Multi-Agent Systems

Gal A. Kaminka

May 20, 2004

1 Overview and Key Themes

Agents are computational entities that sense and act, and decide on their actions in accordance with some tasks or goals. There is no consensus among researchers on an exact definition of agents. However, definitions generally agree that agents are: (i) *situated*, in that their sensing and acting occur within the same environment, (ii) *persistent*, in that their existence and operation are continuous over a non-trivial amount of time (with respect to the environment and their task), and (iii) *autonomous*, in that their control process (i.e., their decision making) cannot be tweaked by external means, other than through their sensors. Agents are *goal-oriented* if they act in accordance with their tasks and goals.

Multi-agent systems (MASs) is the scientific field that studies the behavior of multiple agents, when interacting with each other and with their environment, in various scales and forms of organization. Researchers in this field (1) build theories that predict such behavior in natural and synthetic MASs; (2) discover techniques that guide agents in their social and rational interactions; and (3) create methods for constructing MAS instances that address specific application needs, including but not limited to simulation, testing, and software engineering. The field is influenced by economics, sociology and organization science, philosophy, natural language processing, biology, and artificial intelligence.

The field of multi-agent systems is concerned with *decentralized processes* (*distributed systems*), as each individual agent in the system has its own perception (via sensors), control, and actuation (via actions). Thus, agents may differ in their perception (for instance, due to difference in physical locations), in their control (for instance, different expertise), and in their actuation (for instance, due to having different potential actions). Where such differences are significant to the operation of the MAS, the agents are called *heterogeneous*. In many cases, however, these differences are only significant in that they enable parallelism, and the agents are called *homogeneous*. For example, if any one of a number of agents can carry out a task (all with the same quality), the agents are considered homogeneous—since increasing the number of similar tasks will allow multiple agents to tackle these tasks in parallel.

Whether heterogeneous or homogeneous, agents in MASs interact with each other in some form to achieve their individual goals, since these goals may

depend on one another. Where such dependencies exist, the agents will need to coordinate with one another. Broadly categorized, there are three types of coordination: Agents may *compete* for resources, or to achieve conflicting goals; they may *cooperate* to achieve compatible/complimentary goals; and they may *collaborate* to achieve common goals. Note that while collaboration is inherently two-sided, cooperation and competition can also be one-sided. For example, a stealthy predator may coordinate its movements with that of its prey, but the prey—lacking knowledge of the predator—does not coordinate with the predator. An agent may also be malicious, if its goals involve preventing others from pursuing their goals.

A key challenge in any MAS deals with allowing the agents to interact effectively, by expanding their sensing and acting capabilities to handle social interactions. A single agent, alone in its environment, must be able to sense its environment, reason about it, and act on it, to be effective. Similarly, to be effective in interacting with other agents, an agent in a multi-agent system must be able to sense others, reason about them, and act on them (e.g., through manipulation, persuasion, argumentation, negotiations, command, etc.). Such sensing and acting can be done by manipulating common features in the environment, or by specialized sensors/actuators (e.g., radio devices, Internet). There is a distinction between *communications* (which involve two-way interactions), and *observations* (in which one of the agents does not know that it is being sensed and/or acted upon).

Multi-agent systems involve computational limitations both at the level of individual agents (e.g., memory, computation power, sensor uncertainty) and in communications/observations (e.g., in terms of bandwidth, latency, reliability, security against tampering or eavesdropping, preservation of order of messages, etc.). All of these limitations play a critical role in how agents interact with each other. For instance, a two-agent system, composed of a human user and a software agent, has limited bandwidth; the software cannot continuously bombard the user with information or queries. Thus the software agent must carefully control the content, timing, and form of interaction with the user. Similarly, the user must consider his or her interactions with the software agent, so as to cause it to achieve the required goals.

To ease the computational load on agents (in terms of their choice of interactions), MASs often employ *organizations* that constrain the type of interactions that an agent may employ. Such constraints are called *norms*, and they guide the social behavior of agents, by reducing the number of alternative interactions agents may take. Norms may dictate interaction protocols to be followed when agents interact with each other. Also, organizations may have *roles* within them, that constrain the individual behavior of agents fulfilling them. For example, a customer service telephone operator for a large company may be guided in her interactions with an angry client by norms (e.g., saying "Hello" and dictating politeness towards the client), and also by role (limiting the range of actions the operator may take to noting down the complaint and compensating the client by no more than a fixed amount). Agents in complex MASs may face conflicts within their roles, norms, or both.

Many, if not most, MASs have within them multiple types of organizations, sometimes in nested forms. For instance, a game of soccer has a nested organizational structure. The organization has two teams that cooperate with each other in playing a game of soccer according to standard rules (norms). To play the game, the teams compete with each other to score goals. Each team is composed of players that collaborate with each other to achieve their common goal. To do this, players may organize themselves into sub-teams as necessary, and even compete within the team in order to improve its effectiveness (e.g., a few teammates may run to meet an incoming ball, such that the fastest of them will be able to stop it). Players in soccer also often have roles: For example, a goalie stays behind to protect its own team's net, while the strikers' role is to try to kick the ball into the opponents' net.

Organizations differ not only in their structure and coordination types, but also in *scale* and *openness*. The scale of an organization is defined by the number of agents that participate in it. Empirically, different types of interactions occur in small *groups* (up to a hundred agents) then in *swarms* (thousands to millions of agents). *Closed* organizations maintain their agent membership throughout their lifetime. *Open* organizations allow agents to join and leave dynamically, and as a result cannot typically dictate the internal controls of the agents. Instead, participation in open organizations is typically achieved by maintaining interaction standards, for example standard communication languages and communication protocols. Often, open organizations rely on *middle agents* to provide services such as *match-making* (connecting agents requiring a service to agents providing it), *brokering*, *certification*, etc.

Indeed, organizations may dynamically change over time, not only in membership, but also in the roles assigned to members, and also in tasks or goals. The problem of forming a new organization by choosing agent members such that their interactions and roles best carry out (cover) a set of tasks is called the *coalition formation problem*. The problem of assigning (and re-assigning) tasks to given agent members such as to maximize overall organizational effectiveness is called the *task allocation problem*. While organizations are mostly concerned with carrying tasks for specific goals (whether competitive or not), their disbandment does not simply occur with goal achievement. Some organizations may disband based on environmental conditions (for instance, a soccer game ends with time). In others, norms may dictate that agents remain members of an organization even once the organizational goal is achieved.

We can now restate the goals of the field of multi-agent systems in the terms we introduced:

- Build theories that predict the interactions and organizations that allow agents to carry out given tasks, in given environments, and given their computational limitations.
- Discover techniques that allow agents to overcome computational and system limitations in order to effectively coordinate/interact with each other.
- Create methods for forming organizations that address specific application

needs.

Often, advances towards one of these goals lead to further advances in others. For instance, techniques that proscribe effective teamwork behavior can lead to predictions as to the interactions observed in well-coordinated teams, and vice-versa. All of these goals are pursued within multi-agent systems using a variety of approaches, described below.

2 Approaches in Multi-Agent Systems

Historically, MAS has evolved from earlier attempts with the artificial intelligence community to consider questions that arise out of the study of multiple problem-solving agents that work in parallel. These earlier attempts are often referred to as *DAI* (Distributed Artificial Intelligence) or *DPS* (Distributed Problem Solving). MAS has also borrowed from social sciences, including sociology, economics, and organizational science. These different backgrounds lead to different approaches within the field.

2.1 Distributed Problem Solving

Distributed Problem Solving (DPS) deals with MASs in which agents cooperate or collaborate with each other to solve a common problem (the results of their problem solving may be centralized or distributed to the participants). This type of MASs arise naturally in many industrial and computational problem settings, where a large-scale problem may benefit from a significant speed-up if it is decomposed into many sub-problems that are solved in parallel. DPS also matches well with problems where agents are heterogeneous in their capabilities or the resources they have, and can achieve their (common) goals by collaborating with each other. For instance, if different agents have access to different information, or have different computational power, they may solve problems together, that none of them could tackle individually.

There are several distinct arch-types of DPS, which differ in the basis for the decomposition, and in the centralization of the solution(s). In one type, the agents focus on different sub-problems, but they all have access to the same inputs in principle. Thus the main purpose of decomposition is to speed up problem solving. The decomposition itself can be challenging, as alternative decompositions are often possible. Moreover, in more complex settings, the heterogeneous capabilities of the agents are taken into account in the decomposition, such that sub-problems are allocated to the agents best suited to handle them. In such allocation, an important objective is *load-balancing*, which distributes resource usage as fairly as possible. A good example of this type of DPS is multi-agent computation; different parts of a complex computation are handed off to different agents, and the results are combined once the agents are done, each with its own process.

In a different type of DPS, agents focus on tackling the same problem, but using different expertise or knowledge. In such settings, the solution is often

formed through iterative process of agents computing partial results, which are passed to their peers to be refined (and assist the other agents), and then posted back. A global solution is constructed out of these iterations over partial results. A good example of this DPS type is distributed management of cellular phone base-stations. Each base-station (agent) can only monitor and communicate with phones in some limited-range local area (cell), but must adjust its frequency and resource usage to match that of other stations, whose cells overlap. Load balancing here involves making sure no single cell is carrying out too much of the traffic.

Complex applications often involve a combination of these two types of DPS, and no single technique is known yet that addresses all of the challenges involved in DPS. Moreover, challenges are raised not only during the planning phase, but during run-time, where due to the nature of dynamic environments, the decomposition of the task or results must change dynamically, and the agents must coordinate their run-time responses. Some techniques have repeatedly been demonstrated to solve important subsets of such challenges. These include blackboard architectures (in which agents exchange partial results by using shared memory), contract-net protocols (which allow agents to consider their task load when negotiating over allocation of tasks), and distributed constraint satisfaction techniques (which determine globally-coherent solutions).

An important instance of DPS deals with *collaboration*—also called *teamwork*—in which agents are not only committed to a shared goal, but also to an agreed-upon way of achieving it, and to providing mutual support and assistance to their teammates. Thus for instance, team members cannot terminate their activities within the team without gaining their teammates' agreement, and they are committed to taking over tasks from teammates, proactively providing relevant information to teammates, etc.

2.2 Rational and Economic Approaches

While DPS techniques and models assume that agents have banded together to solve a common problem, *distributed rational* approaches stemming from economics and game-theory make no such assumption. Instead, agents are assumed to be *rational* and *self-interested*, in the sense that they seek to maximize (by their chosen actions) some individual utility function with no concern for the global good. Such models fit naturally with systems in which independent businesses or individuals interact. Key questions in such settings involve the prediction of the action sequence (called *strategy*) of each agent, and the design of the protocol (*mechanism*) which governs their interaction, such that the MAS displays required characteristics.

There are several alternative criteria for evaluating an MAS based on self-interested agents. First, we may ask as to the *social welfare* of the system—the sum of its agents' utility values. We may also want the system to be *stable*, in that agents are motivated out of their own self-interest to choose the desired strategy. For instance, if each agent, given the strategies of its peers, cannot improve its reward by selecting a different strategy, the system is said to be

in a *"Nash equilibrium"*. Ideally, we would prefer a mechanism that maximizes social-welfare and is also stable. However, these two criteria can sometimes be at odds, for instance in the Prisoner's Dilemma game. Other criteria exist, such as *manipulability*, which considers the ability of a single or a coordinated group of participants to bias the outcome of a mechanism in their favor.

A wide variety of mechanisms exist. However, some key types are: (i) *social choice*—also known as *voting*—mechanisms, in which all agents provide input as to a preferred outcome, and all agents are committed by the output of the mechanism; (ii) *auctions*, in which all agents provide input, but the outcome only commits a subset of the agents, auctioneer(s) and bidder(s); (iii) *markets*, which optimize resource production and consumption by allowing consumers and producers to negotiate over prices; (iv) *contract nets*, which facilitate distributed task allocation. Each one of these main types represents a large number of variations, which exhibit different characteristics.

3 Multi-Agent Systems and HCI: Key Areas of Overlap

Multi-Agent Systems and HCI have overlapping areas of research, which have resulted in a number of productive investigations, and offer still many opportunities for future technologies. These overlapping areas can be generally categorized based on the cardinality of the interaction: (a) one-to-one interactions (a human user interacting with a single computer as a two-agent multi-agent system); (b) one-to-many interactions between a human user and a set of computational agents; (c) many-to-many interactions, where a mixed group of humans and computational agents interact with each other; and (d) many-to-one interactions, where a single computational agent interacts with multiple human users.

In terms of one-to-one interactions, teamwork theories which have been developed in multi-agent systems have been successfully used to improve user-interface mechanisms. By modeling the two-agent system as a team, and accounting for the different capabilities of the agents (the user and the computer), improved interactions have resulted in which the computer can take a more proactive collaborative role. Also, modeling the interaction as collaboration facilitated improved communication from the computer, and reduce the load on the user. The techniques have also been used in two-agent human-robot interactions.

An expanding area of research deals with one-to-many interactions, in particular in providing methods for a single human user (often, the operator) to monitor, visualize, and command a group of agents that work on its behalf. Methods for allowing command of groups vary from providing commands to a centralized agent (which distributes them to its peers), to sequential one-on-one interactions between the operator and a single agent, as needed. In general, one-to-many interactions require significant underlying autonomy by the agent

group members. A key challenge lies in monitoring the group, as agents are physically and logically distributed, and thus mostly unobservable to the user. To gather the monitoring information, agents may be required to communicate their activities to the user, assuming reliable and cheap communications. Alternatively, a technique called *overhearing* allows the user to monitor the agents by listening in to their routine conversations.

Finally, applications have recently emerged for many-to-many interactions, in which agent groups consist of multiple humans and multiple computational agents. For example, future search-and-rescue operations will include software and robotic agents, which will provide logistics and physical labor services to human rescue workers, to limit danger to humans and improve rescue efforts. In addition to task and resource allocation issues here, challenges also include using agents to represent their human users in interacting with other humans (e.g., as *avatars*) or with other agents. In such cases, agents must decide, through techniques of *adjustable autonomy*, on the scope of their authority to act on behalf of their user(s) without asking her for guidance.

References

[1] J. A. Adams. *Human Management of a Hierarchical System for the Control of Multiple Mobile Robots.* PhD thesis, University of Pennsylvania, 1995.

[2] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 35, 1991.

[3] Y. Elmaliach. Single operator control of coordinated robot teams. Master's thesis, Bar Ilan University, 2004.

[4] B. J. Grosz and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.

[5] B. J. Grosz and S. Kraus. The evolution of SharedPlans. In M. Wooldridge and A. Rao, editors, *Foundations and Theories of Rational Agency*, pages 227–262. 1999.

[6] B. J. Grosz and C. L. Sidner. Plans for discourse. In P. R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 417–445. MIT Press, Cambridge, MA, 1990.

[7] N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[8] G. A. Kaminka, D. V. Pynadath, and M. Tambe. Monitoring teams by overhearing: A multi-agent plan recognition approach. *Journal of Artificial Intelligence Research*, 17, 2002.

[9] N. Lesh, C. Rich, and C. L. Sidner. Using plan recognition in human-computer collaboration. In *Proceedings of the Seventh International Conference on User Modelling (UM-99)*, Banff, Canada, 1999.

[10] K. L. Myers and D. N. Morely. Human directability of agents. In *Proceedings of the First International Conference on Knowledge Capture, K-CAP 2001*, Canada, 2001.

[11] C. Rich and C. L. Sidner. COLLAGEN: When agents collaborate with people. In W. L. Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents (Agents-97)*, pages 284–291, Marina del Rey, CA, 1997. ACM Press.

[12] P. Scerri, L. Johnson, D. Pynadath, P. Rosenbloom, M. Si, N. Schurr, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, 2003.

[13] P. Scerri, D. V. Pynadath, and M. Tambe. Towards adjustable autonomy for the real-world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

[14] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[15] A. D. Tews, M. J. Mataric, and G. S. sukhatme. A scalable approach to human-robot interaction. In *Proceedings 2003 IEEE International Conference on Robotics and Automation*, Taiwan, May 2003.

[16] G. Weiss, editor. *Multiagent Systems: A modern approach to distributed artifical intelligence.* the MIT Press, 2000.

[17] H. A. Yanco, J. L. Drury, and J. Scholtz. Beyond usability evaluation: Analysis of human-robot interaction at a major robotics competition. *Human-Computer Interaction*, 19(1–2), 2004.