Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

A Novel Multi-Agent System for Complex Scheduling Problems

Peter Hillmann
Tobias Uhlig
Gabi Dreo Rodosek
Oliver Rose
Department of Computer Science
Universität der Bundeswehr München
Neubiberg, 85577, GERMANY

ABSTRACT

Complex scheduling problems require a large amount computation power and innovative solution methods. The objective of this paper is the conception and implementation of a multi-agent system that is applicable in various problem domains. Independent specialized agents handle small tasks, to reach a superordinate target. Effective coordination is therefore required to achieve productive cooperation. Role models and distributed artificial intelligence are employed to tackle the resulting challenges. We simulate a NP-hard scheduling problem to demonstrate the validity of our approach. In addition to the general agent based framework we propose new simulation-based optimization heuristics to given scheduling problems. Two of the described optimization algorithms are implemented using agents. This paper highlights the advantages of the agent-based approach, like the reduction in layout complexity, improved control of complicated systems, and extendability.

1 INTRODUCTION

There is a rising interest in agent-based software development since it promises efficiency paired with a high mobility while approaching complex problems. Intelligent software agents are tools which handle their tasks independently within a larger context. These agents collectively operate in a multi-agent software (MAS) architecture, solving tasks by common endeavor. This paper introduces a concept of a highly flexible multi-agent system, that is applicable in various problem domains.

Our approach is evaluated using a combinatorial sequential scheduling problem and compared to traditionally object and component based approaches. We show, that using agents supports the development, by providing a better overview and thereby assist in handling the system complexity. With agents dependencies become visible and we obtain an impression at the phenomena of emergence.

Emergence refers to occurrence of new structures or properties, resulting from the cooperation of single elements in a complex system. These structures are not directly obvious and only arise from the interaction between the elements of the system (Stephan 1998). The common endeavor of more or less specialized directives surpasses the accumulated capability of all individual system elements (Ferber 1999). The Micro-Macro-problem from the domain of distributed artificial intelligence illustrates this challenge very well (Hillebrandt 1999). Well known emergent systems are ant colonies and particle swarms. They have been adapted for optimization problems.

There are two basic approaches to model the cooperation and organization in agent systems. We employ the role model which, in contrast to the service model, relies on describing the capabilities of an agent to perform a task instead of binding a fixed task to an agent. Currently there is, however, no standardized concept for these architectures (Reenskaug et al. 1995, Nguyen et al. 2010).

In Section 2 we describe the requirements we try to address using the MAS approach. Subsequently we provide a short overview of related work. In Section 4, we present our MAS concept. Section 5 and 6 outline the application area and the simulation-based optimization algorithms used for the evaluation in Section 7.

2 REQUIREMENTS

Agents are used in many different application areas, e.g., in information retrieval for automatic filtering, in electronic commerce for transactions (XJ-Technologies 2000), in network management systems to detect intrusions (Koch and Golling 2013), or in simulations of entire hospitals (Herrler 2007). Usually a MAS is used when a single agent cannot meet all necessary requirements. Generally, the most important requirements are the following:

- Handle complex systems and challenges in a practicable way, relying on emergence (Brenner, Zarnekow, and Wittig 1998).
- Guarantee flexibility by being extendible and adaptable, generally applicable, and platform independent.
- Provide a client-server environment for cluster systems enabling distributed calculation, parallelism, and dynamic load balancing.

Figure 1 presents the overall concept of our approach relying on agents on different levels. The interaction of generally applicable and specialized agents guarantees an extendible approach.

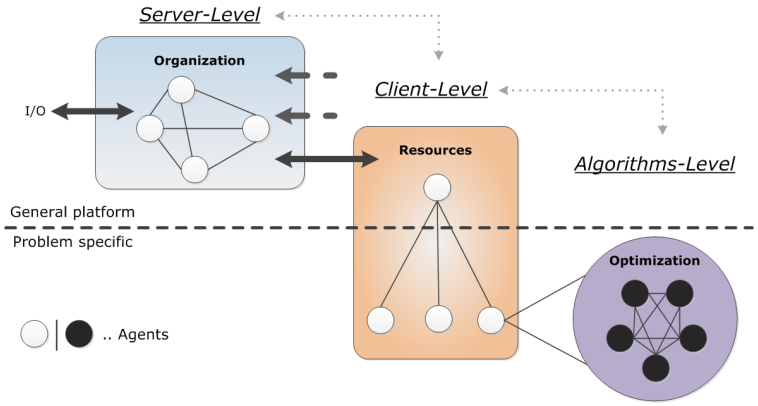

Figure 1: Abstract architecture of the MAS.

3 BACKGROUND AND RELATED WORK

There is a plethora of agent architectures, however not single standard has emerged so far. The most common architecture is called reference model, which describes the general construction of a MAS. This reference is provided by the FIPA, a nonprofit group of various companies. This working group deals with the communication, interaction and management of agents as well as transport of agent messages and interoperability with different network architectures. Other well known agent architectures include: BDI-Modell (Georgeff et al. 1999), PAGE-Modell (Russell and Norvig 2009), InteRRaP (Oluyomi et al. 2006) and ACT-R (Anderson 1996). These architectures focus mainly on the description of agents, while providing only little information considering the cooperation and organization of multiple agents. A common but not necessarily optimal approach is using hierarchical structure to manage agents (Schillo 2000). Consequently one agent controls multiple helper agents, which unfortunately often leads to the occurrence of bottlenecks. The realized system FABMAS (Mönch et al. 2003) come close to our requirements and has strong influence

on our design. But it has a hierarchical production control scheme. The system is also specific for semi conductor factories and maps their structure to the agent approach. The ideas in (Yilamz and Ören 2009) describe a similar vision, but an implementation and feasibility evaluation are missing. We want to promote a flexible MAS with independent agents, focusing on cooperation instead of simple delegation. It is therefor mandatory that these intelligent agents exhibit autonomic behavior.

For the interaction and communication, the agents use the agent communication language (ACL) from FIPA (Odell 2004). This language is very abstract and it relies on a intricate ontology service (Bellifemine et al. 2007). Based on the ACL, this paper presents a new concept to simplify and customize this aspect while keeping the flexibility.

4 CONCEPT OF THE MAS

Our concept relies on two types of agents. General applicable agents to realize the MAS and special agents with a problem dependent implementation. Instead of using a hierarchical structure we employ independent agents that cooperate using a kind of artificial intelligence. The MAS follows an model-view-controller principle.

The following section will illustrate the architecture of the MAS and describe the agents and its roles and functionality. Subsequently the agent communication and the processing of tasks is explained. Our agents register in a MAS by putting their name and role into a central listing-service. This is realized as a role of the agent and is an integral part of the environment. For the implementation, we use the framework JADE (Telecom Italia Lab 2000). It offers a basic agent environments and agent functionalities. The simulation of the cluster tool is done using the SAGE framework. SAGE is an inhouse software solution including a discrete event simulator for typical scheduling problems and several optimization heuristics.

4.1 Architecture of the MAS Lagoon

Our system is separated into a server and a client application to address the distributed processing (see Fig. 2). This enables the construction of a private or open cluster combining the power of multiple host-systems for intensive computations.

Every agent we use relies on three basic behavior schemes. The **Initial-behavior** is run only once upon creation of an agent. The agent registers with his role at the listing-service and performs basic setup operations. After initialization the agent is either working or communicating. During **Working-behavior** the actual tasks are processed. The tasks arrive via messages. The **Messaging-behavior** relies on an incoming and an outgoing queue for messages. Each incoming task will be queued according to its priority. Control instructions are processed immediately. Other tasks are buffered for the Working-behavior. Outgoing messages will be send whenever the receiver is reachable and able to accept them. Working and messaging are processed in parallel with a separate thread for each behavior.

The constitution of the server part of the MAS starts with the creation of a **Server-Agent**. This agents initiates the MAS by spawning all server related agents. These are: Task-Contractor-Agent, Controller-Agent, Collector-Agent, Splitter-Agent, Answer-Agent, and Jabber-Agent. The Server-Agent is also responsible for eventually terminating the created agents.

The **Task-Contractor-Agent** loads the inital tasks, that should be processed by the MAS. Upon creation it loads a list of task and assigns priorities to them. Then it sends those task to appropriate Agents and terminates after sending all Tasks. The Task-Contractor-Agent is most basic option to provide tasks to the MAS.

The central manager for optimization intelligence is the **Controller-Agent**. It chooses a fitting optimization approach and selects an adequate set of parameters. It adapts parameters dynamically during run-time and is capable of learning depending on the current state of the MAS.

The **Collector-Agent** takes a package of tasks and separates them into individual tasks for parallel processing. After completion of all tasks it collects them and evaluates the generated results. This offers the

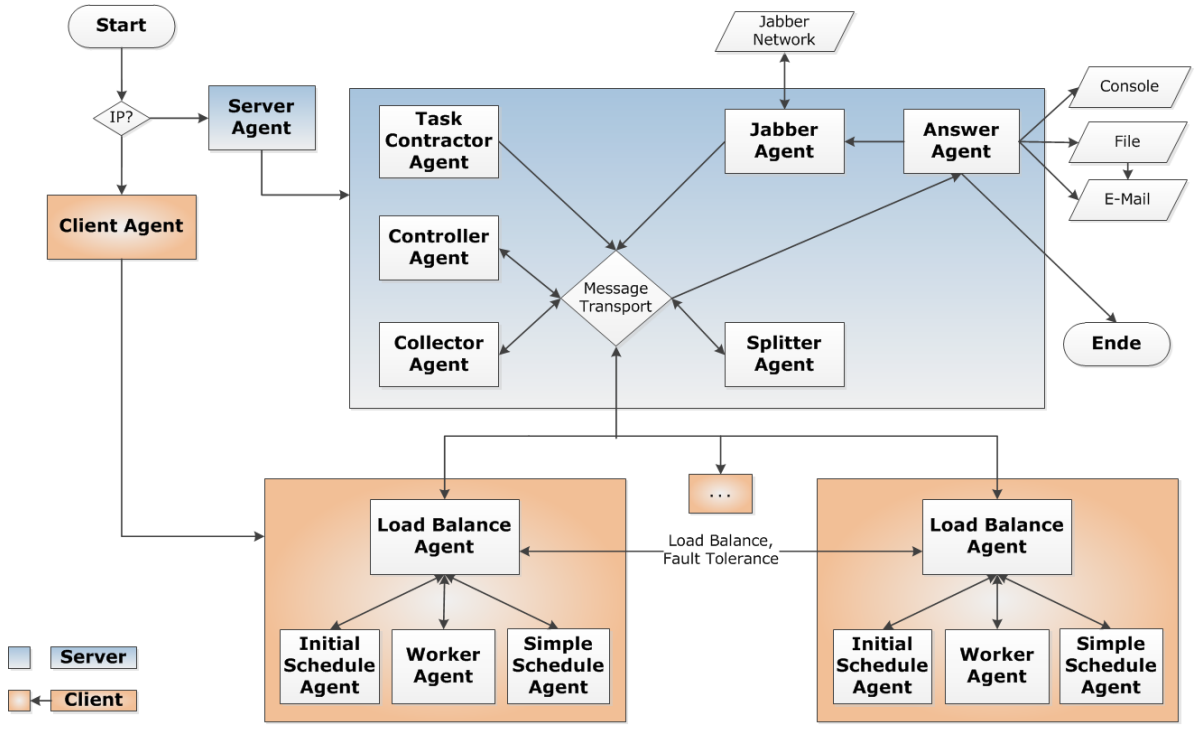

Figure 2: Architecture of the MAS. The blue colored agents are part of the server application and the orange colored agents belong to the client version of the MAS. Communication interfaces are shown in rhomboids.

possibility to process a single task with different methods. It identifies appropriate solutions and transfers the results to other agents for further processing, e.g, output or learning.

Parallel processing is facilitated by the **Splitter-Agent**. In contrast to the Collector-Agent which handles multiple tasks, it splits a single complex tasks into independent sub-tasks that can be processed in parallel. Ultimately the processed sub-task are reassembled to complete the original task.

The output of the MAS is handled by the **Answer-Agent**, providing complete decoupling of input and output. The desired way of output communication is specified for each task and the Answer-Agent delegates the output to the appropriate interface/agent. Basic output is realized as writing to a file or sending a mail message. For more elaborate communication other agents are used.

For example the **Jabber-Agent** provides an interface to communicate using the instant messaging protocol XMPP (Saint-Andre 2004) In addition to his role of providing output via XMPP the Jabber-Agent serves to accept input for the MAS. For example simple text messages can be used to start new tasks.

The **Client-Agent** is similar to the Server Agent with regards to his main responsibilities but with a focus on the client side of the MAS. It individually generates agents to fulfill a certain role on the client. For example it could create a Load-Balancer Agents that is connected with the MAS supporting it with the available computation power on the client. Alternatively it creates an Contractor-Agent to submit tasks to the MAS.

A key Agent is the **Load-Balancer-Agent** that manages the intersection between task management and actual task processing. Depending on the incoming tasks it dynamically creates specific worker-agents to handle them or assigns tasks to already existing workers. It is capable to manage the load of the client system, by creating an appropriate number of workers. Furthermore, it can dynamically delegate tasks to or request tasks from Load-Balancer-Agents operating on other clients. The Load-Balancer-Agent periodically

exchange messages regarding their load state to identify agents with high or low load. This enable network wide load balancing in the whole MAS.

The group of **Worker-Agent**, as their name indicates, are the agents that perform the actual optimization. Depending on their setup they fulfill a certain role for processing. These Agents are highly problem related since they use a special algorithm to solve the given challenge. However we provide an general interface to easily implement a fitting worker-agent for each respective problem. With regard to our problem domain, the system has worker agents that generate initial schedules (Initial Schedule Agent) and other workers that iteratively optimize the initial schedule (Simple Schedule Agent). Idle Worker-Agents are terminated.

4.2 Communication of the agents

Communication in the MAS is realized using data packages based on the ACL (Agent Communication Language). Our messaging concept is simpler than the complete ACL architecture, but nevertheless remains universally applicable. We use wrapped objects that conform with the ACL, called **Coat-Package**. The idea is inspired by the OSI model and effectively decouples messaging in the MAS from the problem dependent parts in the tasks. A Coat-Package may transport any task and contains information not standardized by the ACL. Additionally to the specification it holds control parameters for the MAS, like priority or selected output channel. Each package has a unique identifier and a log for tracking purposes.

All information with problem dependent bits is kept in an **Task-Package**. For our scenario we use a Scheduling-Task-Package. It contains a test setup including all required parameters and the desired objective function. It also holds a list of necessary processing steps to generate a valid solution. A **Collection-Package** can be used to bundle multiple Task-Packages.

Problem independent **Control-Packages** enable the user to regulate the MAS during run time. Every agent has the required logic to process incoming control packages that should effect them. For example a control-package may be used to shutdown the MAS or activate debugging in agents. Valid targets for control packages are certain agents, all agents of the MAS or other packages.

During run time, the MAS is dynamic, e.g., client can join or leave the MAS. Therefore it is necessary to track all task, to avoid losing some of them. Managing agents keep a copy of a task until the receive a notice that a certain procedure has actually been complete by a worker agent. In case of failures, like connection losses, task are resubmitted to other available agents. Furthermore, it is possible to migrate agents from one system to another one during run time. This concept supported by the JADE framework guarantees failure safety based on redundancy. This approach is scalable and guarantees the redundancy to successfully operate in a dynamic environment.

5 APPLICATION AREA

We evaluate our approach applying it to a combinatorial sequence-dependent problem. This problem considers the scheduling of job for cluster tools in the production line of semiconductor manufacturing (Unbehaun and Rose 2007). Cluster tools can process jobs in parallel, however the processing time varies dependent on the processing of the jobs. Our objective is to minimize the completion time of the entire sequence to maximize the throughput. Figure 3 illustrates the global improvement of processing jobs in parallel at the cost of slowing down the individual jobs due to shared resource. Discerning the optimal processing sequence correspond to the problem of finding the most fitting permutation of jobs with given recipes (R_i). The amount of permutations (S_{all}) can be calculated with formula 1. Using Stirlings approximation illustrates the exponential growth of the resulting search space depending on the queue length (L).

$$S_{all} = \frac{L!}{\prod_{i=1}^{r} R_i!} = \frac{L!}{R_1! \cdot R_2! \cdot \ldots \cdot R_r!} \qquad L! \approx \sqrt{2 \cdot \pi \cdot L} \cdot \left(\frac{L}{e}\right)^L \tag{1}$$

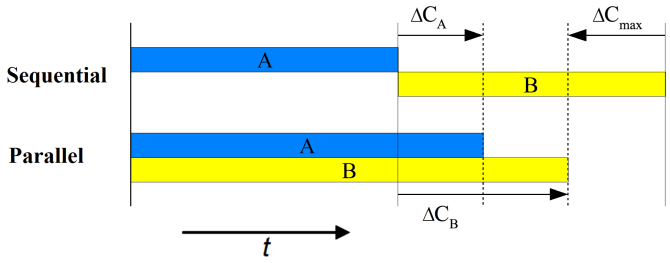

Figure 3: Comparison of sequential and parallel processing.

The problem we face is a NP hard with a multidimensional and multimodal search space. An extended problem we consider employs parallel cluster tools, resulting a multi-step problem, which is still NP-hard, since the partitioning and sequencing influence each other.

6 SCHEDULING ALGORITHMS

We developed different optimization algorithms for our reference problem. Of these algorithms, Particle-Swarm-Optimization and Central Complex are implemented as separate agent systems. Fundamentally the approaches fall into two categories – sequencing and partitioning algorithms. Partitioning considers the assignment of jobs to a certain machine, while sequencing is about finding an adequate processing order of jobs on one machine (see Fig. 4 and 5). Both, sequencing and partitioning, are performed in order to find a schedule that leads to a minimal makespan (C_{max}). Both subproblems are NP-Hard and interlock with each other.

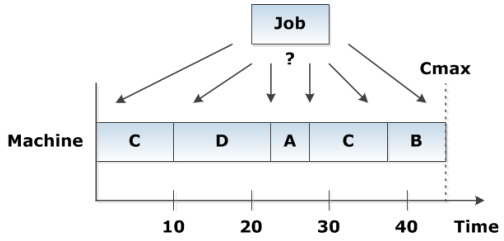

Figure 4: Sequencing Problem.

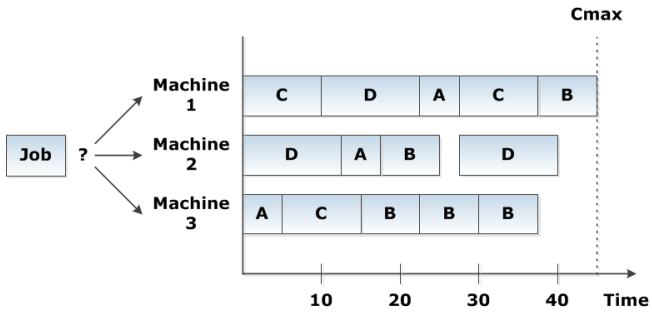

Figure 5: Parallel Machine Problem.

We consider two distinct approach to partition jobs. The basic approach attempts to discern an overall sequence of jobs, that is cut into equisized chunks. The resulting subsequences are assigned to the respective machines. The advanced approach is to use dynamic partitioning where the partitions are no longer require to be equisized. Dynamic partitioning is especially usefully for problems with multiple different machines and if you consider constraints like different processing speeds depending on the machine or machine qualifications

6.1 Random Down Swing

The first proposed algorithm is a stochastic hill climbing approach with optional reinitialization. This Random Down Swing (RDS) approach is a simple sequencing algorithm – for problems with multiple machines we uses the basic cutting approach to generate sequences. It starts with a randomized initial sequence and iteratively tries to improve the schedule by swapping two jobs in the sequence. The two jobs are selected randomly. A swap is accepted, if it leads to an improvement. The algorithm resets with a new initial sequence when a certain number of swaps were tried without improving the schedule. The limit

before reinitialization is used, should be adapted to a given problem instance, for our scenarios we use an empirical found value of 700. The optimization terminates after a fixed number of steps and returns the best found schedule.

6.2 Simplex Method

The Simplex Method is a popular algorithm for linear programming and was developed by (Dantzig 1963). It is part in the CPLEX library from IBM (Lima 2010). This optimization algorithm is specialized for the search in constrained solution spaces. The simplex method start at any vertices of constraints and walk along edges of the constructed polytope to extreme points with better objective values, until it reaches the optimum. It is based on the assumption that the global optimum is part of the constructed polytope, so the algorithms searches only on the edges of the constraints. But the algorithms has performance problems with multi-dimensional and multi-modal spaces, because of the high variability. Other disadvantages are the adaption to the application area as well as the worst-case complexity with exponential time (Klee and Minty 1972).

6.3 Particle-Swarm-Optimization

The Particle-Swarm-Optimization (PSO) was originally proposed by Kennedy and Eberhart (1995) and is inspired by bird flocks. We adapted it to our problem and the discrete search space. Each particle is modeled as an individual intelligent agent. Therefore the resulting PSO is itself a MAS. Initially particles are spread randomly across the search space. We allow only positions that correspond to valid solutions and assign the respective fitness value to them. Each particle contains a sequence of jobs to encode a certain schedule. A single step to move in the search space is realized by swapping two jobs. With respect to the fundamental movement rules used in PSO, i.e., move towards global/local optimum or move randomly, we choose a suitable swap of two jobs for each particle.

Based on the agent approach, it is very simple to extend and improve the classical PSO. Additional agents are added to the algorithm as pseudo particle. These additional particle do not according to the classical PSO rules. These special particles encapsulate a traditional optimization heuristics, however the it can be influenced by the swarm and can in return influence the swarm. For example the special agents may perform local optimization on solutions found by a moving particle and then provide the optimization results as guidance for other particles.

The MAS implementation of PSO has some interesting advantages. The main benefit is the asynchronous distributed optimization. Particle agents explore and evaluate solutions largely independently and therefor may be distributed in a network of multiple computers. Asynchronous exchange of information between agents allows a timely update of discovered global optima. Asynchronous communication can obviously lead to outdated information regarding the global optimum persisting in certain parts of particle swarm. We however observed no negative influence resulting from the obsolete communication artifacts.

6.4 Central Complex

The Central Complex (CC) algorithm is a combination of partitioning and sequencing algorithms. The RDS algorithm is used to generate sequences for each machine. Partitioning is approached dynamically, starting from a random set of partitions. For each iteration RDS determines a good sequence for a given machine. Then a random job is shifted from the machine with the highest makespan to the machine with the lowest one. Shifting is managed in a way to avoid assigning jobs to unqualified machines. After shifting a job from one machine to another machine, the RDS optimizes the sequences again. Reinitialization is used to repeatedly start from different initial partitioning sets. A Meta-Optimization calculates appropriate parameters for the algorithm, e.g., number of reinitializations and iterations used for RDS.

7 SIMULATION AND ASSESSMENT

We put our concept to the test, using "Lagoon" a prototypic implementation to optimize a set of scheduling problems. We chose a set of test setups reflecting typical real world challenges (see Table 1). The performance of the algorithms stagnate at around 10k simulation calls.

Table 1: Parameters of test setups.

Test Setups	
Machines	1..4
Machine Types	uniform or mixed
Job Count	16..60
Job Types	3..10 recipes
Simulation Calls	1k, 10k or 100k
Repetitions	100

Figure 6 and 7 summarize the results of our study. For two of the test problems we were able to calculate an optimal solution using a distributed brute force approach for an adapted single machine test setup. Generally a Monte Carlo (MC) optimization approach serves as a reference. It simply tests a random candidate solution for each iteration. An effective approach should obviously return better results than repeatedly guessing a solution.

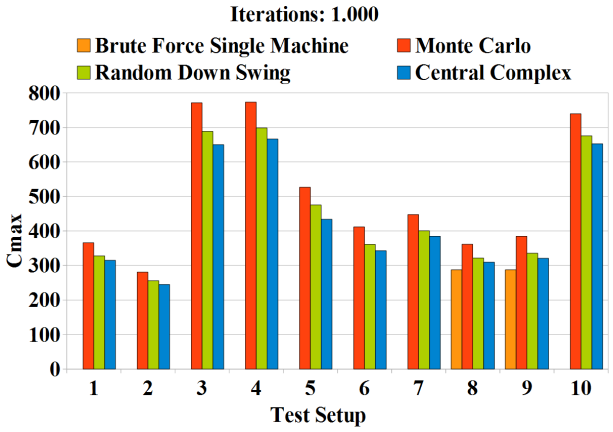

Figure 6: Simulation results using 1.000 iterations. machines.

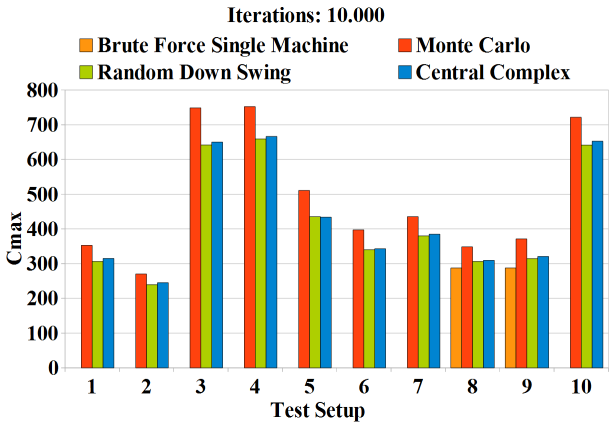

Figure 7: Simulation results using 10.000 iterations. machines.

As we expected our approaches exceeded the performance of MC. Random Down Swing (RDS) consistently provided the best results for all test setups when given enough calculation time (10k iterations). On the other hand the Central Complex method returned acceptable results with fewer iterations (1k). For test setup 5, a setup with a mixed set of machines and unequal quantity of job types, slightly outperformed RDS even for test using more iterations. Overall the quality of results was satisfying and came close to the optimum for the instances where brute force was used.

In addition to the numerical results we did observe the expected advantages of an MAS. It enabled us to effectively distribute the processing across multiple servers (mixed architecture containing N servers), significantly shortening the calculation time. During the development process we successively added new features benefiting from the flexibility and extendability of the approach.

We conclude our survey with a comparison of our approaches in contrast to the simplex method. Table 2 and 3 show typical results for this comparison – both of them are camparable in complexity. The theoretical value for the simplex method results from the calculated simplex tableau. In contrast the practical value is

reached when we construct an actual solution based on the tableau. For test setup 2 the simplex method clearly outperformed the other approaches, returning a better overall result while needing less computation time. Regarding test setup 1 however, the simplex method fell short. The practical result did not come close to the predicted theoretical value and it took a relatively large amount of time to generate the solution. For this scenario even MC outperformed the simplex approach while the best results were achieved by RDS.

Table 2: Simulation of test setup 2

Algorithms	Max	Mean	Min	Time
Simplex-theo.	-	371	-	-
Simplex-prac.	-	385	-	0.03 s
MC	417	405	388	0.25 s
RDS	413	394	379	0.25 s

Table 3: Simulation of test setup 1

Algorithms	Max	Mean	Min	Time
Simplex-theo.	-	365	-	-
Simplex-prac.	-	421	-	26 s
MC	436	416	396	0.26 s
RDS	417	398	375	0.27 s

8 CONCLUSION AND OUTLOOK

We reached a productive cooperation of agents without relying on a hierarchical structure or fixed grouping. The concept of role models supports the developer while organizing the intelligent agents. The combination of agents, role model and wrapped communication packages enables an orthogonal implementation of single components based on the model-view-control principle. Algorithms and other agents can be added to the MAS dynamically. Several users can send tasks to the MAS in parallel. We observed a reduction in layout complexity and improved control regarding complex systems. In contrast to traditional object-oriented and component-based approaches the agent concept is more flexible – especially during runtime – and supports distributed computation. Furthermore, the design of our MAS allows us to easily adapt it to various problem domains. This paper provides an approach that overcomes the biggest challenge of MAS, which is the need of an elaborate organization of agents.

With regard to the scheduling domain our evaluation showed that computation intensive tasks can be solved efficiently using the developed MAS and the proposed simulation-based algorithms. Both RDS and CC algorithms reach reliably good results.

ACKNOWLEDGMENTS

This work was partly funded by FLAMINGO, a Network of Excellence project (ICT-318488) supported by the European Commision under its Seventh Framework Programm.

REFERENCES

Anderson, J. R. 1996. “ACT: A Simple Theory of Complex Cognition”. *American Psychologist Association* 51:355–365.

Bellifemine, F., G. Caire, and D. Greenwood. 2007. *Developing Multi-Agent Systems with JADE*. England: John Wiley & Sons Ltd.

Brenner, W., R. Zarnekow, and H. Wittig. 1998. *Intelligente Softwareagenten - Grundlagen und Anwendungen*. Springer.

Dantzig, G. B. 1963. *Linear Programming and Extensions*. The RAND Cooperation, Princeton: University Press.

Ferber, J. 1999. *Multi-Agent Systems - An Introduction to distributed Artificial Intelligence*. Harlow: Addison Wesley Longman.

Georgeff, M., B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. 1999. *The Belief-Desire-Intention Model of Agency*. Berlin Heidelberg: Springer.

Herrler, R. 2007. "Agentenbasierte Simulation zur Ablaufoptimierung in Krankenhäusern und anderen verteilten, dynamischen Umgebungen". Dissertation, Faculty of Computer Science, Julius-Maximilians-Universität Würzburg, Würzburg, Germany.

Hillebrandt, F. 1999. *Künstliche Sozialität? - Allgemeine soziologische und sozialphilosophische Reflexionen zur VKI-Forschung.* Work Package 4, Technische Universität Hamburg-Harburg, Hamburg, Germany.

Kennedy, J., and R. Eberhart. 1995. "Particle Swarm Optimization". In *Proceedings of the 1995 IEEE International Conference on In Neural Networks*, Volume 4, 1942–1948. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Klee, V., and G. J. Minty. 1972. "How good is the simplex algorithm?". In *Proceedings of the Third Symposium on Inequalities*, edited by O. Shisha, 159–175. University of California: Academic Press, Inc.

Koch, R., and M. Golling. 2013. "Architecture for evaluating and correlating NIDS in real - World networks". In *Proceedings of the 5th International Conference on Cyber Conflict (CyCon)*, edited by K. Podins, J. Stinissen, and M. Maybaum, 335–354. Tallinn, Estonia: NATO CCD COE Publications.

Lima, R. 2010. "IBM ILOG CPLEX - What is inside of the box?". Last modified July 15th, 2014. http://egon.cheme.cmu.edu/ewocp/docs/rlima_cplex_ewo_dec2010.pdf.

Mönch, L., M. Stehli, and J. Zimmermann. 2003. "FABMAS: An Agent-Based Sytem for Production Control of Semiconductor Manufacturing Processes". In *Proceedings of the First International Conference on Industrial Applications of Holonic and Multi-Agentsystems for Manufacturing*, edited by V. Marik, D. McFarlane, and P. Valckenaers, 258–267. Prague, Berlin Heidelberg: Springer.

Nguyen, D. N., K. Usbeck, W. M. Mongan, C. T. Cannon, R. N. Lass, J. Salvage, W. C. Regli, I. Mayk, and T. Urness. 2010. "A Methodology for Developing an Agent Systems Reference Architecture". In *Proceedings of the 11th international conference on Agent-oriented software engineering (AOSE'10)*, edited by D. Weyns and M. P. Gleizes, 177–188. Berlin Heidelberg: Springer.

Odell, J. 2004. "FIPA Agent Management Specification". Last modified March 18th, 2004. http://www.fipa.org/specs/fipa00023/SC00023K.pdf.

Oluyomi, A., S. Karunasekera, and L. Sterling. 2006. "Design of agent-oriented pattern templates". In *Proceedings of the 2006 Australasian Software Engineering Conference (ASWEC)*, edited by J. Han and M. Staples, 113–121. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Reenskaug, T., P. Wold, and O. A. Lehne. 1995. *Working with objects*. Manning Pubns Co, Oslo: Taskon Work Environments.

Russell, S. J., and P. Norvig. 2009. *Artificial Intelligence: A Modern Approach*. 3rd ed. New Jersey: Prentice Hall.

Saint-Andre, P. 2004. "Extensible Messaging and Presence Protocol". Jabber Software Foundation, Network Working Group, RFC 3920.

Schillo, M. 2000. "Vertrauen und Betrug in Multi-Agenten-Systemen". *DFKI Research Reports*.

Stephan, A. 1998. *Emergenz. Von der Unvorhersagbarkeit zur Selbstorganisation*. 2nd ed. Paderborn: Mentis-Verlag.

Telecom Italia Lab 2000. "Framework JADE. Current version is 4.3.2. http://jade.tilab.com/".

Unbehaun, R., and O. Rose. 2007. "Predicting cluster tools behavior with slow down factors". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1755–1760. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

XJ-Technologies 2000. "AnyLogic Simulation Software". Last modified July 15th, 2014. http://www.anylogic.com/consulting/.

Yilamz, L., and T. Ören. 2009. *Agent-directed Simulation and Systems Engineering*. 1st ed. Wiley-VCH.

AUTHOR BIOGRAPHIES

Peter Hillmann is a Ph.D. student at the Universität der Bundeswehr München (UniBwM), Germany. He received his M.Sc. in Information-System-Technology in 2011 from Dresden University of Technology. His areas of research are network and system security with focus on cryptography as well as scheduling and optimization problems. His email address is peter.hillmann@unibw.de

Tobias Uhlig is a research assistant at the Universität der Bundeswehr München, Germany. He received his M.Sc. degree in Computer Science from Dresden University of Technology. His research interests include evolutionary computation and its application to scheduling problems. He is a member of the IEEE RAS Technical Committee on Semiconductor Manufacturing Automation. His email address is tobias.uhlig@unibw.de

Prof. Gabi Dreo Rodosek holds the Chair of Communication System and Network Security at the Universität der Bundeswehr München, Germany. She received her M.Sc. from the University of Maribora and her Ph.D. from the Ludwig-Maximilians University Munich. She is spokesperson of the Research Center Cyber Defence (CODE), which combines skills and activities of various institutes at the university, external organizations and the IT security industry (for instance Cassidian, IABG or Giesecke & Devrient). Her email address is gabi.dreo@unibw.de.

Prof. Oliver Rose holds the Chair for Modeling and Simulation at the Universität der Bundeswehr München, Germany. He received an M.Sc. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of IEEE, INFORMS Simulation Society, ASIM, and GI, and General Chair of WSC 2012. His email address is oliver.rose@unibw.de.