Task Scheduling: A Decentralized Negotiation Method via GGP

Jieke Shi[1], Junwu Zhu[1] *, Xiaowei Zhang[1], and Yonglong Zhang[1]

College of Inform ation Engineering, Yangzhou University, Yangzhou Jiangsu, China
jwzhu@yzu.edu.cn

Abstract. In this paper, a real-time decentralized task scheduling algorithm is used to enhance human disaster rescue capability by using rescue agents and other resources in some scenarios for tasks with contiuous damage. We mainly focus on finding the optimal task scheduling strategy. We first introduce the task scheduling problem we are tackling, then we design an automatic negotiation framework, which realizes the real-time decentralized automatic negotiation of agents. Beyond this, using General game playing (GGP) technology as a tool, an automatic negotiation algorithm is implemented, which enables agents to adjust their plans dispersively. By comparing a heuristic algorithm based on greedy strategy, we prove that the efficiency of our strategy is about 8% lower than that of the centralized method in the ideal case of unlimited communication, and the algorithm is more effective than the centralized algorithm under communication range restriction.

Keywords: Task scheduling · Automated negotiation · Multi-agent System · Decentralized algorithm· General game playing.

1 Introduction

With the rapid development of technology, multi-agent system has been applied in more and more fields. Compared with centralized artificial intelligence, distributed multi-agent system has more advantages in executing tasks, and can accomplish disaster management, rescue management, industrial task scheduling and other scenarios. The key problem we need to study is how to allocate tasks accurately according to the working environment and situation, make full use of the collaborative function of agents, and improve the efficiency of the whole system.

Under normal circumstances, completing tasks can provide some payoffs for the system. The first goal of task allocation is to maximize the utility of the whole system. However, in many rescue tasks, the completion of the task will not bring any positive returns to the system, and the task will continue to destroy the system. For example, there is a burning building in a city. Fire did not generate any positive income. An intuitive conclusion is that the greater the

*Corresponding author.

fire, the greater the damage to the surrounding environment. Our goal is to find a scheduling scheme that minimizes the total damage to the system caused by all tasks.

In this paper, a real-time distributed task scheduling algorithm in rescue scenarios is proposed. We first introduce the task scheduling problem we are solving, and then design an automatic negotiation framework to realize the real-time distributed automatic negotiation of agents. Beyond this, using GGP technology as a tool, an automatic negotiation algorithm is implemented, which enables agents to adjust their plans dispersedly. Finally, the experimental part is to verify the superiority of our algorithm.

2 Related Work

Task assignment and scheduling are key issues in the research of multi-agent systems. This problem can be expressed as a set of given tasks and agents. Appropriate strategies should be adopted to assign different agents to corresponding tasks and optimize certain specific objectives. There is a lot of research work in this area. Finding an optimal task allocation problem has been proved to be NP-accomplished[2], and task allocation can be divided into centralized control and distributed control methods[1, 10].

In a centralized system, one of the agents has a higher status and is assumed to have the complete information of the system, that is, the central agent. The central agent calculates the optimal or approximate optimal decision, which maximizes the effectiveness of the multi-agent system. Centralized decision-making process is always modeled as integer programming[8][3], but the results of task assignment may not be able to achieve the goal in an ideal time[5].

Although these methods can find the best solution, it is difficult to satisfy the requirement of complete information[7], which increases the need for distributed or decentralized allocation methods. Negotiation is an effective way for distributed agents to cooperate with each other[6, 11, 9]. However, in many works, the agent is not independent of the domain protocol[12].

General Game Playing is a relatively new topic. Although earlier work has been done, it really started to draw widespread attention in the AI community after the introduction of GDL [4] and the organization of the annual AAAI GGP competition since 2005.

3 Problem Description

In this section we introduce the problem we are tackling. The task scheduling problem can be concluded that for a given set of tasks and agents, a proper strategy should be applied to assign various agents to corresponding tasks, optimizing some certain targets. In this model, $R = \{1, 2, ..., N\}$ is the set of agents. It means that there are N agents available to be allocated. $T = \{1, 2, ..., M\}$ is the set of tasks.

For a agent $i \in R$ in the task, c_i represents the agent i can complete in a unit time. For a task $j \in T$, we define that $w_j^t \geq 0$ is the remaining workload of task j at time t. Time in this model is discrete to be each single time point. A interval like $[t, t+1)$ is used to describe a period of time. C_j^t is the number of agents assigned to task j at time t.

The remaining workload of a task only relies on the initial setting and scheduling results at each time point. The changing of workload can be represented by the following recursion formula:

$$w_j^{t+1} = w_j^t - \sum_{i=1}^{C_j^t} c_i \tag{1}$$

According to this formula, the agent allocation strategy at time t will not change the remaining workload instantly. In our model, agents work on tasks at time period $[t, t+1)$, updating remaining workload of tasks at time point $t+1$.

One of the main innovations of this model is that tasks with continuous and dynamic damage are firstly introduced. We define a function $H : w \to R$ to quantity such damage. At some time point t, $H(w_j^t)$ represents the damage task j causes to the system during time period $[t, t+1)$ as w_j^t is the remaining workload of j at time t.

If the initial workload is known, the time when all tasks are finished can be easily calculated:

$$t_d = \frac{\sum_{j=1}^{M} w_j^0}{\sum_{i=1}^{N} c_i} \tag{2}$$

The target of our model is that when the completing time t_d is determined, choosing the proper reallocation plan at each time point to minimize the total damage suffered by the system, that is

$$\min \sum_{t=0}^{t_d} \sum_{j=1}^{M} H(w_j^t) \tag{3}$$

4 A Real-time Decentralized Scheduling Algorithm

4.1 A General Automated Negotiation Framework

We model the automated negotiation in agent task allocation scenarios with GGP. According to some GDL and automated negotiation researches, we initially define a framework at agent task allocation automated negotiation as follows:

- $Ag = \{a_1, a_2, \cdots, a_n\}$ is the set of agents who participate in a negotiation. The negotiation host is not included in Ag.
- $Ac = (Ac_1, Ac_2, \cdots, Ac_n)$ is a tuple in which Ac_i represents the action set of negotiation agent $i \in Ag$.

- W is a non-empty set of states. $w_0 \in W$ is the initial state of a negotiation. $T \subset W$ is the set of terminal states $w \in W$.
- $L = (L_1, L_2, \cdots, L_n)$ is a tuple where each $L_i : W \backslash T \to 2^{A_i}$ is the legal function for a_i. It determines the actions a_i can take at some state.
- A state update function is defined as $u : W \times Ac_i \to W$ to represent a state will change to another at what condition.

Such definition of negotiation is unified GGP , which allows us to use GDL to describe automated negotiation scenarios in agent task allocation. We use GDL to implement this negotiation protocol on General Game Playing platform. It is of little meaning to compare the running time of our negotiation method and the centralized method because the negotiation method requires a branch of communication among agents, which is time-consuming. However, the actual computational burden of the negotiation method will be shared by each agent participating in the task, so it is more practical in the real world.

4.2 Applicable Algorithms

Now we propose a distributed task allocation algorithm. We view each task as agents and call them "task agent". We assign one of them as the central agent. Although there is still a so-called central agent existing in this method, it just aims to maintain the negotiation and do little computation. It can be replaced by other external agents. This distributed method can be described as follow:

Step 1. For all $j \in Ag$, agent j proposes two values:$u_j(1)$ and $u_j(-1)$. The central agent combines all those values as two sequences $U(1)$ and $U(-1)$. If $\min U(-1) > \max U(1)$, the algorithm terminates.

Step 2. Assuming that $u_i(-1)$ is the minimal in $U(-1)$, the corresponding agent i picks agent $j \neq i$ whose $u_j(1)$ is maximal in $U(1)$. If the condition $u_i(-1) < u_j(1)$ is satisfied, agent i will transfer a agent of itself to agent j then both agent i and j is deleted from $U(1)$ and $U(-1)$. If the condition is not satisfied, algorithm returns to Step 1.

This procedure can be represented in pseudo-code as Algorithm 1.

We assume that at some time t the amount of agents at task j is C_j^t. Then the damage caused in $[t+1, t+2)$ is $H_j(w_j^t - C_j^t)$. If we add one more agent to task j at time t, then the damage changes into $H_j(w_j^t - C_j^t - 1)$. We define $u_j(1) = H_j(w_j^t - C_j^t) - H_j(w_j^t - C_j^t - 1)$. Similarly, $u_j(-1)$ is defined as $H_j(w_j^t - C_j^t + 1) - H_j(w_j^t - C_j^t)$. There are some special cases like $C_j^t = 0$, $C_j^t = w_j^t$ or $C_j^t = N$. In these cases, $u_j(1)$ or $u_j(-1)$ is assigned as -1.

4.3 Effectiveness Against with Centralized Strategy

In this section, we show the application of our centralized strategy in *RoboCup Rescue Simulation*(RCRS). RCRS is a simulation of disaster response scenarios in large cities (for more information, see http://www.robocurescue.org).

Algorithm 1: A real-time decentralized allocation algorithm

```
repeat
    U(1) = U(−1) = ∅;
    for each i ∈ Ag do
        propose(u_i(−1), u_i(1));
        U(1) = U(1) ∪ u_i(1);
        U(−1) = U(−1) ∪ u_i(−1);
        repeat
            i = arg min U(−1);
            delete i from U(1) and U(−1);
            j = arg max U(1);
            if u_i(−1) < u_j(1) then
                agent i gives a task to agent j;
                delete j from U(1) and U(−1);
            else
                Algorithm Stops
        until true
until true
```

We did two batches of experiments. Firstly, the communication range of the agent is not limited. Figure 2 shows that, the solution of distributed generation is less than 8% of the centralized method. Considering comprehensively, these results show that the distributed method is a good approximate solution in RCRS scenarios. More specifically, both algorithms perform better in Kobe and Foligno scenarios than in Berlin scenarios, we think it is because Berlin is a larger and unreasonably structured map than Kobe or Foligno, it is difficult to complete rescue tasks there quickly. Our algorithm will lead to the aggregation of agents, which means that agents will deal with the same tasks after negotiation and transfer to new tasks together after dealing with the current tasks.

In the case of ideal computing state and complete communication, this heuristic algorithm can lead to better completion of all tasks by agents between tasks. However, under the circumstance of limited communication range, serious resource allocation errors may occur in both algorithms, but the performance of the distributed algorithm remain relatively stable.

5 Conclusion and Future Work

In this paper, we are the first to introduce a novel and practical kind of task into agent resource allocation field, which we call tasks with continuous and dynamic damage. We propose a decentralized allocation algorithm for agents to adjust schedules themselves. By comparing a heuristic algorithm based on greedy strategy, we prove that our algorithm is more effective than the centralized algorithm under communication range restriction.

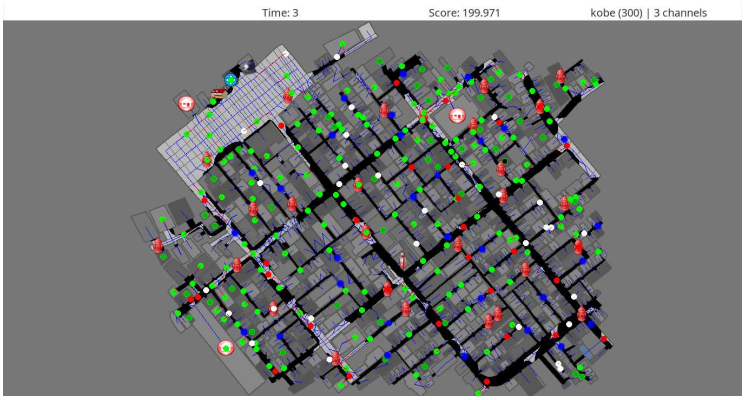

Fig. 1. Example of a map used in RoboCup Rescue Simulation.

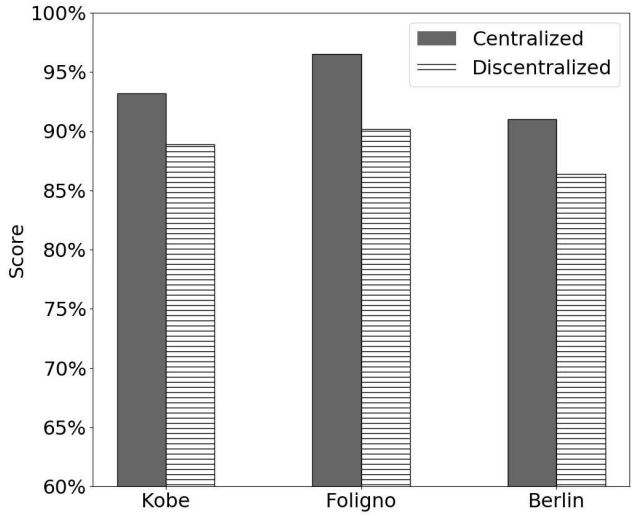

Fig. 2. Comparing the algorithms across three maps under unrestricted communication.

Due to limited abilities of authors, there are many shortage in this work. In reality, the workload of a task will grow naturally, and the transfer time of reallocation is not considered in this paper. In the future work, we will take these two parameters into consideration to modify the model and expanded GDL like SGL[13, 14] to implement protocols and strategies is a research direction as well.

References

1. Amador, S., Okamoto, S., Zivan, R.: Dynamic multi-agent task allocation with spatial and temporal constraints. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. pp. 1495–1496. International Foundation for Autonomous Agents and Multiagent Systems (2014)
2. An, B., Lesser, V.: Characterizing contract-based multiagent resource allocation in networks. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **40**(3), 575–586 (2010)
3. Booth, K.E., Nejat, G., Beck, J.C.: A constraint programming approach to multi-robot task allocation and scheduling in retirement homes. In: International Conference on Principles and Practice of Constraint Programming. pp. 539–555. Springer (2016)
4. Genesereth, M., Love, N., Pell, B.: General game playing: Overview of the aaai competition. AI magazine **26**(2), 62 (2005)
5. Hooshangi, N., Alesheikh, A.A.: Agent-based task allocation under uncertainties in disaster environments: An approach to interval uncertainty. International Journal of Disaster Risk Reduction **24**, 160–171 (2017)
6. Jin, L., Li, S.: Distributed task allocation of multiple robots: A control perspective. IEEE Transactions on Systems, Man, and Cybernetics: Systems (2016)
7. Korsah, G.A., Stentz, A., Dias, M.B.: A comprehensive taxonomy for multi-robot task allocation. The International Journal of Robotics Research **32**(12), 1495–1512 (2013)
8. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics (NRL) **2**(1-2), 83–97 (1955)
9. Ramchurn, S.D., Farinelli, A., Macarthur, K.S., Jennings, N.R.: Decentralized coordination in robocup rescue. The Computer Journal **53**(9), 1447–1461 (2010)
10. Shi, J., Yang, Z., Zhu, J.: An auction-based rescue task allocation approach for heterogeneous multi-robot system. Multimedia Tools and Applications pp. 1–10 (2018)
11. Stranders, R., Tran-Thanh, L., Fave, F.M.D., Rogers, A., Jennings, N.R.: Dcops and bandits: Exploration and exploitation in decentralised coordination. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. pp. 289–296. International Foundation for Autonomous Agents and Multiagent Systems (2012)
12. Wang, G., Wong, T., Wang, X.: An ontology based approach to organize multi-agent assisted supply chain negotiations. Computers & Industrial Engineering **65**(1), 2–15 (2013)
13. Zhang, D., Thielscher, M.: A logic for reasoning about game strategies. In: AAAI. vol. 15, pp. 1671–1677 (2015)
14. Zhang, D., Thielscher, M.: Representing and reasoning about game strategies. Journal of Philosophical Logic **44**(2), 203–236 (2015)