Multiagent Systems Viewed as Distributed Scheduling Systems: Methodology and Experiments

Sébastien Paquet, Nicolas Bernier, and Brahim Chaib-draa

DAMAS Laboratory, Department of Computer Science
and Software Engineering, Laval University, Canada
{spaquet;bernier;chaib}@damas.ift.ulaval.ca

Abstract. In this article, we present a design technique that facilitates the work of extracting and defining the tasks scheduling problem for a multiagent system. We also compare a centralized scheduling approach to a decentralized scheduling approach to see the difference in the efficiency of the schedules and the amount of information transmitted between the agents. Our experimental results show that the decentralized approach needs less messages, while being as efficient as the centralized approach.

1 Introduction

In this short paper, we present how a multiagent problem can be modelled as a task scheduling problem and how this formulation can help to find good scheduling algorithms. We define scheduling as the problem of assigning limited resources to tasks over time to optimize one or more objectives [1]. Furthermore, in multiagent systems, the scheduling can be done in a centralized or decentralized way [2] and in this article we study the impact on the agents' efficiency and on the amount of information transmitted when using these two approaches.

2 Modelling of the Tasks Scheduling System

In a tasks scheduling system, we use a set of resources to accomplish a set of tasks in an order maximizing an optimization criterion [3]. For example, we could want to accomplish the set of tasks as fast as possible or we could want to accomplish as many tasks as possible in a given time. This article focuses on multiagent systems in which the work of some agents can be described as a tasks scheduling system. So, agents are considered as resources that can complete tasks. Evidently, not all multiagent systems can be modelled as a task scheduling system. However, it can be possible in several cases, because in many multiagent systems, agents have to accomplish tasks and the order of these tasks influences the efficiency of the system. To structure the modeling process, we present a methodology with three steps: (1) scheduling problem definition, (2) scheduler type definition and (3) scheduling algorithm definition.

2.1 First Step: Scheduling Problem Definition

Firstly, we have to identify the characteristics of the scheduling problem which consists of defining the set of tasks to execute, the tasks' parameters (execution cost, deadline, etc.) and the optimization criterion. When the scheduling problem has been carefully analyzed, we can formalize it using the following notation [4]. A scheduling problem, with the goal of managing a set of tasks T, is described with three fields separated with the character "|", as in: $\alpha \mid \beta \mid \gamma$.

- α : the machines' environment. This field tells how many machines are present and what their characteristics are.

1 :	Mono-machine environment.
P_m :	Multi-machines environment with m machines.

- β : the constraints and the characteristics. This field tells for example, if the tasks have deadlines or if there is a cost to change from one task to another.

p_j :	The execution cost of the task j.
d_j :	The deadline of the task j.
s_{jk} :	The cost to change from task j to task k.

- γ : the optimization criterion. This field defines what the scheduler is supposed to optimize.

  $\sum C_j$: The sum of completion times of all tasks.
  $\sum U_j$: The sum of unit penalty of all tasks where:

$$U_j = \begin{cases} 1 & \text{if } C_j > d_j \\ 0 & \text{if not} \end{cases}$$

2.2 Second Step: Scheduler Type Definition

The scheduler, in a scheduling system, represents the abstraction level where the tasks ordering is decided in order to maximize the optimization criterion. We can create two main categories of scheduler: centralized and decentralized.

In the centralized approach, the tasks ordering is done by only one agent. This agent has to schedule and distribute the tasks for all the agents. To do that, it needs a global knowledge of the environment that it can acquire by exploration or by inter-agent communication.

In the distributed approach, the tasks ordering is not the responsibility of one agent, but many agents. All agents schedule their own tasks according to what they know about the environment. However, to stay coordinated, they need to synchronize themselves using inter-agent communication.

2.3 Third Step: Scheduling Algorithm Definition

In this third step, we have to choose the task scheduling algorithm to solve the problem defined in the first step. Many optimal and approximation algorithms already exist in the literature to solve different types of scheduling problems [5]. This shows the advantage of formalizing a multiagent problem in a scheduling formalism because, by doing so, we can search in the scheduling theory literature to find good algorithms for our problem.

3 Application to a Multiagent Problem

In this section, we show how to use our methodology in a multiagent system (the RoboCupRescue simulation) by explaining all the steps and specially focus on the scheduler part. This simulation environment consists of a simulation of an earthquake happening in a city [6]. The goal of the agents (representing firefighters, policemen and ambulance teams) is to minimize the damages caused by a big earthquake, such as civilians buried, buildings on fire and roads blocked.

In this article, we focus only on the work of the ambulance team agents. In the simulation there can be between 0 to 8 ambulance team agents that are in charge of rescuing civilians. The civilians are wounded when they are buried in collapsed buildings and they can die if they are not rescued fast enough. Rescuing a civilian is considered as a task and since the health state of a civilian is uncertain, the parameters of the tasks could change in time. Also, there are important constraints on the communications, thus it becomes critical to manage them efficiently.

3.1 Step 1: Scheduling Problem Definition

To rescue as many civilians as possible, ambulance team agents have to make a choice about in which order they will try to rescue the civilians. In this context, the task's duration is the time needed to save a civilian and the deadline of a task is the civilian's estimated death time.

Formally, our problem can be expressed as: $P_m|s_{jk}|\sum U_j$. The problem as it is defined has been proven to be NP-Hard [7]. Thus, we have to relax some constraints and make some changes to the original problem so that it can become solvable in polynomial time. First, we have relaxed the s_{jk} constraint by giving a value of 0 for each s_{jk} and by adding a constant travel time to the rescuing time. We also modified the scheduling problem definition by setting $m = 1$. In other words, it means that we consider that we have only one agent. Since this is not true, in practice it will result in all agents working on the same civilian because, for the scheduler, the group of agents is only seen as one indivisible resource.

Formally, our new problem can now be defined as: $1||\sum U_j$, which means that we consider a centralized execution in which all agents are considered to be one big resource working on one task at a time and trying to maximize the number of tasks accomplished in the time allowed.

3.2 Step 2: Scheduler Type Definition

In this step, we must choose between centralized and distributed scheduling. For this article, we compare both approaches in order to schedule the tasks of the ambulance team agents in the RoboCupRescue simulation.

In this article, we propose an implementation for each one of the two approaches and we compare their performance on the efficiency of the schedule and on the required amount of communication. Both approaches use the same scheduling algorithm, i.e. EDD (Earliest Due Date) [8]; we will justify this choice in section 3.3. This greedy algorithm orders a set of tasks by sorting them in increasing order of their deadline. An interesting property of this algorithm is that it is possible to have a distributed version that does not lose any efficiency.

With a centralized scheduler, there is one agent taking alone the decision about the ordering of the tasks. In brief, the steps of the scheduling process are:

1. All agents send their perceptions about possible tasks to the scheduler agent.
2. The scheduler agent combines all the information received.
3. The scheduler agent applies the EDD algorithm to schedule the tasks.
4. The scheduler agent sends the best global task to all agents.

As we can see, at step 1 all the agents send the information they know about all possible tasks. These messages can be quite long. Afterwards, at step 4, the scheduler agent sends the best global task to all agents, which then accomplish it.

In the decentralized approach, the scheduler is an entity composed of many agents. In brief, the steps of the scheduling process are:

1. All agents build their own list of possible tasks.
2. All agents apply the EDD algorithm to find the best local task to accomplish.
3. All agents broadcast their best local task to all other ambulance team agents.
4. All agents build a list with all the best local tasks received.
5. All agents apply EDD to find the best global task to accomplish.

As we can see, agents send messages only at step 3 and those messages are quite small because they contain only the information about one task.

3.3 Step 3: Scheduling algorithm definition

In the preceding step, we have already identified the scheduling algorithm, which is EDD. This algorithm can be executed in time $O(nlog_n)$ [9]. This type of greedy algorithm is well adapted to a problem of decentralized decision making because it is never necessary to reconsider a decision previously made. This enables us in practice to find the next task to accomplish in time $O(n)$. This algorithm is interesting because it is optimal if there is no overload, i.e. it is possible to accomplish all the tasks in the given time. Although some overload could happen in our environment, the performances of this algorithm stay good and its simplicity enables us to demonstrate how we can distribute the decision making in a scheduling system.

4 Results

The goal of these tests is to compare the performances and the communication burden of the decentralized scheduling approach compared to the centralized approach. For our experiments, we have created six different simulation scenarios.

Figure 1 presents the comparison between the performances of each approach. The centralized approach is slightly better in five scenarios out of six. However,

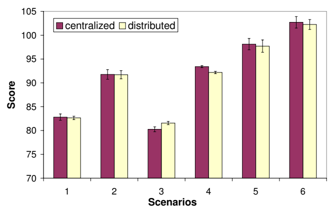

Fig. 1. Performances.

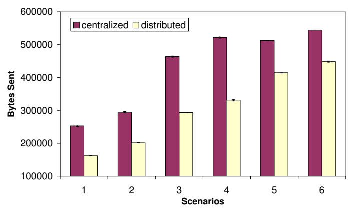

Fig. 2. Number of bytes sent.

this difference is really subtle and if we consider the 95% confidence interval, the two approaches can be considered equal.

However, the diminution of the communication burden is really at the advantage of the decentralized approach. Figure 2 presents the comparison of the number of bytes sent by the agents. On average, there is a 30% reduction of the quantity of information sent. This is mainly because the agents do not have to send all the information they know, but only the most interesting task.

5 Conclusion

In short, we have presented a methodology that can take advantage of algorithms developed in scheduling theory and to apply them in multiagent settings. Conversely, we have shown how we can use some multiagent ideas to decentralized a scheduling algorithm and gain on the amount of information transmitted.

We have demonstrated the efficiency of the decentralized approach in a complex environment (partially observable, uncertain and real-time). In brief, the decentralized approach is able to obtain the same performance with 30% less information sent. Future work could be to also distribute the execution and thus enabling the agents to be split on more than one problem at a time.

References

1. Mali, A.D., Kambhampati, S.: Distributed Planning. In: The Encyclopaedia of Distributed Computing. Kluwer Academic Publishers (1999)
2. Durfee, E.H.: Distributed Problem Solving and Planning. In Weiss, G., ed.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press, Cambridge, MA (1999) 121–164
3. French, S.: Sequencing and Scheduling. Wiley (1982)
4. Lawer, E., Lenstra, J., Kan, A.R.: Recent developments in deterministic sequencing scheduling : A servey. Deterministic and Stochastic Scheduling (1982) 35–74
5. Blazewick, J.: Scheduling computer and manufacturing processes. Springer (2001)
6. Kitano, H.: Robocup rescue: A grand challenge for multi-agent systems. In: Proceedings of ICMAS 2000, Boston, MA (2000)
7. Pinedo, M.: Scheduling: Theory, Algorithms and Systems. Prentice Hall (1995)
8. Jackson, J.R.: Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science, University of California (1955)
9. Brucker, P.: Scheduling Algorithms. Springer (2001)