principle, we could adjust the workload and duration arbitrarily. But there exists a minimum workload limit for most tasks in the real world. On the other hand, after a threshold, even if you allocate more workload, you might not be able to improve the quality achieved. We denote the threshold workload allocation as the highest workload for a given task. Thus, we get the adjustable ranges of workload and duration needed for a given task with a specific quality requirement. Figure 4 shows the adjustable ranges of workload and duration of a task with a specific quality requirement. For the same quality requirements, the efforts needed are equivalent (iso-curves). The rectangles in Figure 4 have the same area and hence effort. We should clarify that there are some tasks for which the workload and duration might not be adjustable. In our model, we can view the workload range is not an interval but a constant.

All the above mathematical relations relate to a single agent. Considering multiple agents, we could not ignore the different skill levels from different agents for a same task. Intuitively, letting an agent with higher skill level to do a task is more efficient than assigning the task to an agent with lower skill level. In our mathematical model, that means that different efforts are needed for different agents to achieve a task with a given quality requirement. We view the skill as a multiplier that increases the effect of a given amount of effort proportionally. The smaller the value of skill, the larger the effort needed for the same task. An agent could be not capable of some tasks at all. In this case, its skill value can be zero. Meaning, no matter how much effort the agent pay for a task, it could not achieve the given quality of the task. Currently, we treat the skill value as a coefficient. In practice, it could be more complex than that. For example, a person could improve his skill level for a specific task after he has done it many times. Hence, skill value could be a function of session, time, or some other parameters.

2.2. Definition of Capability

We proceed by introducing formal definitions of an agent's capabilities based on the above mathematical model in this section. Our basic idea is that an agent can be said to be capable of doing a task if it can find a schedule that can accommodate it with other on-going tasks and still satisfy all constraints. The application of scheduling to reasoning about workload in multi-task environments has been considered in various cognitive models [14]. In a static environment, the agent schedules all given tasks based on its capability before it starts to execute them. In a dynamic environment, the agent may execute some tasks as well as evaluate whether it can accept new arrival tasks.

Given: a set of task $X = \{x_1, \ldots, x_n\}$, which has the corresponding quality requirement set $Q = \{q_1, \ldots, q_n\}$. For each tasks x_i in X, the release time and the deadline is the corresponding interval $\{r_i, d_i\}$, $i = 1, \ldots, n$. The definition of the agent capability in a static environment is as follows:

Definition 1: Agent Capability in Static Environment

Agent A has the *capability* to achieve the tasks in X, or *CapStatic(A, X),* iff ∃a schedule S of X, which is { $\{t_{s1}, t_{e1}\}$, …, $\{t_{sn}, t_{en}\}$ }. For each task x_i, the workload w_i and the effort e_i are calculated by the equations as follows:

$$w_i = \begin{cases} f_i(t) & \text{if } t_{si} \le t \le t_{ei} \\ 0 & \text{otherwise} \end{cases}$$

and effort is a function of quality required with a skill multiplier:

$$e_i = g(q_i)/s_i$$

also, e_i should satisfy the following equation:

$$e_i = \int_{t_{si}}^{t_{ei}} f_i(t)\,dt$$

while at any time point t' during the time interval $\{earliest(t_{si}), latest(t_{ei})\}$,

$$\sum w_i = \sum f_i(t') \le MW$$

and for each task x_i

$$t_{si} \ge r_i \quad \& \quad t_{ei} \le d_i$$

where t_{si} and t_{ei} denote the start time and the end time of task x_i respectively, s_i is the skill level of agent A to achieve task x_i.

In a dynamic environment, assume an agent A is running with schedule $S = \{\{t_{s1}, t_{e1}\}, …, \{t_{sn}, t_{en}\}\}$of a task set $X = \{x_1, …, x_n\}$ at time t_0 when a new task set $Y = \{y_1, …, y_m\}$ arrives. There is a subset $Z = \{z_1, …, z_k\}$of the task set X, which includes the tasks which are being executed but are not finished and are interruptible or the tasks which are waiting to be executed at time t_0. The workload allocation distribution of tasks in X-Z is $dw = w(t)$. The corresponding quality requirement set of Y is $Q = \{q_1, …, q_m\}$. For each task y_i, the release time and deadline is given by the interval $\{r_i, d_i\}$ $(i = 1, …, m)$. The definition of the agent capability in this dynamic environment is as follows:

Definition 2: Agent Capability in Dynamic Environment

Agent A has the *capability* to achieve the tasks in Y without violating the tasks of X-Z in execution, or *CapDyn(A, X, Y), iff* ∃ a new schedule S'= $\{\{tz_{s1}, tz_{e1}\}, …, \{tz_{sk}, tz_{ek}\}, \{t_{s,k+1}, t_{e,k+1}\}, …, \{t_{s,k+m}, t_{e,k+m}\}\}$ of the task set $Z \cup Y$. For each task in this union, the workload w_i and the effort e_i are calculated by the following equations:

$$w_i = \begin{cases} f_i(t) & \text{if } t_{si} \le t \le t_{ei},\ \{t_{s1}, t_{e1}\} \in S' \\ 0 & \text{otherwise} \end{cases}$$

$$e_i = g(q_i)/s_i$$

also, e_i should satisfy the following equation:

$$e_i = \int_{t_{si}}^{t_{ei}} f_i(t)\,dt$$

while at any time point t' during the time interval $\{earliest(t_{si}), latest(t_{ei})\}$,

$$w_i = f_i(t') \le MW - dw(t')$$

and for each task y_i

$$t_{si} \ge r_i \quad \& \quad t_{ei} \le d_i$$

where t_{si} and t_{ei} denote the start time and the end time of each task in $Z \cup Y$ respectively, s_i is the skill level of agent A to achieve the task.

We give an example in Figure 5 to show how an agent manages its workload allocation to satisfy its processing capacity and the deadlines of its tasks and how the skill levels of different agents affect reasoning about agent capability. Assume a professional secretary agent could input a file into a computer in 1 hour with a given accuracy if he pays all his attention (maximum processing capacity) to this inputting task. But a novice secretary needs 2 hours. The professional can input some files into a computer and listen to the radio at same time. In this case, he can finish this file in 1 and half hour. The novice could not handle these two tasks parallel at all. So the boss asks the professional secretary to listen to the financial news simultaneously with inputting the file. Figure 5 depicts how the secretary to handle these two tasks simultaneously by reducing the workload of the inputting task, which is represented by x_0. Suppose the deadline of x_0 is time T_0+2, and the financial news is from time T_0 to T_0+1. In this case, the secretary can finish both tasks. However the novice can only either makes the task x_0 meet its deadline or gets the financial news. Here, the task of listening to the financial news could not be adjusted because the time constraints of listening to news are not adjustable. This example shows the benefit of adjusting the combination of workload and duration of a task. Also it shows different skill levels will affect the performance of a task.

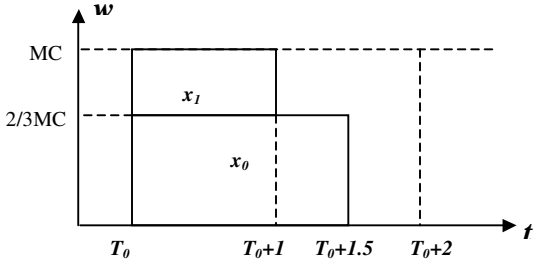

Figure 5: Secretary Example

It is straightforward to extend the above definitions to the capabilities of a group of agents. What we could do is to decompose the given task set to some task subsets. The number of the task subsets is equal to or less than the number of agents in the multi-agent system. The decomposition of the task set should satisfy the capability of the corresponding individual agents in the group. Hence, to know whether the multi-agent system is capable of achieving tasks in a task set, we need to know if there exists a task assignment for all agents in the group, which satisfies the capabilities of the corresponding agents. Kleinman et al. [5] present a model of task distribution in human teamwork that has a similar flavor.

Suppose that M is composed of an agent group $\{A_1, …, A_m\}$. Given; a set of tasks X=$\{x_1, …, x_n\}$, which has the corresponding quality requirement set Q =$\{Q_1, …, Q_n\}$. For each task x_i in X, the release time and the deadline is the corresponding interval $\{r_i, d_i\}$, $i = 1, …, n$. The definition of the capability of a group of agent is as follows:

Definition 3: Capability of A Group of Agents

Group M has the *capability* to achieve the tasks in X or *CapGroup(M, X), iff* ∃ task assignment $Y = \{ Y_1, …, Y_m \}$, Y_j is the task subset of agent A_i and

$$\bigcup_{j=1}^{m} Y_j = X$$

CapSingle(A_j, Y_j) for $j = 1, \ldots, m$. This means there exist m task schedules S_j for each Y_j respectively, and each S_j satisfies the capability of the single agent A_j.

According to our definition of agent capabilities, the key issue of reasoning about agent capability is to search for a feasible task schedule that satisfies both the performance requirements of given tasks and the agent capabilities. For a multi-agent system, a task assignment algorithm is needed to satisfy the constraints of all agents in the system.

3. Task Scheduling

We should clarify what the exact meaning of "schedule" is in the above definitions. It is not just a linear sequence of a set of tasks. There could be some parallel tasks in a schedule. Also a schedule is not only searched by some mathematical scheduling algorithms [1], but also could be searched by pattern matching, rule-based reasoning or some other AI approaches [16] with satisfying the criteria in above definitions. As for scheduling algorithms, there exist many mathematical scheduling algorithms, which could be borrowed for our capability reasoning from the operating system, real-time system fields or manufacturing industry. One of the unique aspects in our scheduling algorithms is that the workload allocation and execution duration can be adjustable in accordance with the trade-off between workload and duration of a task. This feature helps an agent to schedule its tasks more flexibly and efficiently, but it makes the scheduling algorithm more complex. Also, the variance of the quality requirements of tasks will generate a different workload allocation. Therefore, besides considering the common constraints of tasks, such as dependence, preemption, release time, deadline etc. scheduling algorithms need to satisfy agent capability criteria.

Before introducing the formal scheduling algorithms, we need to make some assumptions. At first, we assume the workload allocation is not a function of time but a constant. That implies that the relation among effort requirement, workload allocation and execution duration of a task is described by the following equation:

$$effort = workload * duration; \qquad (3\text{-}1)$$

If the workload allocation of each task could be adjusted arbitrarily, the search space could be infinite. In our algorithms, we only consider two alternatives: one is the highest workload a task needs; the other is the lowest workload the task needs. Correspondingly, there are two alternatives of execution durations of the task: shortest duration and longest duration. As for the attributes of tasks, we assume all tasks in a task set are independent, meaning there are no predecessors or successors of each task. Tasks are also non-preemptive, meaning tasks could not be interrupted after they start to be executed.

We call the scheduling algorithm for agents in a static environment ERFS (Earliest Release time First Scheduled), meaning the earlier a task is ready, the earlier it is scheduled.

Algorithm 1: Earliest Release Time First Schedule

Input: Task set $X = \{x_1, \ldots, x_n\}$ with the corresponding quality requirements $Q = \{q_1, \ldots, q_n\}$. $(i = 1, \ldots, n)$. Each task x_i has the effective time interval $\{r_i, de_i\}$. Agent A has the maximum workload MC.

Output: A feasible schedule S of tasks accepted in X.

Begin (ERFS)

(1) Agent A computes effort e_i for each task x_i by a function of quality: $e_i = g(q_i)$;

(2) Agent A gets the workload ranges by functions:

$$w_{hi} = h(e_i) \qquad w_{li} = l(e_i)$$

where w_{hi} and w_{li} refer to the highest workload and lowest workload for task x_i respectively;

(3) Agent A gets the duration range by functions:

$$d_{si} = e_i / w_{hi} \qquad d_{li} = e_i / w_{li}$$

where d_{hi} and d_{li} refer to the shortest duration and longest duration for task x_i respectively;

(4) Set all tasks of X *acceptable* at first;

(5) Set tasks *unacceptable* if they satisfy $w_{li} > MC$ or $(r_i + d_{si}) > de_i$;

(6) Replace the w_{hi} by w_{li} and d_{hi} by d_{li} if $w_{hi} > MC$ but $w_{li} < MC$;

(7) Sort acceptable tasks by r_i in increasing order.

(8) Schedule these tasks by the order in (7) to get an original schedule $S = \{\{T_{s1}, T_{e1}\}, \ldots, \{T_{sm}, T_{em}\}\}$. m refers to the number of acceptable tasks.

(9) *If* S satisfy the deadlines of these tasks,

{*Return* S; }

Else

{ Put tasks without satisfying their deadlines in queue Qu. Let $S = S - Qu$;

Adjust all pairs $\{T_{sj}, T_{ej}\}$ in S to minimize the idle time which is caused by removing tasks from S;

Schedule the tasks in Qu parallel with S based on the principle: $\forall t, \sum w_k \le MC$; *t refers to time.*

Set tasks in Qu which could not be scheduled in S as *unacceptable*;

Return S; }

End (ERES).

In a dynamic environment, assume tasks arrive stochastically over time. The agent is executing the tasks in an existing schedule as well as scheduling the new arrival of tasks generated by the task generator. At any time, there is at most one new task generated. The agent gets the task before the release time of the task and schedules it without violating the currently executed schedule. We call the algorithm in this environment DS (Dynamic Schedule).

Algorithm2: Dynamical Schedule

Input: Task x has the quality requirement q and the feasible time interval $\{r, de\}$. Agent A is executing the tasks in the schedule $S = \{\{t_{s1}, t_{e1}\}, \ldots, \{t_{sn}, t_{en}\}\}$. The workload allocations for tasks in S is $W = \{w_1, \ldots, w_n\}$.

Output: Acceptance of x.

If it is acceptable, add x to S.

Begin (DS)

(1) Agent A computes effort e for each task x by a function of quality: $e = g(q)$;

(2) Agent A gets the workload range of x by functions:

$$w_h = h(e) \qquad w_l = l(e)$$

where w_h and w_l refer to the highest workload and lowest workload for task x respectively;

(3) Agent A gets the duration range by functions:

$$d_s = e / w_h \qquad d_l = e / w_l$$

where d_h and d_l refer to the shortest duration and longest duration for task x respectively;

(4) *If* $w_l > MW$ or $(r+d_s) > de$;

{ *Return* Acceptance = *false*;}

else

{Get the workload distribution of tasks in S by $dw(t)$;

Search the earliest interval {*start, end*}after release time r, in which the workload of x could satisfy the equation $w+dw(t) \le MW$, w is either w_h or w_l. *end* is equal to $start+d_s$ or $start+d_l$ depending on which is allocated, w_h or w_l. Also, *end* must satisfy $end \le de$.

If above searching is successful,

{ *Add* {*start, end*} to S and w to W;

Return Acceptance = *true*; }

Else

{ *Return* Acceptance = *false*; } }

End (DS).

DS extends the EFRS algorithm by attempting to place new tasks in the schedule of existing tasks, by trying various combinations of shortest-duration/highest-workload or longest-duration/lowest-workload variants of each task, while assuring that current tasks are not interrupted.

4. Experimental Evaluation

We designed two experiments to evaluate the above task scheduling algorithms. The goal of these experiments is to demonstrate that an agent could utilize its available workload more efficiently to achieve more tasks by reasoning about its capability quantitatively.

The framework of the first experiment consists of an agent, the task handler, which schedules tasks in a given task set generated by a task generator. The agent has two schedulers: one is without capability reasoning; the other is with capability reasoning. Here, the capability reasoning is based on the definition of agent capability. This means the agent can adjust its workload to a given task in order to achieve more tasks, and the agent is allowed to execute more than one task in parallel. In this experiment, the scheduler with capability reasoning schedules the tasks by the ERFS algorithm. Conversely, the scheduler without capability reasoning only schedules the tasks based on a fixed workload allocation, and the agent executes the tasks sequentially.

The task generator generates a random set of 1 to k tasks, each having a unique minimum quality threshold of between 0% and 100%. Each task is assigned a minimum and maximum duration of up to 100 time steps, and the workload range is set inversely proportional to the range of durations. Figure 6 shows the task acceptance rates of the schedulers in experiment #1, which is equal to the number of tasks accepted divided by the tasks generated. We set the parameter of the task generator k to {100, 300, 500, 1000, 2000, 3000, 5000}. For each k, we ran the agent 100 times and computed the mean of the tasks generated by the task generator. The result of Experiment #1 shows that the scheduler with capability reasoning can accept more tasks than the scheduler without capability reasoning for all parameters k we picked. We can see that the scheduler runs very stably. For different numbers of tasks, the acceptance rates are almost the same. The acceptance rate of the scheduler with capability reasoning is roughly twice that of the scheduler without capability reasoning.

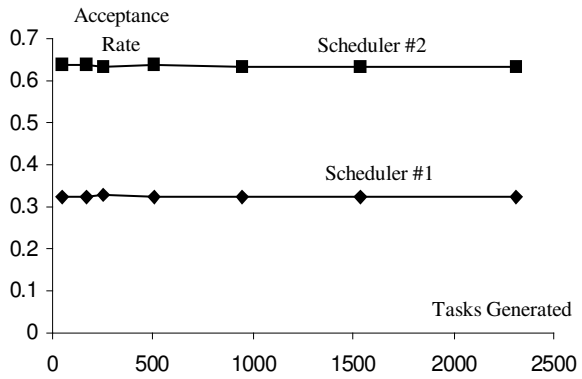

Figure 6: Task Acceptance Rates in Experiment #1

In our second experiment, the agent handles tasks generated by a task generator dynamically. There are two synchronized threads running in this experiment. One is running the task handler, or the agent, while another is running the task generator. There are two task streams generated by the task generator with different frequencies over the running lifetime of the agent. The time unit we use here is milliseconds. The tasks are generated similarly to those in Experiment #1, except that the durations are set randomly to between 1 and 1000ms. The frequency of task stream #1 is one task per 20 milliseconds. The frequency of task stream #2 is one task per 100 milliseconds. In Experiment #2, the scheduler with capability reasoning schedules the tasks by the DS algorithm. Each task scheduler in this experiment has to maintain the information of the tasks, which have been scheduled before in order to calculate the workload available at the current time. Each time the agent accepts a new task, the agent will update this information immediately.

We set the lifetime of the agent to 10,000ms and ran the agent 100 times to get the mean of the tasks generated by the task generator and the tasks accepted by the schedulers. On average, the task streams generated 462 tasks, of which the agent with capability reasoning accepted 117 (25%), while the agent without capability reasoning was only able to accept 64 (14%). Furthermore, the average workload (or resource utilization) was nearly twice as high: 50% for the agent with capability reasoning versus 28% without.

5. Discussions and Conclusion

The aim of this paper is to present a mathematical model for reasoning about the capabilities of task-performing agents. This model tries to capture the quantitative relationship among four performance constraints of given tasks and the executive agents: the quality requirements and time constraints of tasks, as well as the workload limitations and skill levels of the agents. Our model of capability is based on scheduling of concurrent tasks so as not to violate maximum workload capacity and/or time constraints.

This model of capability and workload estimation could be used in an incremental algorithm for constructing team plans. For example, as agents are assigned to tasks, estimates (predicted over time) of their workload might be updated, and assignment of subsequent tasks could be done so as not to exceed 100% workload for any one agent at any given time. However, it would probably be best to do this in a distributed way, since it is likely that no single centralized agent will know either the performance characteristics (task parameters such as speed, effort, and quality tradeoffs) of each agent, their other task demands (e.g. from private/individual goals or simultaneous participation in other teams), or other local resources or constraints. Furthermore, in accepting new tasks, agents might internally re-arrange processing of their responsibilities, and these accommodation decisions might not necessarily be known by the other agents. There are a number of multi-agent planning algorithms, and these can be extended to utilize this information in different ways to produce workload-balanced team plans.

Our model could also be used in handling different types of interactions between agents in a dynamic environment. For instance, in a dynamic and distributed environment, negotiation among agents requires them to reason about their own capabilities and sometimes each other's capabilities. For example, a client agent in a contract net [10] has to evaluate its capability to make a decision whether it bids a new task. The self-interested agents [8] may deceive its manager to get some bids to achieve a higher income, even if they cannot accomplish some tasks on time. In this case, the manager may need to reason about the capability of its bidders to avoid an unexpected delay of its task.

We view our framework as applying to reasoning about both agents and humans. Both have processing capacity limits that affect the assessment of their capability. Just as agents might have computational limits, such as on CPU speed or amount of RAM, so humans have limits on cognitive resources like memory and attention. For the case involving humans, agents need a quantitative model to understand the relative demands, capacity, and possibly skill level of the human, in order to make appropriate decisions about how to interact [3].

An important limitation of our current model is that it treats processing capacity as a single, unified resource. A significant direction for future research is to extend our model to incorporate multiple types of resources, so we can handle different dimensions of interaction based on the types of the tasks being performed. Also, we would like to incorporate different task priorities into the decision-making.

6. References

[1] Brucker, P. (1998). Scheduling Algorithms, Second Revised and Enlarged Edition. Springer.

[2] Dean, T. and Boddy, M. (1988). An Analysis of Time-Dependent Planning. In Proc. National Conf. on Artificial Intelligence, pp. 49--54, 1988.

[3] Ioerger, T.R., He, L, Lord, D., Tsang, P., (2002). Modeling Capabilities and Workload within Teams. in Proc. 24th Annual Conference of the Cognitive Science Society.

[4] Hoek, W., Linder, B. and Meyer, J.C., (1994). A logic of capabilities. *Proc.of the Third Int'l Symp. on Logical Foundations for Computer Science*, 366-378.

[5] Kleinman, D., Luh, P.B., Pattipati, K.R., and Serfaty, D. (1992). Mathematical models of team performance: A distributed decision-making approach. in Teams: Their Training and Performance. R. Sweezy and E. Salas (eds.). New York: Ablex.

[6] Kanfer, R. and Ackerman P.L (1989). Motivation and Cognitive Abilities: An Integrative/Aptitude-Treatment Interaction Approach to Skill Acquisition. Journal of Applied Psychology, Vol. 74, No. 4, 657-690.

[7] O'donnell, R.D. and Eggemeier, F.T. (1986). Workload assessment methodology. In K.R. Boff, L. Kaufman, and J. Thomas, Eds., Handbook of Perception and Human Performance: Volume II. Cognitive Processes and Performance. New York: John Wiley, Chap. 42.

[8] Sandholm, T. and Lesser, V. (1995). Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. Proceedings of the First International Conference on Multiagent Systems, 66-73.

[9] Schaerf, A., Shoham, Y. and Tennenholtz, M. (1995). Adaptive load balancing: A study in multiagent learning. *Journal of Artificial Intelligence Research*, 2:475-500.

[10] Smith, R.G. (1980) The contract net protocol: High-level communication and control in distributed problem solver. IEEE Transactions on Computers, 29(12):1104--1113.

[11] Singh, M.P. (1999). Know-how. in Foundations of Rational Agency, A Rao and M. Wooldridge (eds.), Kluwer. 105-132.

[12] Tambe, M. (1997). Towards flexible teamwork. *Journal of Artficial Intelligence Research*, 7:83-124.

[13] Tsang P. and Wilson, G.F. (1997) Mental Workload, in Handbook of Human Factors and Ergonomics, Second Edition. A Wiley-Interscience Publication, John Wiley & Sons INC. pp.417-449.

[14] Tulga, M.K. and Sheridan, T.B. (1980). Dynamic decision and workload in multitask supervisory control. *IEEE Trans. On Systems, Man, and Cybernetics*, 10(5):217-232.

[15] Wickens C.D. (1999). Attention, time-sharing, and workload. in Engineering Psychology and Human Performance. C.D. Wickens and J.G. Hollands (eds.), Prentice Hall. p. 439-479.

[16] Zweben, M. and Fox, M.S. (1994). Intelligent Scheduling. Zweben, M. and Fox, M.S. (Eds), San Francisco: Morgan Kaufmann Publisher.