"MULTI-AGENT CO-ORDINATION AND CONTROL USING STUDY OF SOFTWARE AGENT"

Abha Jaiswal
(Research Scholar)
UPRTOU, Allahabad, India

Ravi Prakash Jaiswal
(Senior Technical Assistant)
MGKVP, Varanasi, India

Abstract— Multi-agent co-ordination is a fundamental problem in multi agent systems that has acquired a variety of meanings in the relative literature. In this paper we focus on a setting where multiple agents with complementary capabilities cooperate in order to control non-conflicting plans that achieve their respective goals. We study two situations. In the first, the agents are able to achieve their sub goals by themselves, but they need to find a coordinated course of action that avoids harmful interactions. In the second situation, some agents may ask the assistance of others in order to achieve their goals. We formalize the two problems and present algorithms for their solution. These algorithms are based on coordination and control between agents which is used by the agents to control their behavior, but also to find controls that are consistent with those of the other agents. It can be seen as an attempt to establish a stronger link between Interethnic co operation and multi-agent Systems.

Keywords-agent based system; intelligent system; Decentralization, Agent co operation;

1 Introduction

Multi-agent coordination and cooperation are important issues in the multi-agent field. Several works have been proposed, covering different aspects of the problem of coordinating the plans of several agents operating in the same environment. These include scenarios where plan generation is distributed, where planning is centralized and plan execution is distributed, or where both planning and execution are distributed. However, only few works (e.g. [2], [5]) tackle aspects of the cooperation problem in the context of multi-agent planning.

In this paper we study both problems of coordination and cooperation of multiple agents. In particular, we consider two scenarios that we believe cover an important number of applications. In the first, agents have individual (private) goals that they can achieve by themselves. The agents can generate and execute their plans independently. However, as they operate in the same environment, conflicts may arise. Therefore, they need to coordinate their course of action in order to avoid harmful interactions. We will call this situation multi-agent *coordination*. In the second scenario, which is a special case of multi-agent *cooperation*, an agent can ask some other agent to establish preconditions of actions that appear in his plan. We will call this situation multi-agent *assistance*.

In a multi-agent coordination scenario, a number of agents need to generate individual plans that achieve their respective goals and are not in conflict with each other. We restrict ourselves to the case of two agents, α and β, and study a multi-agent coordination scenario that is defined by two main characteristics. The first is that each agent is able to achieve his goals by himself. This distinguishes coordination from cooperation. The second characteristic is that plan length is the criterion for evaluating the quality of both the individual and the joint plans, with preference given to the joint plan length. Therefore, agents seek to minimize the length of the joint plan, even in the case where this leads to non-optimal individual plans.

It applies to a wide range of application domains in which decision-making must be performed by multiple collaborating agents such as information gathering, distributed sensing, coordination of multiple robots, as well as the operation of complex human organizations. While substantial progress has been made in planning and control of single agents using, a similar formal treatment of multi-agent systems has been lacking. Existing techniques tend to avoid a central issue: agents typically have different information about the overall system and they cannot share all this information all the time. Sharing information has a cost that must be factored into the overall decision process. Three approaches to communication are studied based on a cost/benefit analysis of the amount of communication, search in policy space, and transformations of the more tractable centralized policies into decentralized policies. The resulting techniques are evaluated in the context of several realistic applications. This paper facilitates a better understanding of the strengths and limitations of existing approaches to coordination and offers new approaches based on more formal underpinnings.

I.

A multi-agent system (MAS) consists of multiple independent agents that interact in a domain. Each agent is a decision maker that is situated in the environment and acts autonomously, based on its own observations and domain knowledge, to accomplish a certain goal. A multi-agent system design can be beneficial in many AI domains, particularly when a system is composed of multiple entities that are distributed functionally or spatially. Examples include multiple mobile robots (such as space exploration rovers) or sensor networks (such as weather tracking radars). Collaboration enables the different agents to work more efficiently and to complete activities they are not able to accomplish individually. Even in domains in which agents can be centrally controlled, a MAS can improve performance, robustness and scalability by selecting actions in parallel. In principle, the agents in a MAS can have different, even conflicting, goals. We are interested in fully-cooperative
MAS, in which all the agents share a common goal. In a cooperative setting, each agent selects actions individually, but it is the resulting joint action that produces the outcome. Coordination is therefore a key aspect in such systems. The goal of coordination is to ensure that the individual decisions of the agents result in (near-) optimal decisions for the group as a whole. This is extremely challenging especially A multi-agent system (MAS) consists of multiple independent agents that interact in a domain. Each agent is a decision maker that is situated in the environment and acts autonomously, based on its own observations and domain knowledge, to accomplish a certain goal. A multi-agent system design can be beneficial in many AI domains, particularly when a system is composed of multiple entities that are distributed functionally or spatially. Examples include multiple mobile robots (such as space exploration rovers) or sensor networks (such as weather tracking radars). Collaboration enables the different agents to work more efficiently and to complete activities they are not able to accomplish individually. Even in domains in which agents can be centrally controlled, a MAS can improve performance, robustness and scalability by selecting actions in parallel. In principle, the agents in a MAS can have different, even conflicting, goals. We are interested in fully-cooperative
MAS, in which all the agents share a common goal. In a cooperative setting, each agent selects actions individually, but it is the resulting joint action that produces the outcome. Coordination is therefore a key aspect in such systems. The goal of coordination is to ensure that the individual decisions of the agents result in (near-)optimal decisions for the group as a whole. This is extremely challenging especially

A **software agent** is a computer program that acts for a user or other program in a relationship of agency, which derives from the Latin *agere* (to do): an agreement to act on one's behalf. Such "action on behalf of" implies the authority to decide which, if any, action is appropriate.[1][2]

Related and derived concepts include *intelligent agents* (in particular exhibiting some aspect of intelligence, such s learning and reasoning), *autonomous agents* (capable of modifying the way in which they achieve their objectives), *distributed* agents (being executed on physically distinct computers), *multi-agent systems* (distributed agents that do not have the capabilities to achieve an objective alone and thus must communicate), and *mobile* agents (agents that can relocate their execution onto different processors).

The basic attributes of a software agent are that agents

- are not strictly invoked for a task, but activate themselves,
- may reside in wait status on a host, perceiving context,
- may get to run status on a host upon starting conditions,
- do not require interaction of user,
- may invoke other tasks including communication.

The term "agent" describes a software abstraction, an idea, or a concept, similar to OOP terms such as methods, functions, and objects.[*citation needed*]The concept of an agent provides a convenient and powerful way to describe a complex software entity that is capable of acting with a certain degree of autonomy in order to accomplish tasks on behalf of its host. But unlike objects, which are defined in terms of *methods* and *attributes*, an agent is defined in terms of its behavior [3]

- *persistence* (code is not executed on demand but runs continuously and decides for itself when it should perform some activity)
- *autonomy* (agents have capabilities of task selection, prioritization, goal-directed behaviour, decision-making without human intervention)
- *social ability* (agents are able to engage other components through some sort of communication and coordination, they may collaborate on a task)
- *Reactivity* (agents perceive the context in which they operate and react to it appropriately).

A. **impact of software agents**

Software agents may offer various benefits to their end users by automating complex or repetitive tasks.[5] However, there are organizational and cultural impacts of this technology that need to be considered prior to implementing software agents.

Organizational impact

Organizational impacts include the transformation of the entire electronic commerce sector, operational encumbrance, and security overload. Software agents are able to quickly search the Internet, identify the best offers available online, and present this information to the end users in aggregate form. Therefore, users may not need to manually browse various websites of individual merchants; they are able to locate the best deal in a matter of seconds. At the same time, this increases price-based competition and transforms the entire electronic commerce sector into a uniform perfect competition market. The implementation of agents also requires additional resources from the companies, places an extra burden on their networks, and requires new security process.

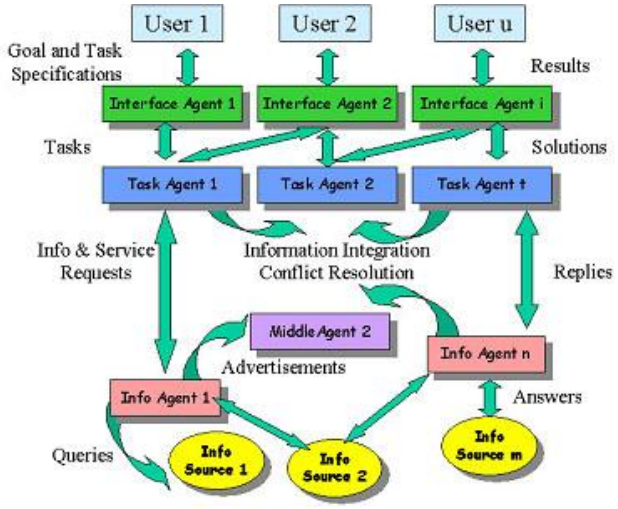
Work contentment and job satisfaction impact

People like to perform easy tasks providing the sensation of success unless the repetition of the simple tasking is affecting the overall output. In general implementing software agents to perform administrational requirements provides a substantial increase in work contentment, as administering the own work does never please the worker.[6]

The effort freed up serves for higher degree of engagement in the substantial tasks of individual work. Hence, software agents may provide the basics to implement self-controlled work, relieved from hierarchical controls and interference.[6] Such conditions may be secured by application of software agents for required formal support.

Cultural impact

The cultural effects of the implementation of software agents include trust affliction, skills erosion, privacy attrition and social detachment. Some users may not feel entirely comfortable fully delegating important tasks to software applications. Those who start relying solely on intelligent agents may lose important skills, for example, relating to information literacy. In order to act on a user's behalf, a software agent needs to have a complete understanding of a user's profile, including his/her personal preferences.[5] This, in turn, may lead to unpredictable privacy issues. When users start relying on their software agents more, especially for communication activities, they may lose contact with other human users and look at the word with the eyes of their agents. It is these consequences that agent researchers and users need to consider dealing with intelligent agent technologies.[7]



Agent systems are used to model real-world systems with concurrency or parallel processing.

- Agent Machinery - Engines of various kinds, which support the varying degrees of intelligence
- Agent Content - Data employed by the machinery in Reasoning and Learning
- Agent Access - Methods to enable the machinery to perceive content and perform actions as outcomes of Reasoning
- Agent Security - Concerns related to distributed computing, augmented by a few special concerns related to agents

The agent uses its access methods to go out into local and remote databases to forage for content. These access methods may include setting up news stream delivery to the agent, or retrieval from bulletin boards, or using a spider to walk the Web. The content that is retrieved in this way is probably already partially filtered – by the selection of the newsfeed or the databases that are searched. The agent next may use its detailed searching or language-processing machinery to extract keywords or signatures from the body of the content that has been received or retrieved. This abstracted content (or event) is then passed to the agent's Reasoning or inter ferencing machinery in order to decide what to do with the new content. This process combines the event content with the rule-based or knowledge content

provided by the user. If this process finds a good hit or match in the new content, the agent may use another piece of its machinery to do a more detailed search on the content. Finally, the agent may decide to take an action based on the new content; for example, to notify the user that an important event has occurred. This action is verified by a security function and then given the authority of the user. The agent makes use of a user-access method to deliver that message to the user. If the user confirms that the event is important by acting quickly on the notification, the agent may also employ its learning machinery to increase its weighting for this kind of event.

CONCLUSIN

The formation coordination between the two robots is achieved without providing the robots with global knowledge of other robots' positions or headings. The leader will not communicate to the follower and the follower is the only one responsible of maintaining the formation. In this paper, we have investigated how works with agent under provider team. To do so, we have developed a formal model for reciprocity in multiagent systems. Strategies are updated through an evolutionary process based on a genetic algorithm. This lets us incorporate the notions of bounded rationality, learning, and adaptation into the analysis.

REFERENCES

[1]Nwana, H. S. (1996). *Software Agents: An Overview*. **11**. Cambridge University Press, Knowledge Engineering Review. pp. 205–244.1996. Software Agents: An Overview. Knowledge Engineering Review, Vol. 11, No. 3, 205-244, Cambridge University Press

[2] Schermer,, B. W. (2007) (paperback). *Software agents, surveillance, and the right to privacy: A legislative framework for agent-enabled surveillance*. **11**. Leiden University Press. p. 140. ISBN 978-0-596-00712-6. Retrieved 2012-10-30.

[3] Wooldridge, M.; Jennings, N. R. (1995). *Intelligent agents: theory and practice*. **10(2)**. Knowledge Engineering Review. pp. 115–152.

[4]Franklin, S.; Graesser, A. (1996). "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents". University of Memphis, Institute for Intelligent Systems. Archived from the original on 1996.

[5] Serenko, A.; Detlor, B. (2004) (PDF). *Intelligent agents as innovations*. **18(4)**. pp. 364–381.

[6] Adonisi, M. (2003) (PDF). *The relationship between Corporate Entrepreneurship, Market Orientation, Organisational Flexibility and Job satisfaction* (Diss.). Fac.of Econ.and Mgmt.Sci., Univ.of Pretoria.

[7]Serenko, A.; Ruhi, U.; Cocosila, M. (2007). *Unplanned effects of intelligent agents on Internet use: Social Informatics approach*. **21(1-2)**. Artificial Intelligence & Society. pp. 141–166.

[8]Haag, Stephen (2006). *Management Information Systems for the Information Age*. pp. 224–228.