Journal of Achievements in Materials and Manufacturing Engineering

VOLUME 37
ISSUE 1
November
2009

Cooperating agents approach to task execution planning

J. Madejski*
Division of Materials Processing Technology, Management and Computer Techniques in Materials Science, Institute of Engineering Materials and Biomaterials, Silesian University of Technology, ul. Konarskiego 18a, 44-100 Gliwice, Poland
* Corresponding author: E-mail address: janusz.madejski@polsl.pl

Received 13.08.2009; published in revised form 01.11.2009

Analysis and modelling

ABSTRACT

Purpose: The aim of the paper is to investigate the behaviour of the cooperating agents having to agree task execution.
Design/methodology/approach: A heuristic method is proposed for scheduling tasks that should be carried out by a group of agents. An important issue is that the agents have to complete their negotiations to commit to the task and carry out it in real time. This task was solved by splitting the problem solving into two layers which make it possible to obtain the rough solutions first and further improve them next as much as possible.
Findings: The first layer proposed refers to tasks assigned static priorities and its goal is to meet the real time requirements. Next, the agents can proceed to obtain the optimized solution, provided there is time to do it before the task execution has to begin.
Research limitations/implications: Analysis of the negotiations procedure was done and model examples were worked out to develop a planning system based on these design requirements.
Practical implications: Implementing the real time task execution planning in artificial agents brings them closer to the job shop floor real life requirements, making it possible to develop software entities mimicking reactions of humans and capable of joint task execution planned on the fly as needed.
Originality/value: Analysis of the real time task planning and execution of groups of agents.
Keywords: Artificial agents; Real time; Execution planning; Negotiations; Group of agents

Reference to this paper should be given in the following way:
J. Madejski, Cooperating agents approach to task execution planning, Journal of Achievements in Materials and Manufacturing Engineering 37/1 (2009) 52-56.

1. Introduction

Running manufacturing processes calls for real-time information which can only be reliably and acquired from plant control systems collecting production flow data from all available monitoring devices. The issue is that real life production conditions are seldom crisp and certain with uncertainty in process identification being predominant. All this information is expected to be dealt with by the autonomous agents in contemporary factory automation. Their architecture is made of three components – *sensors*, collecting and transmitting data to the *cognition element*, analysing them, taking into account the current activity plans and deciding actions to take – individually or along with other agents, and finally communicating their decisions the action command to the *effectors*, which eventually may be other agents [1-3]. An agent features an entity that perceives its environment through *sensors* of any nature. The agent proceeds a problem-solving process based on its perception to obtain a single action or a set of actions. The agent then acts upon the environment using *effectors* (required to execute actions).

The issue is that it is difficult to reach optimum decisions in the uncertain conditions in general and to do it in real time, which make the problem even more complicated [4-6]. Therefore, the automated manufacturing process should incorporate various agents which are designed to solve specific problems, being grouped usually at several hierarchical layers [7]. In most cases linguistic variables and rules-of-thumb are used to form the fuzzy logic models, heuristics, based on the relevant domain experts' experience in running production or maintenance processes [8-10]. The proven method of solving this agent design problems is to use the historical data to train and optimise the fuzzy models reqired for the uncertain conditions. The resulting models are implemented into the event driven agents – best the internet-based ones. The quick response requirement, obvious in the real time systems may in many times require co-operation of many agents, based on the inference results of some other fuzzy agents [11-12]. Providing quick response and good quality, in terms of efficiency, agent work plans, which are continuously updated when any new events occur that may affect the plan carried out call for splitting the plan development into two levels: the first level assuring fulfilment of the strict temporal constraints, and the second one being dedicated to acquiring results of higher quality. The second level algorithms attempt to reuse results obtained previously, in similar cases, to make better use of the existing processor time, whenever feasible, however this is not critical for the system operation, as the scheduler operating at the first level has to take actions to meet the deadlines [13-16].

The fuzzy procedures can be invoked to generate decision-making results in the execution stage, which would make up the following framework:

- *Executing agent*: this component is a core component of the proposed model, whose task is to perform the inference processes,
- *Database*: stores the production and system data (usually usinf the MySQL database system). The database is the repository of decision parameters to the executing agent.
- *Data updating agent*: records the selected the currently updated data, while the sensors are gathering new data.
- *Data updating monitor*: this element scans the database and advises the executing agent to start the specified action when the pre-defined conditions are matched.
- *Reporting agent*: this agent displays or broadcasts the generated results.

2. Multi-agent systems

The notion of the multi-agent systems is connected with the behaviour within groups of agents with various features. One should also take into account that there may be agents which may consist of the so-called in-agents, which, makes their co-operation negotiations even more complex. The existence of an agent is not required in advance, before the problem solving context occurs, as the relevant agent or agents may be automatically generated and tuned to solve the specific task. What is important, however, is that the agents are the heterogeneous (in their class) and reusable entities. This means that the agents should be designed for various contexts showing the autonomous and self-interested behaviour, albeit they will have to work for some "common" benefit [17].

Therefore, any multi-agent system should be treated as one demonstrating the following stance:

- *non-benevolence*, as agents should behave in a rational way, trying to maximise their goal function (utility) [5]. This approach makes them less vulnerable to "selfish" or even "malevolent" behaviour of their counterparts, moreover, this is also a way to make them less prone to suffer from "ignorant" (in certain conditions) or incompetent agents who may wish to use them to work with/for them to accomplish certain tasks.
- *autonomy*, as agents prefer to pursuit their own goals, and negotiate with others to make their decision if it is purposeful for them to adopt goals of others, even in part. This is usually implemented in the agent utility functions as a strive to obtain a certain incentive in return for participation of tasks carried out for others.
- *readiness to act to reach multiple goals*, which involves a possibility of situations when conflicting goals may occur. This is due to different local utility measures that the co-operating agents may have.
- *heterogeneity,* which is demonstrated by various utility functions of agents, their different architectures or knowledge representation, all that resulting in adoption of different goals finally.

Reaching any decision, calls for either reactive or deliberative architecture, while the real-time conditions call for using the reactive approach first, which will guarantee deciding the right action, while the deliberative one may be employed provided there is a time slot for it. The decision above may be a decision pertaining to an action to be taken by a group of agents.

It is required that the agents that have to take decisions in real-time are implemented as applications on a real-time platform, best QNX [18], as the industry standard well field proven since years. QNX microkernel architecture provides the built-in distributed processing, which means that a control application – in this case agent on any node - can transparently access resources (including other agents) - disks, ports, protocol stacks, etc. - residing on any other node in the network. Therefore, a network of individual machines becomes, provides a seamless, distributed control over hundreds of thousands of I/O points - sensors.

From a temporal point of view, a critical agent's task is characterized by a period and a deadline. In this way, the available time for the agent to obtain a valid response is strictly bounded and between the time it is (periodically) released and its deadline. In this interval, the agent has to provide a good-enough response to its subproblem, given the current situation of the environment. It also has a priority, and a worst-case execution time for its reactive component.

The supervisory agent can define a deadline and a period associated to each agent of the subordinate level [3]. Reducing the problem complexity of the agent's task may require dividing it into subtasks [3, 4, 11]. When the strictly time dependant part of a task has been completed, the first-level scheduler passes the control to next, second-level scheduler, which in turn can execute the optional component of this task. In every case when the first-level scheduler passes the time dependant goal, having accomplished it, it should communicate to the second-level scheduler the existing available time. This is the time left for the second-level scheduler to perform its planning and to execute the optional plan components.

A number of heuristics are usually used as those that are used for the first-level, that has to provide results in real-time [7, 15, 19].

- *Deadline Monotonic (DM)*: This policy prevails in case of very restrictive deadlines, simply ordering the tasks in accordance with their maximum execution deadline. This approach makes execution possible of the largest possible number of tasks, yet it does not let obtain the high-quality results.
- *Best Importance First (BIF*): This approach orders the tasks according to the importance the tasks have for the agent. This policy makes the resulting agent plan quality better, being quite simple to implement.
- *Earliest Deadline First (EDF)*: This policy takes the task deadline into account; however, in another way than in case of the DM policy. In this case the task execution deadline is taken into account keeping in mind when the agent was activated, which means it analyses the agent's condition more closely, using the remaining time until the task's deadline, not its absolute deadline. The results obtained are better than obtained for DM policy, as the tasks that do not expire soon may be postponed.
- *High Quality First (HQF)*: This approach focuses on task quality rather than its urgency or deadline. Since it does not refer to the deadline, it may not execute tasks which would otherwise be executed using a different ordering method .
- *High Slope First (HSF)*: This approach sorts the tasks according to their quality as a function of their execution time. It attempts to merge two previous arguments (quality and deadlines). The results are improved compared to the previous heuristics, as both approaches are merged.

3. Task planning for cooperating agents

3.1. Resolving of conflicts

The agents may co-ordinate their strategies to reach a certain goal and agree on some *multi-plan* [5, 13]. This process consists in calculating the vectors of utilities for all agents involved in negotiations. The negotiations include bargaining, to let the agents obtain some incentive for spending some of their resources (energy, time, etc.) for a goal, which was not delegated to them initially. However, as the agents are autonomous, they cannot be forced to co-operate, so it may happen that thay can decide to proceed alone without entering into any agreements which might affect adversely their freedom. There is a way to overcome this limitation – the agents may gamble on the group multi-plan, trying to optimise it, instead of randomising their individual plans in a "selfish" way, which eventually might let them enter into co-operation. This approach calls for existence of layered agent architecture [2, 7, 13].

3.2. Scheduling

The real-time scheduling can be efficiently carried out using the anytime algorithm [21, 21] which is an iterative refinement algorithm that can be interrupted and asked to provide an answer at any time. An important feature of such algorithm is that the quality if its results improve to a certain level with the growing amount of its run time. Anytime algorithms are characteristic of their performance profile, being a function that links the run time allowed to an anytime algorithm with the quality of the results produced by the algorithm [22]. The main approach is based on modelling the behaviour of an intelligent agent (a set of anytime algorithms) by providing explicit allocation of resources to each anytime algorithm. The reason to allocate the resources is maximizing the total quality of the computation of the anytime algorithm set – agent actions.

The anytime algorithms have the advantage of being able to provide some solution at any moment, therefore they can respond immediately to changing environmental situations. An important feature is that the quality of the answers they provide may have the controlled quality, by assigning them some extra amount of run time should the environmental conditions allow. This is why the anytime algorithms are very useful in realtime applications. One can apply separate anytime algorithms to each type of activity the agent has to carry out (inclusive the comples agents composed of the in-agents). The main task in all cases is finding algorithms whose expected solution quality (i.e., agent plans) improves in some anticipated monotonic way. The anytime algorithms exist for some classes of problem domains, like scheduling problems, typical for agent behviour modelling – Fig.1.

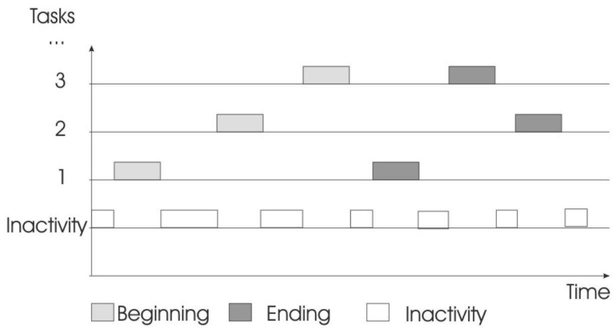

Fig. 1. Example of the agent plan before refinement

The unrefined agent plans, like those shown in Fig.1 are included in the test parameters of the agent bahaviour simulator system (Fig.2). The test data generator provides the agent beavious simultor with the test data sets, with the varying values of the parameters describing its environmental conditions.

Behaviours represent the alternative ways of reacting the environmental conditions charactersitic of the agent (starting activity, halting it, re-planning, assuming goals in co-operation with other agents, etc.). At run time, the agent is always in one behaviour called the current active behaviour. The agent may switch to another behaviour should he detect a certain condition in the environment. An effective approach in deciding how to change the agent behavious is splitting this task into smaller problem-solving entities. This approach makes it possible to specify the problem-solving knowledge in a well structured modular way [13, 16, 17]. By negotiation, all concerned agents (this approach includes also the in-agents mentioned above) can cooperate to solve the entire problem by generating a multi-plan.

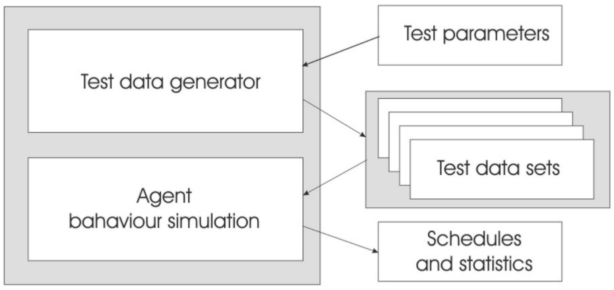

Fig. 2. Agent behaviour simulator

Such cooperation is achieved by sharing of the results obtained by different agents, in a global system memory common to a group of cooperating agents, which may be formed through broadcasting the call for co-operation [13]. Each agent has a reflex layer that assures a minimal quality real-time response (i.e., with the guaranteed execution time, specific for the particular task). Its second, real-time deliberative layer, is used to improve the initial response, time allowing.

All this negotiation and generation of multi-plans, composed of activities of the individual agents are carried out in a modelled world - set of beliefs, being supplemented with all the requred domain knowledge relevant to the agent's tasks (most often in a form of the If-Then rules).

The initial, raw work plans for the agents, their internal mental states and their sets of beliefs are stored in a blackboard which can be accessed by all co-operating agents [23]. This architecture makes it possible to modify the agents' beliefs should it be required to carry out their tasks, or to adapt them to their new applications. This can be also done at run time.

4. Conclusions

The following conclusions can be reached from the results that have been obtained from literature review and own work:

- Effective task planning calls for splitting the process into two stages – the real time stage, calling for the reactive agent architecture, and the slack time optimisation, with the use of the deliberative agent architecture.
- Agents can co-operate assuming, if needed, each other's goals, however in a non-benevolent way, which guarantees obtaining some incentives by them.
- Operation of agents may take place in the uncertain conditions, and the tasks (goals) may be accomplished even by the heterogeneous agent groups, some of which may be generated on the fly.

Elements of the agent environment were modelled using EXSYS Professional and real time models were developed in QNX OS.

References

[1] M. Wooldridge, N.R. Jennings, (Eds.), Agent theories, architectures, and languages: a survey. In: Intelligent Agents, ECAI'94 Workshop on Theories, Architectures, and Languages. Springer, Berlin,1995.

[2] J. Madejski, State-of-the art of Distributed Artificial Intelligence in Manufacturing Systems, Proceedings of the 5th International Conference Achievements in Mechanical and Materials Engineering AMME'96, Wisła, 1996 121-124.

[3] J. Madejski, Agents as the Building Blocks of the Responsibility-Based Manufacturing Systems, Proceedings of the International Scientific Conference - Challenges to Civil and Mechanical Engineering, Wrocław, 1997, 213-220.

[4] W.C., Benton, H. Shin, Manufacturing planning and control: The evolution of MRP and JIT integration. European Journal of Operational Research 110 (1998) 411-440.

[5] J. Madejski, Fuzzy logic approach to the autonomous agent task utility function evaluation, Proceedings of the 8th International Conference "Achievements in Mechanical and Materials Engineering" AMME'99, Gliwice-Rydzyna-Pawłowice-Rokosowo, 1999, 241-244.

[6] A. Beskese, C. Kahraman, Z. Irani, Quantification of flexibility in advanced manufacturing systems using fuzzy concept. International Journal of Production Economics 89 (2004) 45-56.

[7] J. Madejski, Agent architecture for intelligent manufacturing systems, Journal of Achievements in Materials and Manufacturing Engineering 29/2 (2008) 167-170.

[8] M. Shimbo, T. Ishida, Controlling the learning process of real-time heuristic search. Artificial Intelligence 146/1 (2003) 1-41.

[9] F. Barber, V. Botti, A. Crespo, D. Gallardo, E. Onaindi, A temporal blackboard for real-time process control. Journal of Engineering Applications of Artificial Intelligence 7/3 (1994) 225-266.

[10] F. Baykoc, S. Erol, Simulation modelling and analysis of a JIT production system, International Journal of Production Economics 55 (1998) 203-212.

[11] P. Stoop, V. Wiers, The complexity of scheduling in practice, International Journal of Operational and Production Management 16/10 (1996) 37-53.

[12] R.E. White, J.N. Pearson, J.R. Wilson, JIT manufacturing: A survey of implementations in small and large US manufacturers, Management Science 45/1 (1999) 1-15.

[13] J. Madejski, Agents as building blocks of responsibility-based manufacturing systems, Journal of Materials Processing Technology 106 (2000) 219-222.

[14] C. Carrascosa, M. Rebollo, J. Julian, V. Botti, Deliberative server for real-time agents, Lecture Notes in Artificial Intelligence 2691 (2003) 485-496.

[15] C. Carrascosa, J. Bajo, V. Julian, J.M. Corchado, V. Botti, Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications 34 (2006) 2-17.

[16] Y. Qian, M. Zheng, X. Li, L. Lin, Implementation of knowledge maintenance modules in an expert system for fault diagnosis of chemical process operation. Expert Systems with Applications 28 (2005) 249-257.
[17] S. Ossowski, Co-ordination in Artificial Agent Societies, Springer-Verlag, Berlin, Heidelber, New York, 1998.
[18] http://www.qnx.com/
[19] V. Botti, C. Carrascosa, V. Julian, J. Soler, Modelling agents in hard real-time environments. Lectures Notes in Artificial Intelligence 1647 (1999) 63-76.
[20] A. Garvey, V. Lesser, Design-to-time real-time scheduling. IEEE Transactions on Systems, Man and Cybernetics 23/6 (1993) 1491-1503.
[21] A. Garvey, V. Lesser, A Survey of research in deliberative real-time artificial intelligence, Journal of Real-Time Systems 6/2 (1994) 317-347.
[22] S. Zilberstein, Operational rationality through compilation of anytime algorithms. AI Magazine 16/2 (1995) 79-80.
[23] F. Barber, V. Botti, A. Crespo, D. Gallardo, E. Onaindi, A temporal blackboard for real-time process control. Journal of Engineering Applications of Artificial Intelligence 7/3 (1994) 225-266.