CPSC 532a: Multiagent Systems
Agents: Different Meanings to Different People

Christopher L. Elliott
University of British Columbia
Department of Mechanical Engineering
cl3ellio@interchange.ubc.ca

December 9, 2005

Abstract

Agents are a central concept in Distributed Artificial Intelligence. What truly defines an agent and in what manner they are used is very broad. Distributed Problem Solving and Multiagent Systems are fields within Distributed Artificial Intelligence that use the agent concept quite extensively. This paper investigate the manner in which agents play a role in these two fields. An agent based approaches from both Distributed Problem Solving and Multiagent Systems to solving the task scheduling problem are examined and compared. A clear dinction is then made between the two fields.

1 Introduction

Research into the systems comprised of agents has been extensive since the early 1980s in the field of Distributed Artificial Intelligence (DAI). The term agent however can be interpreted very broadly. There is very little to the term that offers any descriptive bounds on its meaning.This project aims to review and discuss how agents help define DAI.

Even when one speaks about DAI the argument can be made that not all systems consisting of agents fall under its umbrella. As this project will show Multiagent Systems (MAS) do not exhibit the classical notion of distributed systems. Yet they are used as a frame work for coordination with distributed network of agents. Hence it is the belief of the author that MAS is a field of studying within DAI, and is treated in such a manner. The other field of DAI that is investigated will be Distributed Problem Solving (DPS), and the role that agents play within it.

The end result of the work in DAI, or any field for that matter, is to produce solutions to problems. Task scheduling is a classic optimization problem, DAI methods for solving this have been developed with both MAS and DPS motivated algorithms. Both approaches will be addressed and the distinction between the two will become apparent.

2 Distributed Artificial Intelligences

The field commonly referred to when discussing MAS is DAI. This field of study has grown since the early 1980s, and looks into solving problems in a distributed manner. Those who have reviewed the field have broken the field into two disciplines, DPS and MAS.

A key abstraction in MAS and DPS is the agent, and its functionality. The issue of DPS solution elements being a proper use of the term agent will be discussed later on. First it is important to explore the agent landscape and the fields of DPS and MAS.

2.1 Agents

There have been many attempts to define what agents are or what constitutes 'agenthood'. A review of several definitions, including there own, is given by Fanklin and Graesser [3]. They define an autonomous agent as 'a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future'.

This definition is broad and still leaves the functionality of the agent to be defined. The broad definition is due to the difficulty in formalizing what an agent is. Jennings et al. [4] view an agent as 'a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives.' This definition provides more clarity by defining two distinct concepts: flexible autonomy and the agent's awareness of an environment. Flexible is defined as a system that is responsive, pro-active and social [11].

In order to look at the application of agents within DAI it is important not only to define what agents are but their functionality as well. When the discussion is expanded to functionality, a typology can be considered. Nwana [6] identifies three fundamental attributes of agents: cooperation, learning and autonomy. Finally identifying several different types of agents:

1 - *Collaborative agents*: these agents are autonomous and cooperative with other agents. Where a group of collaborative are capable of functions that are not within the individual capabilities.

2 - *Interface agents*: these agents typify a great deal of single-agent systems. Since they do not emphasize cooperation, they are not inherently designed to work with other agents.

3 - *Mobile agents*: mobile agents are those whose process is capable of moving from one execution point to another.

4 - *Information agents*: these agents manage, manipulate and collect information.

5 - *Reactive agents*: agents respond in a stimulus-response manner and do not posses an internal model of the environment.

Agents that merge these types together are referred to as:

- Hybrid agents: an agent that exhibits a combination of the previously mentioned capabilities.

- Smart agents: agents that combine all five capabilities to varying degrees.

The typology offered by Nwana does a very good job of identifying the different functional purposes one would want to use an agent. There are some drawbacks, of the five capabilities, mobility is not sufficient in describing a type, it must be coupled with another capability. Otherwise the 'mobile' agent is a process that moves from one location to another with no specific functionality, this does not meet the criteria of agenthood. There is no mention of flexible autonomy, or what methodology the agent uses to base its mobility.

2.2 Distributed Problem Solving

DPS is the study of loosely connected problem solvers who work together on solving a problem that is beyond the capability of an individual problem solver [2]. Shoham and Leyton-Brown [8] characterize four families of techniques:

- Distributed constraint satisfaction algorithms.

- Distributed dynamic programming.

- Auction-like optimization algorithms.

- Distributed solutions to Markov Decision Problems.

In DPS, agents are used as problem solvers, in most cases, to cooperate with one another in finding a solution to a problem. The use of cooperation to solve these problems has several goals in mind [2]:

1. Increase the rate that task are completed through parallelism.

2. Sharing resources allow for the scope of achievable tasks to be increased.

3. Through duplication and/or using different methods to perform task, likelihood of task completion is increased.

4. The interference between task is decreased by avoiding harmful interactions.

The agents described in Section 2.1 can be used to build a framework to solve problems in a distributed manner. One can see the large role that collaborative agents can play, since these agents are built with cooperation in mind, they lend themselves to this application.

One of the earliest applications of of DSP was the Contract-Net Protocol [1]. This protocol dealt with distributing task dynamically through a network, where agent took on the dual role of manager and contractor. Its intended purpose was as a negotiation technique, but when one looks at it carefully in is a method of task allocation [4]. Though the protocol was created over twenty years ago, it is in principle still used today, with many researchers during the 1990s building upon the core idea.

2.3 Multiagent Systems

Defining a multiagent system is a difficult thing to do. Many researchers will group DPS within MAS, Sycara [10] actually cites the definition given by Durfee and Lesser [2] for DPS, but claims it as a definition for MAS. Even within the field of DAI it is unclear where the line between DSP and MAS exist, some believe that there is none at all.However the distinction will become clearer.

Within the definition of DPS there is no mention of how the agents coordinate with one another. Stone and Veloso [9] define MAS as providing the principles for construction of complex systems involving multiple agents and mechanisms for coordinating autonomous agents' behaviour. This definition addresses the problem of systems where we have multiple agents but are not aware of their goals. It would be desirable if we could create systems that will dictate the way the agents behave.

In systems where agents have competing goals, each agent will have a utility function that it wishes to maximize. Mechanism design aims to study how the privately held preferences of the agents can be amassed towards a 'social choice'. This field blends concept from various fields including economics, game theory and social choice.

In the past 20 years this field has gained in interest, but it is often grouped with what others deem as MAS. Some authors [4, 10] tend to describe these systems as economic-based approaches. While the concepts do originate from economics, they are broadly used in systems where resources are scarce and agents wish to acquire those resources in order to maximize their own utility.

3 Task Allocation

The Task Allocation Problem is a classical computer science problem. It states that there a m task that need to be allocated to n agents. Each agent is capable of completing each task in a minimum amount of time t_{ij}, where i denotes the agent and j denotes the task. The goal is to minimize the completion time of the last assignment.

This problem has been approached in many different manners. Here we will review two solutions to the problem, one that uses a DPS approach and another that uses a MAS approach.

3.1 DPS Task Allocation

The algorithm presented by Shehory and Kraus [7] uses coalition formation among autonomous agents to allocate the task to each agent. There are two main stages to this algorithm [7]:

1. The coalition values must be calculated. The coalition value is the joint utility that members of the coalition can reach by cooperating on satisfying a specific task.

2. An iterative greedy process, where the agents determine what coalitions they prefer to form.

The authors have added a third step that modifies the algorithm for a MAS, where the utility amassed must be disbursed to the agents. This modification will not be considered since we are interested in a DPS solution. Let us now look at the two steps in greater detail [7]:

Step 1: Each agent must calculate all possible coalitions that it can be a member of, up to a specific size k. It the goes about communicating with each agent on that list, inquiring about their

capabilities, likewise inform them of your capabilities, allowing both to calculate the coalition values. Once all agents on the list have been contacted, calculate the expected outcome of the tasks that can be performed by the coalition. Choose the maximal expected outcome to be the coalition value.

Step 2: The preferred coalitions are chosen and the configuration is gradually achieved. This is done by iteratively with each agent announcing what coalition what coalition on their list has the lowest coalition cost size ratio. The lowest value announced will be chosen, and all agents from that coalition are deleted. The process is then repeated till all agents are deleted, or no task remain to be allocated, or no possible coalition can fulfill the remaining tasks.

This algorithm is decentralized, meaning no central unit is needed to compute what task is allocated to which agents. Its computational complexity is $O(n^{k-1}m)$ and has a communication complexity of $O(nm)$ for each agent [7].

This algorithm displays the collaborative characteristic of the agents. The agents are working towards maximizing the value of the task allocation in a communal sense. Removing the element of personal utility from the agents, allows this problem to be solved in this fashion. Since we know all agents are working for the same team, there is no need to put into place a central mechanism to control their behaviour. Unfortunately the only manner to ensure that all agents are working towards the same goal, would be to design and construct the agents.A fundamental difference between the two approaches now begins to emerge.

3.2 MAS Task Allocation

Here a mechanism is formulated Nisan and Ronen [5], that optimizes the behaviour of selfish agents. They present a VCG mechanism that provides an approximate solution to the task scheduling problem. Each agent i's valuation is the negative of the sum of all t_{ij}. The MinWork mechanism used is described as follows [5]:

- Allocation: each task is allocated to the agent who is capable of doing it in the shorted amount of time.

- Payment: for each task allocated to an agent, it will pay the time of the second best agent at each specific task.

One can see by inspection that this mechanism is part of the VCG family of mechanism studying in the course. Hence the mechanism is incentive compatible, but as the authors point out only provides an upper bound solution for the problem, claiming that the MinWork is a n-approximation to the task scheduling problem [5]. A randomized mechanism that provides a tighter approximation bound is then later presented by the authors.

This approach though centralized, overcomes the main problem from the DPS solution. We no longer need to be guaranteed what the goals of the individual agents are, a mechanism has been implemented that will put in place a system where only agents that are truthful will benefit. Allowing for new unknown agents to be introduced into the system, without fear of harm.

4 Conclusion

A designer of DPS systems must have access to all the elements of the system. They must know all the characteristics of the agents involved in their solution concept. Without such knowledge it is impossible for the designer to guarantee that there system will have the desired result.

Those who design MAS systems work without this restriction. Their goal is to extract a desired behaviour from a network of interacting agents. This quality is what makes MAS a more powerful concept than DPS, there arises very few instances where one can have complete control over all the resources one needs.

Not to discount the work done in DPS, there is a great body of work that has been produced in the field. MAS however removes a great deal of the need to typecast agents, and places the focus on the systems behaviour as a whole.

References

[1] Davis, R., Smith, R.G., Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence*, Vol.20, No.1, 1983.

[2] Durfee, E.H., Lesser, V.R., Corkill, D.D., Trends in Cooperative Distributed Problem Solving. *IEEE Transactions on Knowledge and Data Engineering*, Vol.1, No. 1, March 1989.

[3] Franklin, S., Graesser, A., Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages*, Springer-Verlag, 1996.

[4] Jennings, N.R., Sycara, K., Wooldridge, M., A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, Vol. 1, No. 1, March, 1998.

[5] Nisan, N., Ronen, A., Algorithmic Mechanism Design. *Proceedings of the 31st Annual Symposium on Theory of Computation*, ACM, pg.129-140, 1999.

[6] Nwana, Hyacinth S., Software Agents: An Overview. textitKnowledge Engineering Review, Vol. 11, No. 3, October/November 1996.

[7] Shehory, O., Kraus, S., Task Allocation via Coalition Formation among Autonomous Agents. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufman, pg.655-661, 1995.

[8] Shoham, Y., Leyton-Brown, K., Multi Agent Systems. Unpublished, 2005.

[9] Stone, P., Veloso, M., Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robotics*, Vol. 8, No. 3, July 2000.

[10] Sycara, K.P., Multiagent Systems. *Artificial Intelligence*, Vol. 19, No. 2, 1998.

[11] Wooldridge, M., Jennings, N.R., Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, Vol. 10, No. 2, 1995.