Multi-Machine Scheduling – A Multi-Agent Learning Approach

Wilfried Brauer and Gerhard Weiß
Institut für Informatik
Technische Universität München
D-80290 München, Germany
{brauer,weissg}@informatik.tu-muenchen.de

Abstract

Multi-machine scheduling, that is, the assigment of jobs to machines such that certain performance demands like cost and time effectiveness are fulfilled, is a ubiquitous and complex activity in everyday life. This paper presents an approach to multi-machine scheduling that follows the multi-agent learning paradigm known from the field of Distributed Artificial Intelligence. According to this approach the machines collectively and as a whole learn and iteratively refine appropriate schedules. The major characteristic of this approach is that learning is distributed over several machines, and that the individual machines carry out their learning activities in a parallel and asynchronous way.

1. Introduction

Multi-machine scheduling (MMS) can be briefly characterized as the activity of assigning a number of jobs to a number of performing machines such that certain performance demands like time or cost effectiveness are fulfilled. This activity, which is best viewed as an optimization or constraint satisfaction task, is ubiquitous in everyday life. As the above characterization indicates, scheduling plays a particularly important and critical role in industrial contexts. Apart from this industrial perspective, however, this characterization allows many other interpretations: jobs and machines can stand for programmes and computers, classes and teachers, military missions and soldiers, ships and dockyards, or patients and hospital equipment. Each of these interpretations shows another context in which the need for scheduling arises, and within each of these and similar contexts numerous concrete scheduling scenarios are possible. Scheduling is a highly complex activity which is known to be NP-hard even for many static settings in which e.g. the number of jobs or machines is fixed and known a priori [5]. This means that in general even static scheduling seems to be computationally intractable for greater problem instances. Moreover, assuming a static setting often is an idealization because typical real-world scheduling environments are of considerable dynamic nature. For instance, machines may break down, new jobs may arrive and others may be cancelled, material may not arrive in time, market conditions may change, and so forth. Given the ubiquity and complexity of scheduling, it is not surprising that there is a large number of related literature. Examples of standard books on scheduling theory are [1, 11, 12]. The more recent literature indicates an increasing interest in scheduling in parallel and distributed systems; for instance, see [2, 3, 14]. Moreover, the challenge of finding appropriate schedules has also attracted many researchers in Artificial Intelligence (AI); for instance, see [10, 19]. Scheduling in general is also an important application field in Distributed AI; for instance, see [4, 8, 7, 15].

This paper offers a learning approach to MMS which follows the multi-agent learning paradigm known from the field of Distributed AI (see, e.g., [6, 13, 17, 18]). According to this paradigm learning is shared among several agents and none of them is required to be able to solve the learning task alone. The approach is formulated such that its intended applicability in industrial contexts is obvious. It can be easily seen, however, that its conceptual framework is general enough to be of use in other application contexts as well. The general idea underlying this approach is to consider the machines as active entities or "agents" that are capable of collectively learning appropriate schedules, and the jobs as passive entities or "resources" that at any time are used or handled by the agents according to schedules learnt so far. The individual machines are assumed to be restricted in their abilities and their knowledge; consequently, the overall performance cannot be improved without appropriate interaction. A major characteristic of this approach is that learning is distributed over several machines which can carry out their learning activities independent of each other and in a parallel and asynchronous way. With that, MMS is not considered as a centralized process that is carried out by a single entity (a machine or a human expert), but

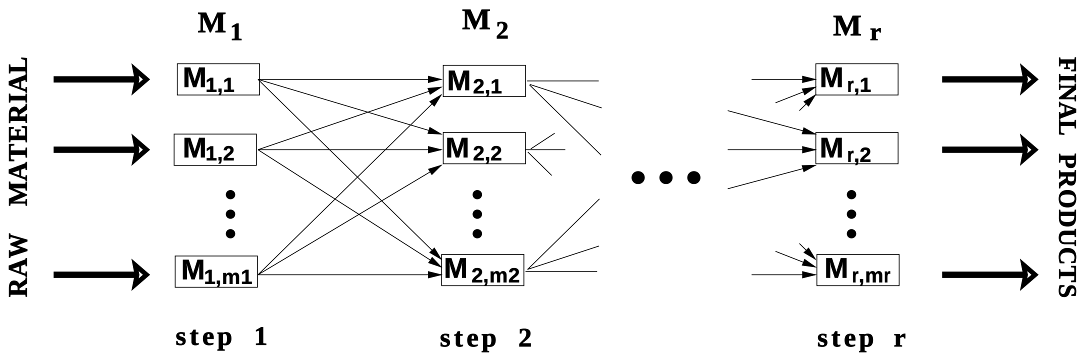

Figure 1. The MMS problem.

as a decentralized process that is conducted by all the machines involved in job execution. This inherent parallelism and distributedness makes the learning approach novel and different from other approaches to scheduling – adapative scheduling as well as scheduling in parallel and distributed systems – which are available inside and outside AI.

The article is organized as follows. The Section 2 describes the MMS scenario addressed in this work. The Section 3 presents the multi-agent learning approach to this scenario. The Section 4 shows experimental results that illustrate the benefits of this approach. The Section 5 concludes the paper with a discussion and an outlook on future work.

2. Multi-Machine Scheduling (MMS)

The general MMS scenario considered in the work described here is shown in the Figure 1. Several machines collectively perform jobs, where a job consists of manufacturing a product out of raw material. For each product the manufacturing process consists of r production steps, where the individual steps are totally ordered in the sense that step i has to be executed before step $i+1$ for all $i \in \{1, \ldots, r-1\}$. A single product (hence, a job) is completed after all production steps have been successively applied to a single piece of raw material. It is assumed that each machine is capable of, or responsible for, executing one of the production steps, and that each production step can be executed by at least one machine. Throughout this paper, M denotes the set of all machines and $M_i = \{M_{i,1}, \ldots, M_{i,m_i}\} \subseteq M$ denotes the subset of machines capable of executing the production step i. It is assumed that $M_i \cap M_j = \emptyset$ for $i \neq j$ and $M = \bigcup M_i$. The machines contained in M_i (M_{i+1}) are called the predecessors (successors) of the machines contained in M_{i+1} (M_i). Each element in $M_1 \times M_2 \times \ldots \times M_r$ consitutes a possible way of manufacturing products, and is called a production path. Until the desired number of products are manufactured, each machine being responsible for production step "1"continuously processes pieces of raw material and immediately forwards the processed material to one of its the successor machines. Associated with each machine being responsible for production step "2"or higher is a waiting queue in which arriving material is put into interim storage if the machine is busy. As soon as a machine becomes idle (i.e., after having completed its production step and having forwarded the processed material to a successor machine), it checks its waiting queue. Material stored in a queue is processed in a first-in-first-out order. The execution of a single production step is considered as an atomic unit, that is, it is non-preemptive and can not be interrupted.

The MMS scenario becomes challenging as soon as time constraints on the manufacturing process are introduced. In the work described here, two elementary types of time constraints are taken into consideration: the time $p_{i,j}$ that a machine $M_{i,j}$ requires for completing its production step, and the time $d_{i,j}^{i+1,l}$ that is required for delivering material from a machine $M_{i,j}$ to its successor machine $M_{i+1,l}$. Both time constraints are assumed to be fixed during the whole manufacturing process (i.e., the completion and delivery times do not change). The Table 1 gives an example of such a time-constrained MMS scenario. In this example the manufacturing process consists of five production steps, where each step can be carried out by four machines. As the table shows, there are differences in the time which the machines require for completing the same production step (i.e., $t_{i,j} \neq t_{i,k}$ for $j \neq k$). For instance, the machines $M_{1,1}$, $M_{1,2}$, $M_{1,3}$, and $M_{1,4}$ require 10, 24, 32, and 39 units of time, respectively, for completing the first production step. Moreover, there are differences in the time required for delivering the material to the same successor machine (i.e., $t_{i,j}^{i+1,l} \neq t_{i,k}^{i+1,l}$ for $j \neq k$). For instance, the delivery from machines $M_{1,1}$, $M_{1,2}$, $M_{1,3}$, and $M_{1,4}$ to $M_{2,1}$ takes 16, 9, 18, and 29 units of time, respectively. All other table entries are to be read analogously. The current waiting time of a piece of material arriving at a machine $M_{i,j}$ (i.e., the time an

i	j	$p_{i,j}$	$d_{i,j}^{i+1,l}$			
			$l=1$	$l=2$	$l=3$	$l=4$
1	1	10	16	8	33	33
	2	24	9	12	8	24
	3	32	18	29	21	14
	4	39	29	12	3	32
2	1	20	37	35	25	30
	2	40	17	21	5	14
	3	37	40	27	12	28
	4	29	6	15	10	18
3	1	19	24	17	6	10
	2	32	6	35	17	9
	3	22	31	15	24	40
	4	7	7	17	28	32
4	1	1	25	20	17	24
	2	30	2	38	6	34
	3	15	24	28	2	36
	4	14	27	22	30	5
5	1	33	-	-	-	-
	2	15	-	-	-	-
	3	23	-	-	-	-
	4	6	-	-	-	-

Table 1. Example of a time-constrained MMS.

arriving piece has to wait in the queue of $M_{i,j}$ until its processing starts) can be approximated by $t_{i,j} \times x_{i,j}$, where $x_{i,j}$ is the current number of pieces of material already waiting in $M_{i,j}$'s queue. The approximated waiting time for $M_{i,j}$ is denoted by $w_{i,j}$. (Obviously it would be more precise to write $w_{i,j}[t]$ and $x_{i,j}[t]$ to indicate that the waiting times vary over time; in order to avoid unnecessary formalism, however, the time index t is omitted.) It is assumed that at each time a machine only knows *(i)* the delivery times d to its successors, *(ii)* the processing times p of its successors, and *(i)* the current approximated waiting times w of its successors. With that, at each time a machine does have only a very limited view of its environment (including the other machines) and the overall status of the manufacturing process.

Introducing time constraints immediately leads to the following basic question called the MMS problem: Given the number of products to be manufactured, what production path should be chosen for each product such that the overall manufacturing time is kept low? What is searched for is a schedule (i.e., an assigment of production paths to the products to be manufactured) that keeps the manufacturing time low. Requiring that the overall manufacturing time is minimized would make the MMS problem particularly difficult, because in general it is a NP-hard problem to find an optimal (shortest) multi-machine schedule. In order to avoid the complexity of NP-hardness, especially in practical and industrial contexts it is only required to find an almost optimal solution in reasonable time. The learning approach described in the next section offers a solution to the MMS problem that aims at fulfilling this more practical requirement.

3. A Multi-Agent Learning Approach

The approach described here offers a multi-agent learning perspective of the manufacturing scenario specified in the preceding section. According to this perspective, each machine takes the role of an active entity or "agent" which in some sense can be said to be intelligent and autonomous, and each piece of material takes the role of a passive entity or "resource" which is used or handled by the agents. The intelligence and autonomy of each machine is manifested in its ability to learn independent of the other machines to which of its successors it should deliver a piece of material in order to keep the overall manufacturing time low. Learning by the individual agents takes place in a parallel way, and this means that the "global" task of searching for an appropriate overall schedule is decomposed into smaller "local" tasks of searching for appropriate successor machines.

The basic idea underlying the multi-agent learning approach is that each machine learns to judge its alternatives of forwarding a piece of material. This judgement is based on the estimated remaining time needed for completing a product. Learning done by a machine consists of two basic activities – deciding about successor machines and adjusting the estimates – as described in the following. These activities are carried out by the machines independent of each other and in a parallel and asynchronous way.

Deciding about successors. $M_{i,j}$ calculates for each of its successor machines $M_{i+1,k}$ the estimated remaining manufacturing time $ERMT_{i,j}^{i+1,k}$, that is, the time that is needed for completing a product out of a piece of material to be delivered to $M_{i+1,k}$, by

$$ERMT_{i,j}^{i+1,k} = \underbrace{\max\{w_{i+1,k} - d_{i,j}^{i+1,k}\ ,\ 0\} + p_{i+1,k}}_{\text{known}} + \underbrace{L_{i,j}^{i+1,k}}_{\text{to be learnt}} , \quad (1)$$

where $w_{i+1,k}$ is the current waiting time of a piece arriving at $M_{i+1,k}$, $d_{i,j}^{i+1,k}$ is the time for delivering a piece of material from $M_{i,j}$ to $M_{i+1,k}$, $p_{i+1,k}$ is the processing time of $M_{i+1,k}$, and $L_{i,j}^{i+1,k}$ is $M_{i,j}$'s estimate of the remaining time required for completing a product after the successor machine $M_{i+1,k}$ will have finished its production step. The expression $\max\{w_{i+1,k} - d_{i,j}^{i+1,k}, 0\}$ takes care of the fact that the waiting and delivery times overlap. As already

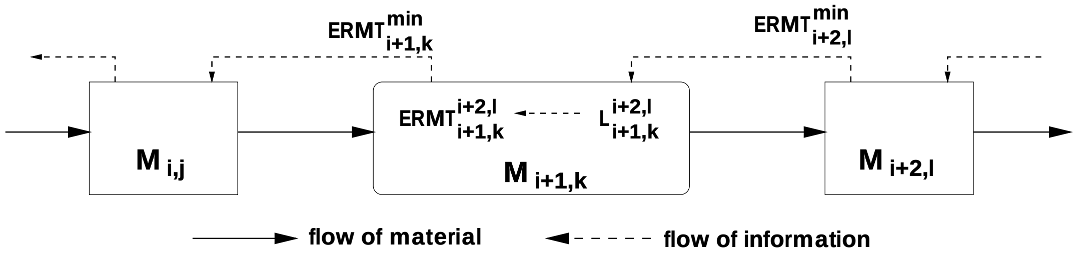

Figure 2. The "chaining effect" of estimate adjustment.

mentioned in the preceding section, the delivery times and the successors' waiting and processing times are known to $M_{i,j}$. What is unknown and has to be learnt by $M_{i,j}$ is the remaining processing time after a piece of material is processed by the successor machine; how this is done is described below. At every time during the manufacturing process a machine $M_{i,j}$ chooses its successors dependent on its estimates $ERMT_{i,j}^{i+1,k}$. More precisely, the probability that $M_{i,j}$ forwards a piece of material to its successor $M_{i+1,k}$ is inversely proportional to

$$\frac{e^{\alpha \cdot ERMT_{i,j}^{i+1,k}}}{\sum_{l=1}^{m_{i+1}} e^{\alpha \cdot ERMT_{i,j}^{i+1,l}}} \quad . \qquad (2)$$

With that, the higher (lower) $ERMT_{i,j}^{i+1,k}$ the lower (higher) is the probability that $M_{i,j}$ forwards pieces of material to $M_{i+1,k}$.

Adjustment of estimates. Assume that $M_{i,j}$ decided to forward a piece of material to $M_{i+1,k}$. After $M_{i+1,k}$ has processed the forwarded piece, it calculates for each of its successors the values $ERMT_{i+1,k}^{i+2,l}$ as described above. Let $ERMT_{i+1,k}^{min} = \min_{l=1}^{m_{i+2}} ERMT_{i+1,k}^{i+2,l}$ be the lowest of these values. $M_{i+1,k}$ informs its predecessor $M_{i,j}$ about $ERMT_{i+1,k}^{min}$, and $M_{i,j}$, in turn, adjusts its estimate $L_{i,j}^{i+1,k}$ according to

$$L_{i,j}^{i+1,k} \quad = \quad L_{i,j}^{i+1,k} + \beta \cdot (ERMT_{i+1,k}^{min} - L_{i,j}^{i+1,k}) \qquad (3)$$

The intended effect of this adjustment, which can be viewed as a simplified Q-learning scheme [16], is to decrease (increase) an estimate L if leads to an overvaluation (undervaluation) of the remaining manufactoring times. It is important to see that modifications in the estimates L are indirectly propagated backward through the production paths, because each value passed back to a predecessor influences the value which this predecessor will pass back. With that, the adjustment aims at "strenghening" ("weakening") chains of machines, or production paths, that contribute to a decrease (increase) in the overall manufacturing time. The Figure 2 illustrates the adjustment.

4. Experimental Results

This section presents experimental results on the time-constrained MMS scenario shown in the Table 1. The experimental setting is as follows. The parameters α and β are set to 1.0 and 0.3, respectively. Learning proceeds in trials, where a trial consists of the manufacturing of n products. At the beginning of the first trial all estimates L are set to zero. The Figures 3 and 4 give the learning curves together with two comparative curves for $n = 50$ and $n = 500$, respectively. Each data value shows the mean overall manufacturing time obtained during the previous 10 trials. The "random" curves resulted from a manufacturing process in which each machine chose its successor by chance. The "$L = 0$" curves were obtained by using the learning framework described above, but with the estimates $L_{i,j}^{i+1,k}$ (see Equation 1) set to a constant (zero). As the curves show, learning clearly leads to a considerable decrease in the overall manufacturing time compared to the "random" and "$L = 0$" approaches. As the Figure 3 (Figure 4) shows, averaged over all trials the learning approach leads to an average overall manufacturing time of about 510 (3505). Against that, the "$L = 0$" approach leads to a mean overall manufacturing time of about 620 (4705), and the random approach leads to a mean overall manufacturing time of about 780 (5570). (The shortest times measured during our experiments were 483 for $n = 50$ and 3364 for $n = 500$.) As the figures also show, learning converges rapidly, and after about 50 trials the curves become flat and stable.

An important question is how the learning approach reacts to environmental changes. The Figures 5 and 6 show how the curves change in response to a breakdown of the machines $M_{1,1}$, $M_{2,1}$, $M_{3,1}$, $M_{4,1}$, and $M_{5,1}$ (hence, of a complete production path) after 50 trials. The learning approach had no problems in adapting to the new situation, and again produced the best results compared to the other two approaches. After less than 50 additional trials, the learning curves converged to 575 (50 products) and 4460 (500 products). The Figures 7 and 8 show what happens if the five defect machines again are available after a period

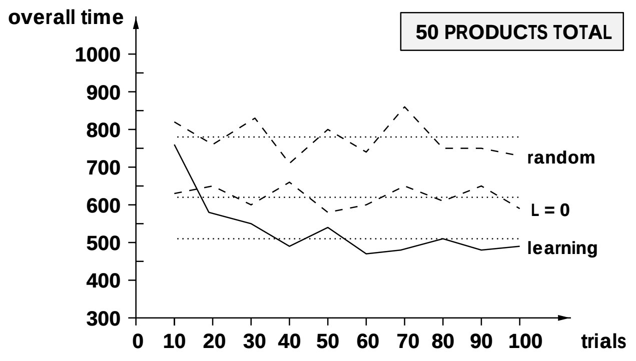

Figure 3. Learning and comparative curves for 50 products under stable conditions.

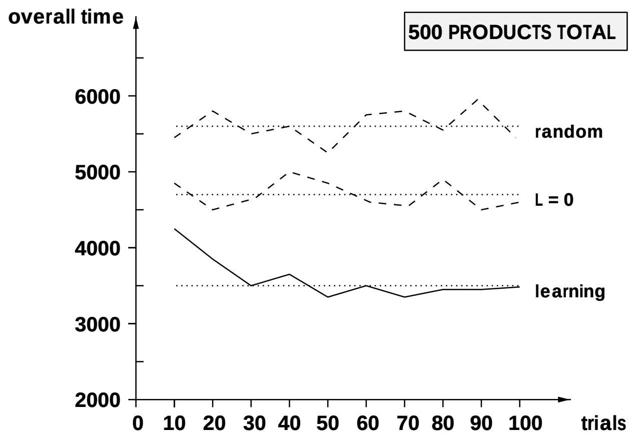

Figure 4. Learning and comparative curves for 500 products under stable conditions.

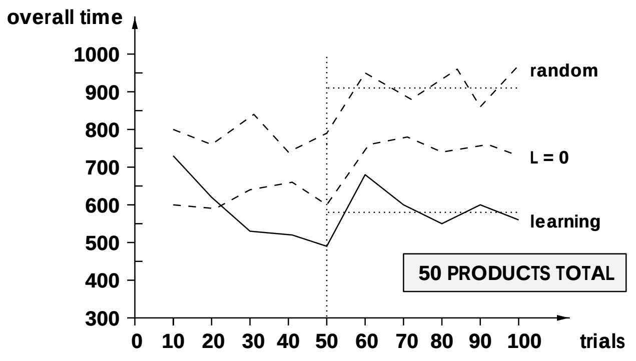

Figure 5. Learning and comparative curves for 50 products in response to a machine breakdown after 50 trials.

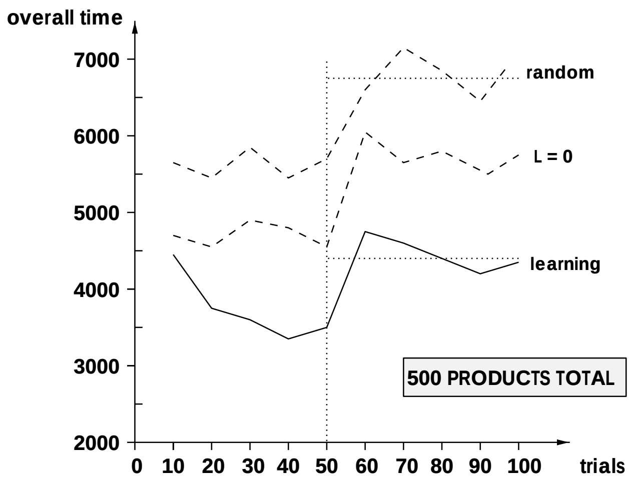

Figure 6. Learning and comparative curves for 500 products in response to a machine breakdown after 50 trials.

of 50 trials. Re-learning did not adapt to the new situation as fast as the comparative approaches (measured in relative changes), but after about 50 additional trials the learning curves reached their initial level.

A number of experiments with further time-constrained MMS scenarios have been conducted, varying the number of production steps and the number of machines involved in the different steps [9]. The results for these scenarios qualitatively coincide with those described here.

5. Discussion and Outlook

This paper described an application of the multi-agent learning paradigm known from Distributed AI to the problem of multi-agent scheduling. The major distinctive characteristic of the proposed learning approach is its inherent parallelism and distributedness w.r.t. the two learning steps – successor selection and estimate adjustment – repeatedly executed by the individual machines. Apart from that, this approach has the following characteristic features:

- Scheduling is realized as an experience-driven and reactive process in which the machines iteratively refine a complete schedule.

- The individual machines act as autonomous entities, although they influence each other in their activities through the exchange of information about estimates and waiting times. This information exchange allows a basic coordination at the machine level and results in a useful overall behavior at the system level.

- Only minimal demands are made on the abilities and the knowledge of the individual machines. Additionally, no specific demands (like “guaranteed delivery time”) on the manufacturing environment are made.

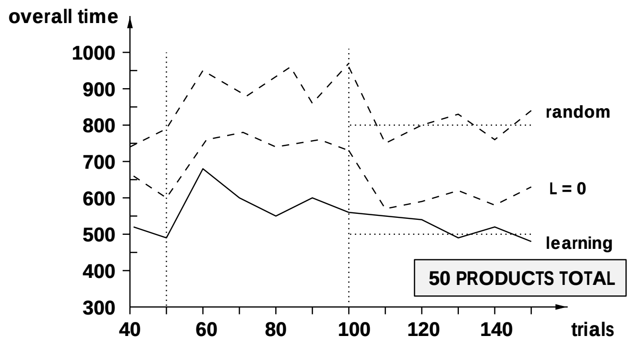

Figure 7. Learning and comparative curves for 50 products in response to machine recovery after trial 100.

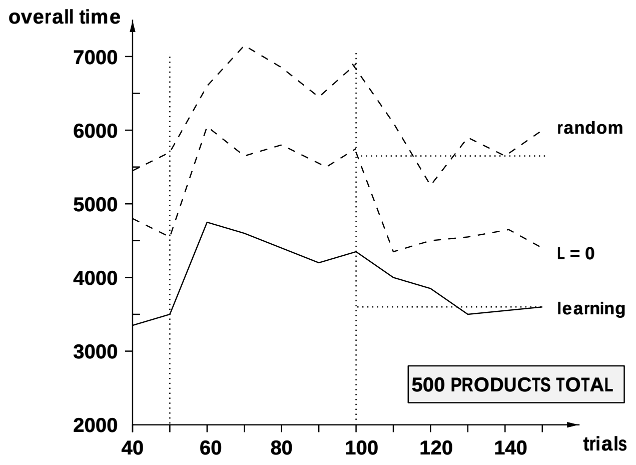

Figure 8. Learning and comparative curves for 500 products in response to a machine recovery after trial 100.

These features make the approach rather flexible, and applicable to a broad range of scheduling problems.

The initial experimental results show that very good schedules can be obtained in short time intervals, and they indicate that it is worth to continue this research. Based on the experience and insight gained so far, the following research activities are planned:

- Further comparative experiments that allow to contrast the learning approach with other, existing approaches to MMS.
- Further experiments with different and more complex scheduling scenarios in order to get a better understanding of the range of applicability as well as the robustness of this approach. Ideally these experiments are conducted in real-world domains like multi-processor scheduling or industrial workflow optimization.
- Bottom-up extension of the learning approach towards advanced cognitive abilities like (distributed) planning and negotiation as they are investigated in the field of multi-agent systems and Distributed AI.

These directions are interrelated and should be pursued in parallel: comparative studies should be also done in more complex scenarios; and the more complex a scenario is, the higher is the need for advanced abilities.

The past years have witnessed an increasing interest in multi-agent learning. The reason for this is mainly based on the insight that multi-agent systems, or Distributed AI systems in general, typically are quite complex and hard to specify in their behavioral dynamics. These systems therefore should be equipped with the ability to self-improve their performance. One of the central challenges in the field of multi-agent learning is to apply the principles and concepts of this kind of learning in real-world contexts. The goal of the research described in this paper is to meet this challenge.

Acknowledgements. The authors would like to thank the anonymous referees for their valuable comments. Gerhard Weiß has been financially supported by DFG (German National Science Foundation) under grant 1718/6-2.

References

[1] J. Blazewicz, K.H. Ecker, G. Schmidt, and J. Weglarz. *Scheduling in computer and manufacturing systems*. Springer-Verlag. 1994.

[2] T.C.E. Cheng and C.C.S. Sin. A state-of-the-art review of parallel machine scheduling research. *Operational Research*, 77:271-292, 1990.

[3] D.G. Feitelson and L. Rudolph (Eds.). *Job scheduling strategies for parallel processing*. Springer-Verlag, 1995.

[4] K. Fischer, J.P. Müller, M. Pischel, and D. Schier. A model for cooperative transportation scheduling. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS95)*, 109–116, 1995.

[5] M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. Freemann, 1979.

[6] I.F. Imam (Ed.). *Intelligent adaptive agents.* Papers from the 1996 AAAI workshop. Technical Report WS-96-04. AAAI Press, 1996.

[7] J.S. Liu and K.P. Sycara. Multiagent coordination in tightly coupled task scheduling. In *Proceedings of the Second International Conference on Multiagent Systems (ICMAS96)*, 181–188, 1995.

[8] R. Kozierok and P. Maes. A learning interface agent for scheduling meetings. In *Proceedings of the ACM SIGGHI International Workshop on Intelligent User Interfaces*, 81–88, 1993.

[9] M. Peceny, G. Weiß, and W. Brauer. *Verteiltes maschinelles Lernen in Fertigungsumgebungen*. Technical Report FKI-218-96. Institut für Informatik, Technische Universität München. 1996.

[10] T.E. Morton and D.W. Pentico. *Heuristic scheduling systems: With applications to production systems and project management*. Wiley, 1993.

[11] J.F. Muth and G.L. Thompson. *Industrial scheduling*. Englewood Cliffs, 1963.

[12] A.H.G. Rinnooy Kan. *Machine scheduling problems. Classification, complexity and computations*. Martinus Nijhoff/The Hague, 1976.

[13] S. Sen (Ed.). *Adaptation, co-evolution and learning in multiagent systems*. Papers from the 1996 AAAI symposium. AAAI Press, 1996.

[14] B.A. Shirazi, A.R. Hurson, and K.M. Kavin (Eds.). *Scheduling and load balancing in parallel and distributed systems*. IEEE Computer Society Press, 1995.

[15] K.P. Sycara, S.F. Roth, N. Sadeh, and M. Fox. Resource allocation in distributed factory scheduling. *IEEE Expert*, 6(1):29–40, 1991.

[16] C. Watkins and P. Dyan. Q-Learning. *Machine Learning*, 8:279-292. 1992.

[17] G. Weiß (Ed.). *Distributed artificial intelligence meets machine learning*. Lecture Notes in Artificial Intelligence, Vol. 1221. Springer-Verlag, 1997.

[18] G. Weiß and S. Sen (Eds.). *Adaption and learning in multi-agent systems*. Lecture Notes in Artificial Intelligence, Vol. 1042. Springer-Verlag, 1996.

[19] M. Zweben and M.S. Fox (Eds.). *Intelligent scheduling*. Morgan Kaufmann Pubishers, 1994.