ARCHITECTURE AND PERFORMANCE OF INTELLIGENT SYSTEM

Rajesh Kumar, Research Scholar, PhD-Computer Science, NIMS University, Jaipur (Raj) India[1]
Dr. B. S. Jangra, Associate Professor (CSE), Haryana Institute of Technology, Bahadurgarh (HR) India[2]
bsjangra@gmail.com[2]

Abstract

In this paper defined the heterogeneous agents that solve problems by sharing the knowledge retrieved from the WEB and cooperating among them for an intelligent system. The control structure of those agents is based on a general purpose Multi-Agent architecture based on a deliberative approach. Any agent in the architecture is built by means of several interrelated modules: control module, language and communication module, skills modules, knowledge base etc. Basically the control module uses an agenda to activate and coordinate the agent skills for a best intelligent system. This agenda handles actions from both the internal goals of the agent and from other agents in the basic environment. In this paper describes how SKELETONAGENT has been used to implement different kinds of agents and a specialized for a Multi-Agent System. The implemented Multi Agent System is the specific implementation of a general WEB gathering architecture, named MAPWEB, which extends SKELETONAGENT. MAPWEB has been designed to solve the basic problems in WEB domains through the integration of information gathering and their resources planning techniques. The system which uses information gathered directly from several WEB sources (plane, train, and hotel Companies etc.) To solve travel problems. The proposed a architecture allows integrating the different agent's tasks with AI techniques like planning to build MAS which is able to gather and integrate information retrieved from the WEB to solve problems.

Keywords: Multi-Agent systems, Architectures & Framework, information gathering, WEB-based systems

Introduction

In an intelligent system a person that learns fast or one that has a vast amount of experience could be called "intelligent" but the systems comparative their level of work performance in reaching its objectives and goals. This implies having experiences where the system learned which actions best let it reach its objectives. The Intelligent Agents and Multi-Agent Systems research fields have experimented a growing domain interest from different research communities like Artificial Intelligence, Software Engineering, Psychology, etc. Those researchers' fields try to solve the basically two distinct goals and several characteristics like autonomy, proactiveness, coordination, language communication, etc. .This goal tries to obtain an adaptive and intelligent program which is able to provide the adequate request to the inputs received from the environment. On the other hand, it is possible to coordinate several of those intelligent system agents to build complex societies. When considering societies of agents, the new issues are arising, like social organization, cooperation, knowledge representation, coordination, or negotiation. Those research fields allow testing and simulating theoretical models and intelligent system architectures and framework in complex and real domains. There is a wide range of different domains that can be used to test agent organizations like business management system, robotics, the WEB crawler etc.

A Basic and common point of interest for previous areas of research that how to define and design the individual information gathering agents that make up these systems and how to coordinate and organize groups of agents. Different arquitectures and models have been successfully implemented in several domains and it is possible to learn from those experiences to build other agent-based models that could be applied in new intelligent system domains. With respect to systems that find and use information from the internet, the closest to our goals is the field of Information Gathering, which intends to integrate a set of different basic information sources with the aim of domain querying them as if they were a single source. However, these architectures do not intend to use AI techniques in a generic way, as we do, but only to select the appropriate WEB sources or control the behavior of agents. In summary, our approach consists of a flexible, reliable and generic MAS architecture and framework that can use Artificial Intelligence and WEB gathering techniques, by means of agenda-based agents. Besides describing the architecture, an important part of the paper deals with its instantiation into a Information Gathering system, named MAPWEB, and how it can be applied into a particular WEB domain (MAPWEB-ETOURISM).

IS: Agent control cycle

Intelligent System describes the control cycle of the agents that follow the model described in the previous section. It can also be seen in Fig. 1, this figure illustrates the control algorithm, which can be summarized as:

- First, the control module checks the agenda periodically until the first act is inserted. When a particular task needs to be achieved, the initial act is inserted in the agenda. This act could be considered as the main goal or the initial goal of the agent. Other acts will be generated in order to achieve this initial goal. New acts could arrive from other agents at any moment. Once several acts have been inserted into the agenda, a *priority-Policy* is applied to sort them out (this evaluation happens at all cycles of the agent, because the priority of the acts change dynamically). The control module will *select* the act with the highest priority (*Acti* = *max*; in Fig. 1).

- When a particular act is selected, the control module *Evaluates*() this act and selects the skill that will perform the associated task (*Evaluate*(*Acti*) selects *Skills*). For instance, when an *act: Planning* is selected, the act and associated parameters are provided to the planning skill of the agent.

- Now, there are two possible situations:

  The act needs information from other tasks and cannot be executed (the act is not *Ready*() yet for execution), so it will be inserted newly in the agenda (its priority will be increased in the next cycle: *Increment priority*(*Acti*)).

  The act is *Ready*() for execution (the act does not need any other information). However, it is necessary to take into account two new situations to know if the act is directly *Executable*(): ∗ If the act can be decomposed into simpler acts, the skill will expand this act into them, assigning new priorities to these acts, and finally adding them into the agenda

  (*n)* =1*Actij .priority* ()).

  If the act is directly executable (all the information is available and it is not possible to expand), it has provided to the correspondent *Execute*() function associated to this skill. This control cycle continues while the agenda contains acts. If the agents have their agendas empty, they will wait for new tasks to perform. This algorithm integrates the agenda, the heuristics and the available skills as related modules to implement different behaviours in the agents. This approach is very flexible because it is possible to modify the behavior of the agent, or the way to achieve a goal by just modifying one or several of the following characteristics of acts:

- The *priority* of a particular act can be modified by the control module or by the skill. The priorities of the acts can be redefined by using rules or policies, thus changing the global solving algorithm performed by the agents.

- The *evaluation* of an act determines which skill will execute the act. If no skill is available, the control module will generate a fail.

- Acts in the agenda are *selected* by means of policies. By changing the policy it is possible to change the behaviour of the agent.

Skeletonagent Agent Architecture

This IS model shown in the previous section our extensions to that model. We address both the agent and the Multi-Agent architecture. The agent architecture of SKELETONAGENT, which is based on the agent model described in the previous section. Agents in SKELETONAGENT are composed of several modules.

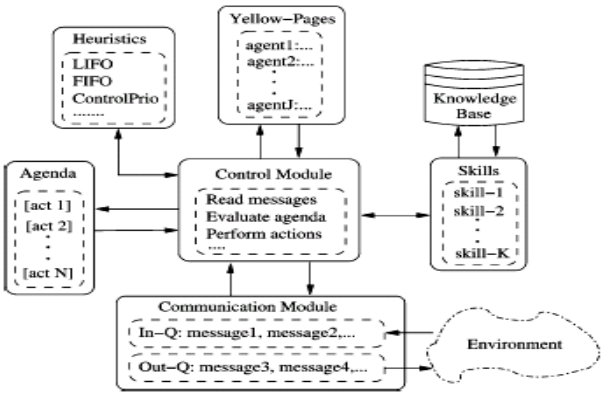

Figure 2. SKELETONAGENT architecture of an intelligent agent

A. Agenda

The Agenda is a dynamic structure that stores their items named *acts*. These acts represent the actions that the agent is considering at a given moment. The agents implemented using SKELETONAGENT architecture share a standard communication language to perform actions over their environment. A message in above section called *performative*. Any performative can be translated into one or many acts

that could be performed by the receiver agent. When any agent receives performative, Once this act is selected, its execution could generate new acts that will be recursively added to the agenda. These performatives are used by different agents to implement several acts like:

i) **(*achieve, tell*):** are used by different agents to require the **execution** of a specific task and to answer with the obtained results. When any agent receives *achieve*, the related act contains the type of skill that will be necessary. For instance, when a planning agent receives an achieve performative to solve a problem, several skills can be used: *planning*, *case-based planning*, or *askothers*. When this same performative is received by a WEB agent other skills like: *caching*, or *access-Web* could be used by the agent to execute the act.

ii) **(*insert, delete*):** are used to insert or delete a specific fact (from the sender agent) in the Knowledge Base of the receiver agent. For instance if an agent is temporarily unavailable in the society, and this fact is known by the manager agent, it is necessary to delete it from the yellow pages of all agents.

iii) **(*register, unregister*):** are used by the controlagents in the Multi-Agent systems to manage the insertion and deletion of the agents in the society.

iv) (*ok, ping*): are used by the control-agents when it is necessary to know the state of a particular agent.

v) **(*wake-up, sleep*):** are used by different agents to activate or suspend temporarily their functions

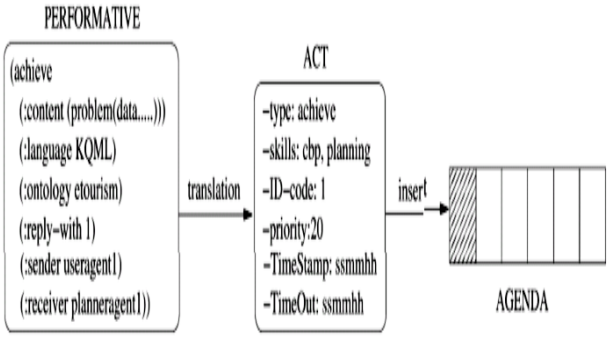

Figure 3. Relation between a per formative and its translated act.

B. Heuristics & Skills of Intelligent System agent

IS Agents having a set of heuristics function that is used to decide at any time what act to select from the agenda. Actually, any agent can be implemented using in its control module three different types of heuristics (*LIFO, FIFO and ControlPrio*). The first two heuristics correspond to the LIFO (Last In First Out) and FIFO (First In First Out) policies. These heuristics can be used when it is not necessary to select the acts in a particular order and allow to test the agents with simple behaviours. However, these heuristics present several problems when it is necessary to apply a priority in the execution of the acts, because there could be acts with high priority that need to be executed quickly.

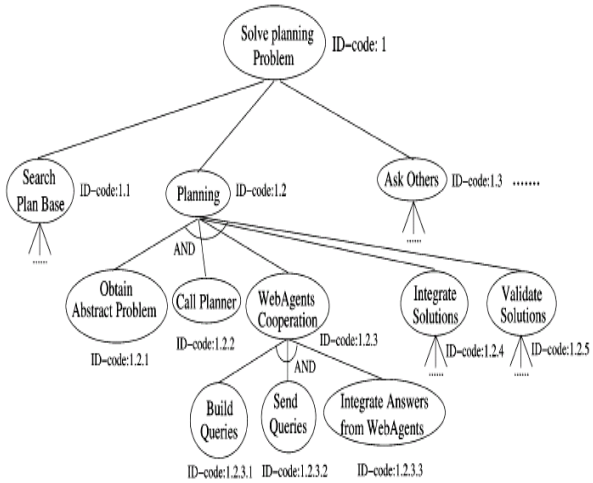

Figure 4. Expansion of an initial act into several subacts.

As explained before, an agenda contains acts. Some of them can be decomposed into lower level acts, which are subsequently introduced into the agenda. When a particular act (atomic, or low level) cannot be decomposed further, it will be executed by the agent. Those executable acts are actually the skills *Si* of the agent. Automatic access to the WEB, or executing a planner are examples of skills. Figure 4 shows how a high level act (to solve a planning problem) can be decomposed into several subacts, taking into account that the Planner Agent who receives the problem has several skills to obtain solutions for planning problems. From the point of view of decomposition, it is useful to divide acts into two types: AND acts and OR acts. AND acts require all its subacts to finish successfully whereas OR acts need only one of them to end. For instance, *act:WebAgentsCooperation* of Fig. 4 is an AND act because it needs its three subacts to end successfully. On the other hand, *act:Solve-Planning- Problem* will end with success if any of its subacts (*act:SearchPlanBase, act:Planning, . . .*) obtains a solution to the planning problem.

C. IS Knowledge base

Knowledgebase shown that and stores knowledge that can be used by the agent skills. For instance, in MAPWEB a set of agents specialized in planning are used in their team. These IS agents need to use several knowledge sources to achieve their main goal: to solve a problem using proper planning. The knowledge used by these agents is: a description of the problem to be solved, the operations (operators) that can be used (represented in a Domain description), several domain dependent heuristics techniques.

Information gathering in the WEB

In IS an information gathering and retrieval technique, extract knowledge from documents stored in the WEB by taking advantage of their inner structure. IG systems like SIMS use information retrieved from relational databases but other IG systems, like Heracles, can use other kinds of sources that provide the information into a semi-structured way (the useful information is stored inside the retrieved document and it is necessary to extract or filter the information previously). Several problems arise when WEB IG systems are designed and implemented:

1. An IG system needs to select, access and filter the information from the appropriate sources, and finally, reason with the gathered knowledge to build a solution.

2. IG systems have to deal with multiple, distributed, and heterogeneous WEB repositories. WEB sources can be heterogeneous in both content and format. Besides, the number and types of those repositories grow over time.

3. WEB servers can be down at some times.

4. The IG system might find a large number of solutions, so it is necessary to manage this overload to provide them in a comprehensive way to the user.

Related work

The main aim of this particular section is to describe the ideas related to the two main issues of this work: agenda architectures and WEB information systems With respect to agenda-based systems, the most closely related to our work are the CooperA and *ABC*2. The CooperA (cooperating agents) platform is a software framework that supports the cooperation of heterogeneous, distributed and semiautonomous Knowledge-Based (KB) systems. In CooperA the KB systems are translated into application agents that are finally be integrated into one system. The users can interact with all the agents using a user interface agent. The CooperA architecture is built by a set of interconnected layers. These layers are: *The CooperA kernel*, *the message-passing mechanism*, *the collection of CooperA system Agents*, and finally *the Community of Application-Specific Agents*. Any agent in the CooperA architecture is a dynamic structure that communicates with other agents in the system through a message passing skill.

Conclusions

In this paper we have presented SKELETONAGENT, an agenda-based flexible architecture for building agents that can participate in MAS. The utilization of an agenda-based architecture for agents, allows coordinating multiple agents with heterogeneous skills in a flexible way. Also, it is very simple to change the behavior of the agents by modifying the policies used to manage the agenda. Although our approach is based on work described in related references, their ideas have been extended in several ways. First, we allow defining acts which are composed of sub acts that can be executed in parallel and can have AND/OR structures. This allows defining alternative ways to achieve a goal and to mix different high level tasks at the minimum level of granularity. For instance, if there are two tasks *A* and *B* composed of several subtasks *A*1, *A*2 . . . and *B*1, *B*2, . . . , every subtask will be processed when they are ready, thus interleaving the two tasks *A* and *B* in the most appropriate order. Second, we have oriented our architecture to facilitate integrating AI solving techniques with IG techniques to work in WEB domains. To do so, we have instantiated SKELETONAGENT to build a MAS, named MAPWEB, that combines AI planning and WEB information gathering techniques. MAPWEB is a generic architecture that can be used by evelopers in any domain requiring planning and WEB sources. We have used it to solve travel assistant problems in the etourism domain (MAPWEB-ETOURISM). This example has also been used to illustrate the behaviour of the agenda-based architecture. Some of SKELETONAGENT features have also been tested in other domains like problem solving through Genetic Programming and WEB News Gathering, which shows that it is a general framework

References

1. R. Aler, D. Camacho and A. Moscardini, Cooperation between agents to evolve complete programs, Chapter in: *Intelligent Agent Software Engineering*, Valentina Plekhanova, University of Sunderland, United Kingdom, Idea Group Publishing ed., 2003, pp. 213–228.
2. J.L. Ambite, G. Barish, C.A. Knoblock, M. Muslea, J. Oh and S. Minton, Getting from here to there: Interactive planning and agent execution for optimizing travel,

in: *The Fourteenth Innovative Applications of Artificial Intelligence Conference (IAAI)*, Edmonton, AB, Canada, 2002.

3. Y. Arens, C.Y. Chee, C.-N. Hsu and C.A. Knoblock, Retrieving and integrating data from multiple information sources, *International Journal of Cooperative Information Systems*
4. M. Balabanovic, Y. Shoham and T. Yun, An adaptive agent for automated web browsing, 1995.
5. R.I. Brafman and M. Tennenholtz, Modeling agents as qualitative decision makers, *Artificial Intelligence* **94**(1–2) (1997), 217–268.
6. W. Brenner, R. Zarnekow and H. Wittig, *Intelligent Software Agents. Foundations and Applications*, Springer-Verlag, New York, 1998, ISBN: 3-540-63411-8.
7. R.A. Brooks, Intelligence without representation, Number 47 in: *Artificial Intelligence*, 1991, pp. 139–159.
8. D. Camacho, R. Aler, C. Castro and J.M. Molina, Performance evaluation of Zeus, Jade and SkeletonAgent frameworks, in: *Proceedings of the IEEE Systems, Man, and Cybernectics Conference (SMC-2002)*, Hammamet, Tunisia, IEEE, 2002.
9. D. Camacho, D. Borrajo, J.M. Molina and R. Aler, Flexible integration of planning and information gathering, in: *Proceedings of the European Conference on Plannin.*
10. P.R. Cohen, A. Cheyer,M.Wang and S.C. Baeg, An open agent architecture, in: *In Working Notes of the AAAI Spring Symposium: Software Agents*, AAAI, Menlo Park, CA, 1994, pp. 1–8.

About Authors:

1. Rajesh Kumar, Post Graduate in Computer Science, Presently a Research Scholar in NIMS University, Jaipur (Raj.)
2. Dr B S Jangra, PhD Computer Science and Working as Associate Professor in HIT.