Distributed Multiagent Resource Allocation with Adaptive Preemption for Dynamic Tasks*

(Extended Abstract)

Graham Pinhey
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
gpinhey@uwaterloo.ca

John Doucette
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
j3doucet@cs.uwaterloo.ca

Robin Cohen
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
rcohen@uwaterloo.ca

ABSTRACT

In this paper we present an approach for multiagent resource allocation that supports preemption in a fully distributed, cooperative setting (i.e. allowing for resources to be reassigned to agents, as the need arises, even as tasks are currently underway), in a way that copes with dynamic task arrivals.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms

Keywords

Agent Cooperation-Teamwork, Coalitions and Coordination; Agent Cooperation-Distributed problem solving; Humans and Agents-Agents for Improving Human Cooperative Activities

1. INTRODUCTION

In this work, we offer a new approach to multiagent resource allocation for a fully decentralized system. Our solution offers a novel combination of planning techniques for coordinating the allocation of resources with learning methods that allow agents to make effective local decisions. By utilizing these techniques, we are able to solve large scale problems even if faced with dynamic arrivals of new agents to the environment.

Similarly to Paulussen et al. [1] we allow our agents to accept personally disadvantageous trades, provided that the global utility will increase. Accurately determining whether such a trade is possible requires estimating the opportunity cost of having a preempted agent hold its resource (i.e. determining the agent's behaviour following a preemption and what the utility derived thereon will be). Whereas Paulussen et al. assume that cyclical preemptions result in the last agent in the cycle being left with no resource at all (and thus underestimate the ability to acquire new resources), we instead estimate the value of the actions that the dispossessed agent could take, without resorting to pessimistic evaluation.

*Financial assistance was received from NSERC.

Appears in: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Lomuscio, Scerri, Bazzan, Huhns (eds.), May, 5–9, 2014, Paris, France.*
Copyright © 2014, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2. SYSTEM AND ALGORITHM

There are four central components of our proposed system. **Tasks** represent a need for some user which must be satisfied by acquiring a resource (where the length of required time remaining in order to be satisfied is known). **Task agents** support tasks by mapping out plans (sequences of actions to attempt to acquire resources based on beliefs about expected value and probability of obtaining the resource) and sharing eu_{plan} (expected utility of executing the plan), thus enabling coordination. **Resources** are modeled as humans with an ability to be overwhelmed with requests. **Proxy Agents** support resources by filtering requests, promoting those where there is expected gain in utility.

The overall algorithm driving the system is one where agents negotiate for resources, making requests based on their modeling of the environment and holding contingency plans.

A task agent's plan consists of a list of actions to be taken in order, timestep by timestep. Each action is an attempt to acquire a particular resource, by asking the proxy agent associated with that resource for permission to use the resource. A task agent builds a plan by maximizing the valuation function in equation 1.

$$V(X) = EU(X_i) \times P(X_i) + (1 - P(X_i)) \times V(X \setminus X_i) \quad (1)$$

where X is the set of all possible actions the agent could take, $EU(X_i)$ is the expected gain in utility if the action X_i succeeds, and $P(X_i)$ is the probability of the action X_i succeeding. Both EU and P can be computed from domain-specific functions.

At each timestep, a task agent performs the action presently prescribed by its plan. It does this by contacting the proxy agent associated with the targeted resource, and asking the proxy whether it may take the resource. At the end of the timestep, the proxy agent computes the expected utility of allowing each request it has received, based on both its up-to-date local information about the resource and information provided by task agents about the state of each task making a request. The proxy filters requests, identifying the request that will produce the highest *positive* expected change in utility among all requests received and only allowing this best request to reach the resource. This minimizes the number of requests that the (human) resource needs to deal with, and ensures that any requests that do get through are as beneficial as possible in terms of improving social welfare. If the resource is currently in use by another task, the proxy agent computes the expected change in social

welfare (loss and gain) resulting from allowing each request to *preempt* the resource (rather than picking the task that gains the most). This can be computed using local information provided by the task that currently occupies the resource, including the expected utility of its contingency plan, computed with $V(X)$.

Agent learning is used to address the error in $V(X)$ resulting from task agents using their local (and potentially stale) information to compute $V(X)$. Task agents dynamically learn two properties of their environment (congestion and churn) which provide an approximate correction for this error. Congestion is the extent to which other task agents in the environment are seeking resources similar to those sought by this agent. It is estimated from the frequency with which the task agent's resources are preempted, and more frequent preemptions cause the task agent to reduce the perceived value of its contingency plans when in possession of a resource, thereby making preemption of its resource more difficult. Churn is the extent to which the task agent's environment is changing. It is estimated by having proxy agents report the newly computed expected utility of each request they receive back to the task agent that made the request. If the value computed by the proxy agent differs from that computed by the task agent, then the task agent's plan was based on information which has become stale. When this happens often, task agents will respond by generating plans with shorter horizons, so that they are also acting on recent information.

Our system is able to handle dynamic task arrivals because of its distributed nature. A new arrival to the system can generate an initial plan by finding a list of resources and then begin attempting to obtain resources immediately, on an equal footing with other task agents that are already in the system. In contrast, solutions like that of Paulussen et al. have residual benefits from remaining in the system for a long time, which disadvantage new entrants and may require a reset of the allocation system to accommodate them.

3. EVALUATION AND CONCLUSION

We ran a simulation that emulates the approach of Paulussen et al. [1] in order to evaluate the relative differences between our model and theirs. We projected our simulation into the domain of mass casualty incidents where patients are tasks and doctors are resources. We set the maximum plan length to 4, sampled dynamic arrival times of tasks from a uniform distribution over the first 50 timesteps. Note that we modeled the expected value of backup plans for the Paulussen model as the change in expected utility for the displaced agent switching to its next-best resource, minus the cost of displacing any agent who owns the backup resource, computed recursively (where the last agent in a cycle reports expected utility of losing without replacement). Our results are displayed in Figure 1, which shows the improvement of treatment times when using our algorithm. The benefits of using our system are most pronounced when it is possible to treat patients quickly, but high load makes it difficult to find a solution if the system does not prioritize properly (e.g. the 5 resources line in the middle graph). In all, we produce effective preemptive multiagent resource allocation with noticeable improvements over more pessimistic competitors through our integration of planning and learning, together with effective use of proxy agents for coordination and declaration of expected utility of backup plans for assessing global utility.

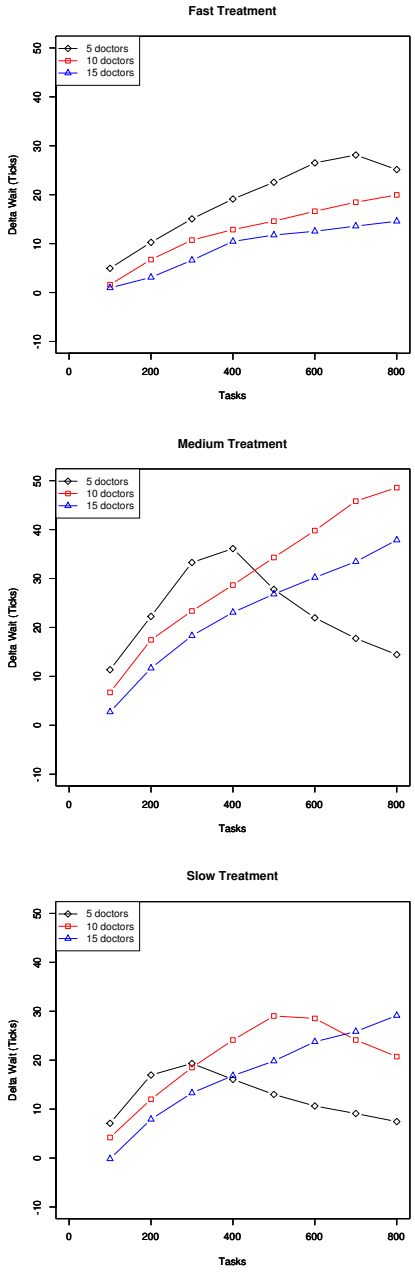

Figure 1: Mean improvements in treatment time over Paulussen. Points above 0 show the advantage of using the new algorithm.

4. REFERENCES

[1] T. O. Paulussen, A. Zöller, A. Heinzl, A. Pokahr, L. Braubach, and W. Lamersdorf, "Dynamic patient scheduling in hospitals," *Agent Technology in Business Applications*, 2004.