Efficiency of Different Task Allocation Methods in Multi-Agent Systems

Krista Gurney and Janelle Miller

Introduction

Multi-Agent Systems and Distributed Problem Solving are the foundations of Distributed Artificial Intelligence. Multi-Agent Systems (MAS) are systems of multiple intelligent agents that interact with each other. Distributed Problem Solving (DPS) focuses specifically on how these agents work together to efficiently solve problems. To accomplish complex tasks, agents must negotiate, cooperate, and coordinate with each other [1]. This introduces the problem of task allocation, which focuses on how agents form coalitions or distribute tasks to other agents in order to solve the problem and maximize system utility [2].

DPS and MAS are complementary research agendas. The focus of MAS has been on how to get agents to interact meaningfully with other agents in a particular environment. DPS research has focused on achieving external properties likes robustness and efficient performance in dynamic environments from agents with established properties [3].

Task allocation is a popular topic of research within DPS. The goal of task allocation is to maximize the number of completed tasks among agents without conflict. A task is completed when the resources required by that task are available for use again. There may be systems where there are more tasks than agents, and as a result, agents need to schedule themselves to attempt each task in turn.

Centralized task allocation or distributed task allocation can be used to distribute tasks among agents. In a centralized approach, a central planner allocates tasks to individual or coalitions of agents. This system is not as robust since a single point of failure is inevitable. There is also a high communication overhead as the network grows larger. Lastly, if a task changes or a new one appears, the central planner must re-allocate tasks from scratch. In a distributed approach, a task can be received by any agent, and agents can communicate with each other to complete the task [1].

Task allocation in a multi-agent system can be used to model and solve a complex real-world problem, like work allocation in an organization. Agents can work individually and/or collaboratively, depending on the abilities of the agent and the properties of the task. Maximum system utility can be achieved when the tasks are properly allocated among agents [4].

Previous Work

In many real applications, all the information in a system can not be acquired by any agent. In these cases, a distributed task allocation method is more efficient than a centralized one.

Dahiya and Singhal [1] studied a negotiation based approach to distributed task allocation. Agents in a system will negotiate with other agents using a communication link to effectively allocate tasks. If an initiator agent is not capable of accomplishing a task individually then it will negotiate with other agents in the system. Agents will form coalitions by negotiating, and coalitions are chosen if they maximize the system utility.

In [5] a multi-agent negotiation model is used to distribute tasks in a priority based environment. The processed is modeled as a finite Markov Decision Process. Markov Decision Processes model decision making in situations where outcomes are partly random and partly under control of the agent [6]. This process was found to allocate tasks in an efficient way and possibly reduce the communication cost.

Agents with models of motivation and leadership have been used in distributed task allocation problems to effectively achieve an even distribution of agents to tasks. These models assist the agents with their coordination. This approach showed that the number of tasks discovered and the number of tasks allocated to agents increased. Also, the approach was successful in evenly distributing tasks among agents [7].

Work has been done by [4] to study distributed task allocation mechanisms for agents, either individually or in a formed coalition, with dependencies among tasks. The Shehory-Kraus's algorithm was used in their research to negotiate and form coalitions to maximize system utility. This approach was simulated using a disaster example. In this simulation, there are rescue agents: ambulance agents, police agents, and fire-fighter agents. The tasks are what would be expected in a real emergency, such as taking injured civilians to hospitals, putting out fires, and evacuating civilians. Each of the agents perform different tasks, but tasks can only be completed by an agent if it has the required capabilities. Dependencies between tasks also adds a layer of complexity to the task allocation process. When the number of tasks are predefined, dependencies between tasks increases the total number of tasks to more than the number of available agents. The priority and dependencies of tasks will also change over time. They concluded that when using this scenario the efficiency of the system was 73.25%.

Distributed task allocation mechanisms can outperform centralized mechanisms in certain environments; however, distributed task allocation in a network with a high growth rate can have a poor outcome. Multi-agent environments have been used to examine the effect of dynamic

networks on distributed problem solving. It has been found that "as the rate at which [network] links are added is increased, the final performance of the agents (the quality of the final solution) also becomes more variable, with performance apparently negatively affected in conditions involving high growth rates..." [8].

Work done by [9] compares distributed task allocation algorithms to a centralized approach in order to determine which method performs better. The task allocation problem is simulated in a search and rescue environment. A modified centralized algorithm based on particle swarm optimization was used as a benchmark against distributed algorithms. Particle swarm optimization algorithms work by populating a solution space with candidate solutions that communicate with each other to iteratively find the best solution [10]. This algorithm was compared to three distributed task allocation algorithms. The first algorithm was the consensus-based bundle algorithm, which is based on the market auction mechanism. The second and third algorithms were the Performance Impact (PI) and the PI with softmax, respectively. These algorithms were used to allocate time-critical tasks [11]. It was found that the centralized algorithm always outperformed the three different distributed task allocation algorithms in terms of time efficiency.

Market-based methods have become a popular approach for task allocation. These methods combine the advantages of centralized task allocation and distributed task allocation. Agents can create a plan based on local information, but once tasks are assigned they can trade tasks with other agents using currency. In [12] two auction clearing algorithms are used to bid on tasks (when trading), then the Hungarian Algorithm was used to solve the assignment problem and optimally match the task to the agent.

Original Work

Many methods for task allocation were discussed in the previous section. However, the success of these methods depend on the complexity of the environment, the agents, and the tasks that must be completed. Our contribution to existing research is to compare different methods of task allocation against each other with varying levels of complexity in agent and task properties.

Agents can have different characteristics like power (how many tasks it can take), what kind of tasks it can complete, and the utility it gets from certain tasks. Tasks can have different properties like size, priority, and what kind of agent is required to complete it. Tasks can also have subtasks or dependencies on other tasks, which makes the problem more complex. Tasks can be completed individually or cooperatively.

We want to look at how efficiently a set of tasks can be completed, depending on how the tasks are allocated. Different task allocation methods will have different efficiencies based on the characteristics of the agents and the properties of the tasks.

Experimental Design

Due to the difficulties of simulating a multi-agent system, a centralized approach was used for task allocation. If applied to a real life situation, this approach is similar to a boss delegating work to his employees (given the boss knows the capabilities of his workers). Because some tasks required a certain worker to complete it, workers are not permitted to share or trade work with each other. However, the method of allocating tasks, the characteristics of the agents, and the properties of the tasks are all varied to compare the efficiencies between different methods.

For our experiment we implemented different methods of centralized task allocation. Each type of task allocation had agent(s) complete 1 million tasks. A timer started at the beginning of each method returned the time when all the tasks were completed. One agent doing all tasks was used as a baseline that other task allocation methods were compared to. We simulated each type of task allocation as best as we could. To make things simple, work was simulated by incrementing a variable. Because this work is easy, each method was completed very fast with varying amounts of time.

The first task allocation method was multiple agents completing equal amounts of tasks. The second method was cascading tasks. In this method, a specific number of tasks were given to the first agent. While that agent was busy, tasks were given to other agents that were not busy. The third task allocation method was a variation of cascading method where agents could only handle a set amount of tasks, and agents would only work on tasks if they were not already busy working on tasks.

The fourth task allocation method worked with agents and tasks that each had a type, and agents could only complete tasks that were the same type as themselves. This method was an example of an emergency response situation: medical personnel can treat patients, police officers can direct traffic away from danger, and firefighters can remove obstacles and stop fires. Each person (agent) can take care of specific tasks that they know about and are capable of completing.

The fifth task allocation method applied to agents that gained utility when they completed tasks they preferred. However, the tasks could be completed by any agent so tasks were distributed randomly. The last allocation method was similar to the previous method but instead of randomly distributing tasks, it adopted a centralized approach. In this method, tasks were given

to the agents that preferred them. This increased the overall utility but also increased communication costs.

Results

The task allocation methods were run for agent(s) doing a total of 1 million tasks. The results are shown in Table 1. Table 2 summarizes the utility found for each of four agents in the last two task allocation methods.

Table 1: Summary of the task allocation methods and the time taken to complete 1 million tasks

Task Allocation Method	Time (s)
One Agent	0.28083
Three Agents Equally Divided	0.12293
Three Agents Cascading Tasks	0.34498
Three Agents Cascading Tasks with Varied Agent Ability	0.34980
Three Agents with Tasks and Types	0.44075
Four Agents with Utility and Task Preference, Randomly Allocated	0.98744
Four Agents with Utility and Task Preference, Centrally Allocated	0.96507

Table 2: Summary of the utility of agents for the task allocation methods involving utility

Utility	Randomly Allocated	Centrally Allocated
Agent A	62852	249912
Agent B	62580	249862
Agent C	62014	250309
Agent D	62586	249917

One agent doing all of the work was used as a baseline for our results. This method does not take the agent's utility, ability, or task types into account. Equally dividing the work among three agents had the best time efficiency for completing work but did not take utility, ability, or task type into account. The cascading allocation had worse results than the baseline.

The cascading allocation with varied agent ability took into account the power of each agent, or how many tasks they could complete. This method yielded similar results to the cascading allocation, which was also worse than the baseline.

Allocating tasks with types between three agents had the worst time efficiency thus far. This is intuitive since adding task types increases the complexity of the problem. Because this is a centralized method, the communication cost increased with complexity, and this decreased the time efficiency of allocating tasks.

Task allocation methods that took utility into account had the worst time efficiency of all the methods tested. This is because it is a centralized approach. In a distributed task allocation method, agents would negotiate with each other to maximize their individual utility and the overall system utility. Because the last two methods had similar time efficiencies, the utility of the agents was taken into account to determine the better method. The values in Table 2 show that the centralized approach had higher agent utility.

Applying these results to a real-life situation of distributing work to employees has a few implications. First, it takes time to allocate work. Assuming all employees worked at the same rate, the most efficient approach would be to equally split the work between employees. However, this does not take into account the actual capabilities of each employee. Some employees may have a limit on how much they can actually do, or what work they are qualified to do. Lastly, taking into account the happiness of employees is very expensive, especially if a large organization is considered. However, long-term, employees could defect to a different company if their happiness is not taken into account.

Future Work

Currently most of our task allocation implementations use a centralized task allocation method. Future work will involve a distributed task allocation approach where agents will negotiate with each other and form coalitions that maximize the utility of the agents and the system. Further study will be spent on different task allocation methods using game theory. We would also like to implement different methods with varying utility and varying number of agents. Further testing on larger systems would provide more insight into the best task allocation method.

Currently we have at most four agents, but in a real world problem the number of agents would be much greater.

References

[1] V. Singhal and D. Dahiya, "Distributed task allocation in dynamic multi-agent system," *International Conference on Computing, Communication & Automation*, Noida, 2015, pp. 643-648. Doi: 10.1109/CCAA.2015.7148452.

[2] G. Anders, C. Hinrichs, F. Siefert, P. Behrmann, W. Reif and M. Sonnenschein, "On the Influence of Inter-Agent Variation on Multi-Agent Algorithms Solving a Dynamic Task Allocation Problem under Uncertainty," *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems*, Lyon, 2012, pp. 29-38. doi: 10.1109/SASO.2012.16.

[3] Durfee, Edmund H., and Jeffrey S. Rosenschein. "Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples." Aaai.org, 7 June 1994, www.aaai.org/Papers/Workshops/1994/WS-94-02/WS94-02-004.pdf.

[4] G. Jezic, M. Kusek, I. Lovrek, R. Howlett, L. Jain. "Agent and Multi-Agent Systems: Technologies and Applications." *Proceedings of the 8th International Conference,* Chania, 2014, pp. 161-181. Doi: 10.1007/978-3-319-07650-8

[5] H. Luo, X. Hu and X. Hu, "Multi agent negotiation model for distributed task allocation," *2010 2nd IEEE International Conference on Information Management and Engineering*, Chengdu, 2010, pp. 54-57. doi: 10.1109/ICIME.2010.5477615

[6] Wikipedia contributors. "Markov decision process." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 6 Apr. 2019. Web. 22 Apr. 2019.

[7] M. K. D. Hardhienata, K. E. Merrick and V. Ugrinovskii, "Task allocation in multi-agent systems using models of motivation and leadership," *2012 IEEE Congress on Evolutionary Computation*, Brisbane, QLD, 2012, pp. 1-8. doi: 10.1109/CEC.2012.6256114

[8] Smart, Paul R, Huynh, Trung Dong, Braines, Dave and Shadbolt, Nigel, "Dynamic Networks and Distributed Problem-Solving", *Knowledge Systems for Coalition Operations (KSCO '10),* Canada, 2010.

[9] N. Geng, Q. Meng, D. Gong and P. W. H. Chung, "How Good are Distributed Allocation Algorithms for Solving Urban Search and Rescue Problems? A Comparative Study With Centralized Algorithms," in *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 478-485, Jan. 2019. doi: 10.1109/TASE.2018.2866395

[10] Wikipedia contributors. "Particle swarm optimization." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 18 Apr. 2019. Web. 22 Apr. 2019.

[11] A. Whitbrook, Q. Meng and P. W. H. Chung, "A novel distributed scheduling algorithm for time-critical multi-agent systems," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, 2015, pp. 6451-6458. Doi: 10.1109/IROS.2015.7354299

[12] B. Kalecı, O. Parlaktuna, M. Özkan and G. Kirlik, "Market-based task allocation by using assignment problem," *2010 IEEE International Conference on Systems, Man and Cybernetics*, Istanbul, 2010, pp. 135-141. doi: 10.1109/ICSMC.2010.5642222