Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 94 (2016) 386 – 391

Procedia Computer Science

The 2nd International Workshop on Communication for Humans, Agents, Robots, Machines and Sensors
(CHARMS 2016)

A Realistic Decision Making for Task Allocation in Heterogeneous Multi-agent Systems

Yongho Kim and Eric T Matson*

M2M lab, Computer and Information Technology, Purdue University, Indiana, United States

Abstract

Task allocation is one of the keys to maximize organizational benefits by handling as many tasks as possible. Many computational multi-agent systems use agent's *capability* for task allocation. When a task arrives at the queue to be delivered a task allocator will determine which takes the task by finding the best-capable agent. In real world situation, each agent should not only consider the new task with their capability, but also tasks that they are currently handling before sending their capability to the task allocator. This research study proposes a CPU-scheduling based algorithm to allow agents to perform rational decision making when they think that they can handle the new task while taking care of its current tasks. The result shows that applying algorithm provide a significant improvement of their performance.

Keywords: Multi-robot Task Allocation (MOTA); Multi-agent Systems (MAS); Scheduling algorithms;

1. INTRODUCTION

Entities of multi-agent systems (MAS) are designed to handle assigned tasks to achieve goals. As they are part of an organization in systems, accomplishing goals is one of their duties as a member and it usually benefits to the organization as well as to the individuals. When agents are cooperative they can be considered as socially-interested agents which seek to maximize social profits for the organization. The *unloading lorries in a warehouse*[1] example evidently showed how socially responsible agents improved the overall system performance by helping each other, which maximizes social benefit.

In order to have maximized social benefit, job allocator — who can be an external agent such as humans or an internal agent which oversees the organization — should assign tasks in a rational manner. Even though '*being rational*' has several meanings depending on context, it could mean in this case that the allocator can analyze the goal to decompose it into necessary tasks and assign the tasks to the best-capable agents, as leaders in human society do.

* Corresponding author. Tel.: +1-765-494-8259 ; fax: +0-000-000-0000.
E-mail address: kim1681@purdue.edu, ematson@purdue.edu

doi:10.1016/j.procs.2016.08.059

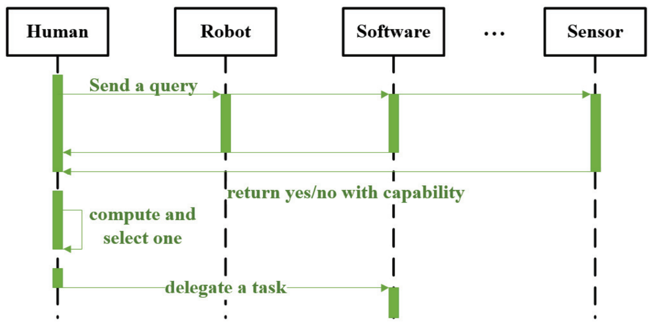


Fig. 1. A sequence diagram of the target scenario.

There are two properties in MAS that make the job allocation process challenging. Some computational models[2,3,4], which are the target environment in this research study, followed distributed artificial intelligence (DAI). Agents in the models have their own decision making capability and actively interact with others to determine its next action. It is difficult for one agent to decide and distribute tasks because it may not have enough information of others such as intentions and desires to proceed. The second property is heterogeneity. Unlike homogeneous models such as ant colony[5], bird flocking[6], and swarm-bots[7], agents in heterogeneous models have more capabilities which increases complexity in decision making process for task allocation.

Fig. 1 shows a flow chart of the target scenario in a heterogeneous MAS that follows DAI. In the scenario the human operator, the job allocator and also be treated as an agent, is trying to assign a task to one of the members in the organization. The most intuitive way to do it is finding the best-capable agent for the task. The following procedure describes three steps for the task allocation:

1. Acquisition: the allocator sends a message to the rosters to get a list of capability for the task. We assume that the rosters are cooperative so that they are honest in sending their capability value to the requester.
2. Selection: According to the list of capability for the task, the allocator selects one agent to delegate. There are numerous ways to determine the selection, but the agent which has the best capability for the task will be selected in this scenario. If more than one agent with the same capability become candidates, the allocator randomly selects one.
3. Delegation: the allocator sends a command to the selected agent. It also sends the detail information of the task, e.g., instructions, deadline, etc.

Even though there are lots of debatable facts in the scenario, we ignore them and focus on the steps mentioned above. For example, if the allocator once knows the rosters capability on a task, it does not need to ask again for the same task in the future. Another debatable issue would be sending a query to only the capable rosters which have been proved (or desired) for the task to reduce network traffic. We also assume that all agents are capable of interpreting messages when communicating.

When the scenario is applied to real world, the responding agents should be able to determine whether or not it can take the new task at the moment before replying its capability. For instance, if an agent is working on a task, it may not be able to take the new task to do. However, it may be possible that the agent holds the current task and proceed the new task if the current task has enough deadline. This could improve performance of the organization, e.g, higher quality of results. Thus, this research study investigates how agents can arrange tasks with the given deadlines, as known as *scheduling* problem. We propose a strategy of such scheduling that guides agents better way in answering the query to job allocator.

2. RELATED WORK

Despite of the aforementioned challenge in decision making, heterogeneous models allow agents to be flexible and scalable for dealing with complex high-level goals. Because agents have different capability, they could cooperate to accomplish the given task more efficiently[8]. Most of computational models such as OMACS[3], gaia[9], and R-object[10] considers *capability* a factor to allocate tasks; the best-capable agent to the task will be selected and assigned (needless to say that not capable agent to the task will not be considered at the task allocation). The capability is typically determined by system designer at the beginning and not likely to be changed unless agents loss their capability by physical or logical damages.

Multi-robot task allocation (MRTA) is a problem of how to assign tasks to members to maximize their benefits, which is associated with finding the optimal solution. According to Ferber[11], the scenario described in the introduction is categorized in '*predefined distributed allocation*'. In this case, the contract net protocol[12], specialized in solving distributed problem, has been used[13] and utilized in advance[14]. Utility is a computable factor that a system can earn by performing tasks and used to optimize system performance. In general, utility is the value that the system can earn as profit subtracted by the amount of resources it used to deliver the given task. Most research studies considered resources to calculate utility for optimization purpose[15,16].

The way an agent determines whether or not it can be assigned the new task is, however, to have an ability to check the schedulability of the agent with its current task(s) and the newly added task. For example in the contract net, an agent could bid if it thinks that it can take the new task and can get it done by deadline of the task; otherwise, it would not bid. This implies that the decision will be made using information gathered locally because decision making process is done by the agents themselves. In fact, one of the characteristics of MAS that follows DAI is that agents have not enough information to reveal a *global/optimal* solution[17], and thus this study will not pursue the true optimal solution, but will try to find sub-optimal solutions that satisfies the given condition. Page et al.[18] presented a heuristic task allocation algorithm using genetic algorithm (GA) to dynamically allocate tasks to the processors. In this study, various types of scheduling such as Round Robin, earliest first, and lightest loaded have been used to test performance of the system.

3. SCHEDULING FOR DECISION MAKING

According to the taxonomy of task allocation[19,20], the problem this paper deals with is a problem of single-task, single-robot, and instantaneous assignment (ST-SR-IA). This means that a system only considers current state to assign an independent task to a robot that can handle one task at a time. We assume that the agents have unlimited resources such that utility only considers the profit they can earn by performing tasks in that the goal is to maximize cumulative utilities over time with a fixed number of agents. Agent's capability to a task that varies from [0, 1] determines how well the agent can handle the task.

The organization of the problem follows OMACS[3] and HARMS[4] model such that the organization is characterized as 1) agents are social so that they are eager to take tasks, 2) capability-based allocation is required, and 3) decision making of whether the agent can take a task is done locally. A task requires a specific amount of effort to be finished and has a hard-deadline, which never passed; otherwise, the organization gets a huge penalty or ceases to exist. Tasks are preemptive and have dynamic-priority based on their deadline.

As shown in Fig 1, each agent has to answer to the query with their capability of the task in the acquisition phase. In order to determine answer of the query, the scheduling algorithms that have been used in processor scheduling in computer science can be used. This study initially considers earliest deadline first (EDF) algorithm because the task has hard-deadline. Each task requires a certain amount of time e to be completed and has a deadline d that could not be exceeded. Utility can be expressed as,

$$\begin{aligned} u_i &= t_i max(c_n) \\ U &= \sum u_i \end{aligned} \tag{1}$$

where u_i is utility value of the i-th task, t_i is a value of importance of the task, c_n is a capability value bid from agent n, and U is total utility of handling the given tasks. In this study the t_i is 1 for all tasks as it means that the tasks are equally important (i.e., equal priority in terms of type). For example, *catching fire* task would have larger t value than *delivering mails* task. The $max(c_n)$ is the maximum c among c received from agents and can be achieved when the job allocator gets all responses from agents. The procedure, shown in Algorithm 1, runs from agent's side and checks if the new schedule regarding the new task satisfies the condition that it does not exceed deadlines of the tasks. Virtual

Algorithm 1 Decision making process with EDF scheduling

```
procedure GetCapability
    Let $Cap$ a capability value of the new task ←-1
    Create a virtual task queue called $VTQ$
    Clone $VTQ$ with the current task queue
    Add the new task into $VTQ$
    Sort $VTQ$ by deadline in ascending order
    for $i = 0 \rightarrow VTQ.Length$ do
        $Cumulative\_e \leftarrow Cumulative\_e + VTQ.e[i]$        ▷ $e$ is a required amount of time to finish a task
        if $Cumulative\_e > VTQ.d$ then
            Set $Cap$ zero, indicating not capable of taking the new task
    if $Cap$ = -1 then
        Set $Cap$ the capability value of the new task
    return $Cap$
```

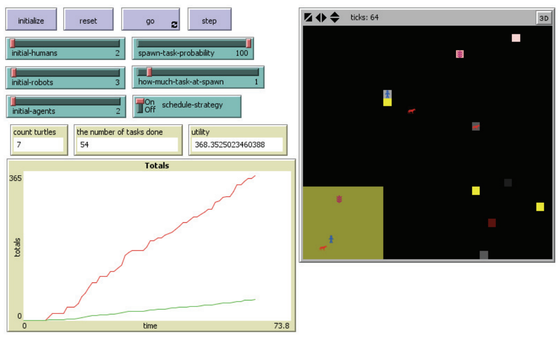
task queue (VTQ) takes all current tasks and the new task to check the schedulability. The agent will respond with its capability or zero to the new task based on whether or not VTQ is schedulable. The capability value is measurement of performance against performing the task and is determined in design phase. The value could dynamically change since the robot's actual capability to the task could differ depending on a situation and time. However, it is assumed to be static in this paper because having less variable in the system would give higher significance in performance. Time complexity of the procedure is the same as $O(n)$ for both job allocator and bidder.

4. EVALUATION

In order to evaluate performance of the scheduling algorithm we proposed, the algorithm is simulated in NetLogo[21] environment. NetLogo is a tool that simulates behaviors of agents and supports interactions between agents and nearby environment. The scheduling algorithm is written in Java as a library and used as an extension[22] in NetLogo.

Fig. 2 shows graphical user interface (GUI) of the simulation in NetLogo. When initialized, agents are spawned according to the number of agents set by operator. Agents consists of humans, robots, and software agents. The GUI determines how often and how many tasks are created in one discrete time. Left-bottom of the GUI shows a graph of how many tasks are completed and the corresponding U. Right side of the GUI is an environment in which spawned agents deliver tasks that come continuously.

In order to evaluate the algorithm, simulations were conducted using two groups: scheduling-enabled (SE) and non-scheduling(NS). In SE group agents use the proposed algorithm and response their capability based on result of



initial-humans 2
spawn-task-probability 100
initial-robots 3
how-much-task-at-spawn 1
initial-agents 2
schedule-strategy
count turtles 7
the number of tasks done 54
utility 368.3525023460388

Fig. 2. NetLogo environment for the simulation. Up-left side consists of control buttons while bottom-left and right side show progress and result of the simulation. Bottom-left corner of the right side image is base of the agents where they await for tasks. Yellow spots on the right side image are newly generated tasks and the spots occupied by the agents are tasks that is being delivered.

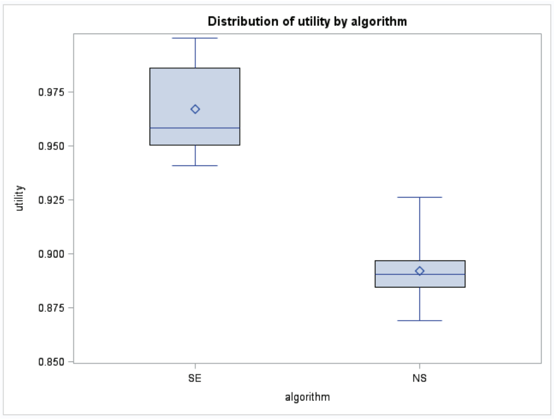


Fig. 3. A boxplot of the simulation result. The U is normalized [0, 1] to compare means of the two cases.

the algorithm. NS group, however, assigns tasks to the best capable agent if the agent is idle, i.e., waiting for task. The agents in the two groups have a matched capability value in order to see difference in their total utility U. For example, if the first agent in SE group has 0.5 capability value for task i, the first agent in NS group also has the same capability for the same type of task. Effort e and deadline d of a task for both groups are randomly determined every time a task is spawned. d is always larger than e and both can be equal, meaning the task requires immediate assignment. Both groups had 500 discrete time for each simulation. Each time a task was spawned with its d and e. Task allocation process was performed every time if there is new tasks that have not been assigned.

Fig. 3 compares difference between the two groups using 10 samples collected from both groups in the simulation. The difference is clearly shown and U of the SE group is significantly larger than NS group. A one-side $t - test$ was also conducted and concluded that the mean of utility of SE group is larger than NS group (p-value is less than 0.0001).

However, when number of agent is less than the maximum e the simulation was likely to be terminated due to deadline-passed task. This fact is understandable because it is obvious that smaller number of agent cannot deliver tasks that require larger efforts. The interesting fact during the simulation was that SE group encountered higher number of termination than NS group. We believe the reason is because agents in SE group tend to take more tasks if possible so that they cannot handle immediate tasks which have larger e when their task queue is almost full. Figuring out what independent/lurking variables affect to the termination would improve U even much higher.

5. CONCLUSION AND FUTURE WORK

Task allocation in DAI-enabled heterogeneous MAS is challenging because each agent has incomplete information to rationally determine which agent takes which task. Considering only an organization view says that it is better to allocate tasks to the best-capable agent as much as possible in order to maximize their benefit. This study proposed a CPU scheduling-based algorithm to allow agents to perform rational decision for the organization. Because the structure of whether or not they take a task is similar with CPU scheduling that has multiple cores (i.e., agents), we believe that more advanced scheduling theories and algorithms, e.g., Round Robin, can be applied to solve the task allocation problem.

The future work of this research study will 1) elaborate characteristics of tasks (e.g., importance of tasks), 2) finding independent variables that affect to performance of the system in terms of utility, and 3) investigate potential solutions to the problem caused by deadline-passed tasks. In particular, applying boost mechanism of agents could deal with tasks that nearly pass its deadline. For instance, an agent could deliver task twice faster by consuming more energy.

Acknowledgements

This material is based upon work supported by the National Science Foundation under the Grant Number CNS-1239171.

References

1. Kalenka, S., Jennings, N.R.. Socially responsible decision making by autonomous agents. In: Cognition, Agency and Rationality. Springer; 1999, p. 135–149.
2. Fischer, K., Schillo, M., Siekmann, J.. Holonic multiagent systems: A foundation for the organisation of multiagent systems. In: Holonic and Multi-Agent Systems for Manufacturing. Springer; 2003, p. 71–80.
3. Deloach, S.A., Oyenan, W.H., Matson, E.T.. A capabilities-based model for adaptive organizations. Autonomous Agents and Multi-Agent Systems 2008;16(1):13–56.
4. Matson, E.T., Min, B.C.. M2m infrastructure to integrate humans, agents and robots into collectives. In: Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE. IEEE; 2011, p. 1–6.
5. Parunak, H.V.D.. "Go to the ant": Engineering principles from natural multi-agent systems. Annals of Operations Research 1997;75:69–101.
6. Olfati-Saber, R.. Flocking for multi-agent dynamic systems: Algorithms and theory. Automatic Control, IEEE Transactions on 2006;51(3):401–420.
7. Mondada, F., Pettinaro, G.C., Guignard, A., Kwee, I.W., Floreano, D., Deneubourg, J.L., et al. Swarm-bot: A new distributed robotic concept. Autonomous Robots 2004;17(2-3):193–221.
8. Kiener, J., Von Stryk, O.. Cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. In: Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. IEEE; 2007, p. 959–964.
9. Wooldridge, M., Jennings, N.R., Kinny, D.. The gaia methodology for agent-oriented analysis and design. Autonomous Agents and multi-agent systems 2000;3(3):285–312.
10. Park, J.W., Son, Y.S., Jung, J.W., Oh, S.M.. R-object model for evolutionary robots using multi-robot cooperation. In: Intelligent Control Systems and Signal Processing; vol. 2. 2009, p. 438–443.
11. Ferber, J.. Multi-agent systems: an introduction to distributed artificial intelligence; vol. 1. Addison-Wesley Reading; 1999.
12. Smith, R.G.. The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on computers 1980;(12):1104–1113.
13. Tripathi, A.. Multi agent system in job shop scheduling using contract net protocol. International Journal of Computer Applications 2014;94(16):24–29.
14. Liang, H., Kang, F.. A novel task optimal allocation approach based on contract net protocol for agent-oriented uuv swarm system. Optik-International Journal for Light and Electron Optics 2016;.
15. Kaihara, T.. Multi-agent based supply chain modelling with dynamic environment. International Journal of Production Economics 2003;85(2):263–269.
16. Chen, J., Sun, D.. Resource constrained multirobot task allocation based on leader–follower coalition methodology. The International Journal of Robotics Research 2011;30(12):1423–1434.
17. Sycara, K.P.. Multiagent systems. AI magazine 1998;19(2):79.
18. Page, A.J., Keane, T.M., Naughton, T.J.. Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system. Journal of parallel and distributed computing 2010;70(7):758–766.
19. Korsah, G.A., Stentz, A., Dias, M.B.. A comprehensive taxonomy for multi-robot task allocation. The International Journal of Robotics Research 2013;32(12):1495–1512.
20. Gerkey, B.P., Matarić, M.J.. A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research 2004;23(9):939–954.
21. Wilensky, U., Evanston, I.. Netlogo: Center for connected learning and computer-based modeling. Northwestern University, Evanston, IL 1999;:49–52.
22. NetLogo extensions; 2016. URL: `http://ccl.northwestern.edu/netlogo/docs/extensions.html`; accessed: 2016-06-13.