Industrial Process Optimisation with JIAC (Demonstration)

Marco Lützenberger Tobias Küster Thomas Konnerth Alexander Thiele Nils Masuch
Axel Heßler Jan Keiser Michael Burkhardt Silvan Kaiser Jakob Tonn Sahin Albayrak
DAI-Labor, Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
firstname.lastname@dai-labor.de

ABSTRACT

In this demonstration we present the EnEffCo project as a showcase for the JIAC V agent framework. EnEffCo addresses the problem of optimising energy consumption in industrial production processes. As the optimisation of the processes is handled by JIAC V agents, the project profits from a number of industry-grade software features of the JIAC V framework, such as stability, reliable communication, standard compatibility and management interfaces.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Multiagent systems*

Keywords

agent framework; mas development; industrial adoption

1. INTRODUCTION

With the increased saturation of devices that are networked and computing enabled, modern software systems become more and more complex and distributed. The domains and scenarios that are to be controlled and managed by software systems require new and extended functionalities that need to be implemented. In order to enable software developers to implement and maintain such systems, elaborate frameworks are necessary that feature encapsulation, robustness, and reuse. Furthermore these frameworks need to fulfil industrial quality and performance standards.

One approach to address these challenges is the concept of *Agent Oriented Software Engineering*, or *AOSE* that dates back as far as 1993, when Yoav Shoham published his widely cited and influential article [8] and established an entirely new branch of research. Today, 20 years later, there are many agent frameworks that support the design, the implementation and the runtime monitoring of multiagent systems. Popular examples for agent frameworks are Jason [2], 3APL [4], Jadex [7], JACK [3] or JADE [1], which is frequently used for proof-of-concept implementations among the agent-research community. Nevertheless, despite the many years of research in agent oriented technologies, it is far from being audacious to say that the agent community was as yet not able to convince industrial players to adopt the multiagent system paradigm. JIAC V[1], or JIAC, provides many features which are usually required for industrial projects and in this demonstration we present these exact capabilities to the agent community.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

2. THE JIAC V AGENT FRAMEWORK

The JIAC V framework [6] focuses on industry-grade features such as robustness, reliable communication, stability, performance and reuse. The main motivation in the development of the JIAC V framework was to create an agent framework that fulfils the requirements of modern software projects and thus bridges the gap between agent technology and the software industry.

JIAC V was implemented on top of a number of standard Java technologies, such as the Spring framework, JMX management interfaces, and JMS messaging. This provides increased robustness and standard compatibility and simplifies the usage of the framework for regular Java developers.

An important aspect of the JIAC V framework is the support for monitoring and the management of agents and applications. The central tool for this is the ASGARD runtime monitor [9]. This monitor allows a developer or administrator to observe and control arbitrary platforms, agents, and applications at runtime. Agents can be stopped, modified or deployed dynamically and applications can be controlled and adapted by accessing the standardised Java JMX interface. The ASGARD monitor will be instrumental in the demonstration of the JIAC V framework and the scenario that will be described in the next section.

3. THE ENEFFCO PROJECT

We present the *EnEffCo*[2] optimisation software [5] as an exemplary JIAC V application that was successfully used in an industrial context, namely, the automotive industry. We sketch the EnEffCo approach first and outline the implementation with JIAC afterwards. Finally we discuss our experiences in applying JIAC for the implementation of the EnEffCo software.

[1]**J**ava **I**ntelligent **A**gent **C**omponentware, version five.
[2]**En**ergie-**Eff**izienz**co**ntrolling am Beispiel der Automobilindustrie (eng. Energy efficiency controlling in the automotive industry)

3.1 Approach

The EnEffCo software optimises production processes of the automotive industry in terms of energy costs. The application exploits the fact that (in Germany), the industry is able to procure energy by means of short-term strategies at the day-ahead energy market, where prices are highly dynamic. It was our idea to minimise energy costs by shifting energy-consuming sub-processes to time slots with low energy costs. Yet, contemporary production processes comprise many co-depending sub-processes and feature a high degree of complexity, thus, shifting parts of the overall process is not as easy as it sounds. We decided to use *Evolution Strategy* to produce reliable optimisation results in a timely manner. An 'optimisation server' receives the logical structure of the production process in the form of a bipartite graph. This graph contains architectural information, such as the sequential arrangement of all production steps, involved machinery and storages, but also meta information, such as the duration and the energy consumption of production steps. The optimisation server also receives a production target, a timeframe and information on the energy price development. Based on the input data, an initial production plan is generated. In the next step, this initial population is mutated inasmuch as sub-processes are randomly shifted. Each optimisation server produces a defined number of mutations. A fitness function is used to evaluate the quality of each mutation and the most effective production plans are selected as input for the next stage of evolution. In this next iteration, the production plans are mutated again. The optimisation finishes whenever the quality of the offspring remains steady across several generations.

3.2 Implementation

We have implemented our optimisation servers as JIAC agents and defined an interaction protocol in which a designated agent—the optimisation client—broadcasts an optimisation problem. Available servers either accept the assignment and initiate the optimisation process or refuse as a result to different reasons (e.g. other activities). Whenever a server finishes, the result is sent back to the client, who compares the different results and selects the best. Finally, the best measured graph is presented to the operator, who is able to configure his production line accordingly.

We used JIAC for the implementation for two main reasons. To start with, JIAC allows for the execution of several optimisation servers (JIAC agents) in parallel and thus counters the problem of stochastic optimisation algorithms to get stuck in local optima. The simultaneous execution of optimisation agents with different initial populations and random mutations increases the chance to overcome local maxima significantly. Since JIAC agents are truely multi-threaded, the application's execution speed remains equal while the quality of the optimisation increases. The second reason for the appliance of JIAC is the reliable communication of the agent framework. It is easy to physically distribute agents among different hosts and thus to increase the overall performance of the optimisation software at will. As an example, for complex optimisation scenarios, additional servers may be added. Later, when the additional calculation performance is not required any longer, those can be removed. The loose coupling between JIAC agents simplifies this process greatly and also supports maintenance issues. Malfunctioning or out-dated optimisation agents can be replaced individually; it is no longer necessary to shut down the entire optimisation software, but to remain functional and to update selected optimisation servers one by one.

4. CONCLUSION

The purpose of this demonstration is to underline the reliability of JIAC V. In order to do so, we present the Enffco project, where the reliability of JIAC is inevitably required. We designed the EnEffCo software to run in an industrial context. To ensure reliability, we deployed several optimisation servers in the form of JIAC agents and have each server processing the optimisation problem, respectively. The distribution contributes to the software's quality in many aspects. We compensate unforeseen errors of individual agents, and also increase the performance of our Evolution Strategy, which demonstrably produces better results, the more initial populations are used. Finally, we exploit JIAC's reliable communication to ensure that results are correctly communicated through the computer network.

5. REFERENCES

[1] F. Bellifemine, A. Poggi, and G. Rimassa. JADE — A FIPA-compliant agent framework. Technical report.

[2] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-agent Systems in AgentSpeak Using Jason*. Wiley Series in Agent Technology. Wiley-Blackwell, October 2007.

[3] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. JACK — Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty, Ltd., 1999.

[4] K. V. Hindriks, F. S. D. Boer, W. V. der Hoek, and J.-J. Meyer. Agent Programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[5] T. Küster, M. Lützenberger, and D. Freund. Distributed optimization of energy costs in manufacturing using multi-agent system technology. In P. Lorenz and K. Nygard, editors, *Proceedings of the 2nd International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies, Maho Beach, St. Maarten*, pages 53–59, March 2012.

[6] M. Lützenberger, T. Küster, T. Konnerth, A. Thiele, N. Masuch, A. Heßler, J. Keiser, M. Burkhardt, S. Kaiser, and S. Albayrak. JIAC V — A MAS framework for industrial applications (extended abstract). In T. Ito, C. Jonker, M. Gini, and O. Shehory, editors, *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, Sait Paul, Minnesota, The USA.*, 2013. To appear.

[7] A. Pokahr, L. Braubach, and K. Jander. Unifying agent and component concepts: Jadex active components. In *MATES*, pages 100–112, 2010.

[8] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[9] J. Tonn and S. Kaiser. ASGARD — A graphical monitoring tool for distributed agent infrastructures. In Y. Demazeau, F. Dignum, J. M. Corchado, and J. B. Pérez, editors, *Advances in Practical Applications of Agents and Multiagent Systems*, volume 70 of *Advances in Intelligent and Soft Computing*, pages 163–175. Springer, 2010.