Simulation supported agent-based adaptive production scheduling

Balázs Cs. Csáji[1], Botond Kádár[1], László Monostori[1,2], András Pfeiffer[1]

[1]*Computer and Automation Research Institute, Hungarian Academy of Sciences*
Kende u. 13-17, Budapest, H-1111, Hungary, Phone: (36 1) 297-6115, Fax: (36 1) 4667-503
e-mail: csaji@sztaki.hu, kadar@sztaki.hu

[2]*Department of Production Informatics, Management and Control, Fac. of Mechanical Engineering*
Budapest University of Technology and Economics, Budapest, Hungary
e-mail: laszlo.monostori@sztaki.hu

Abstract

Distributed (agent-based) control architectures offer prospects of reduced complexity, high flexibility and a high robustness against disturbances in manufacturing. However, it has also turned out that distributed control architectures, usually banning all forms of hierarchy, cannot guarantee optimum performance and the system behaviour can be unpredictable. The paper addresses the area of agent-based manufacturing systems, particularly it is devoted to distributed intelligent techniques for managing complexity, changes and disturbances on shop floor control level. More precisely, the paper outlines an attempt to enhance the performance of a market-based distributed manufacturing system by using reinforcement learning. In order to enable a constructive, decision supporting environment the mentioned techniques are integrated in a discrete event simulation framework. The experimental results demonstrate the applicability of the proposed solutions, which can contribute to significant improvements in system performance, keeping the known benefits of distributed control.

Keywords

Agent-based scheduling, reinforcement learning, simulation

1 Introduction

In manufacturing systems of our days, difficulties arise from unexpected tasks and events, non-linearities, and a multitude of interactions while attempting to control various activities in dynamic shop floors. Complexity and uncertainty seriously limit the effectiveness of conventional control and (predictive) scheduling approaches. Multi-agent based (distributed) control architectures offer prospects of reduced complexity, high flexibility and a high robustness against disturbances. The paper outlines an attempt to enhance the performance of a market-based distributed manufacturing system by using adaptation and machine learning techniques.

Dynamic, open real-world environments call for adaptive and learning systems that are equipped with processes that allow them to modify their behaviour, as needed [Monostori et al., 1996], [Kádár and Monostori, 2001]. Here we concentrate on distributed, agent-based approaches. Distributed manufacturing systems with agent-based control have a number of inherent advantages, such as modularity, reconfigurability, adaptability, fault tolerance, extensibility, etc. The elements of architectures, like this, however, are distributed, usually have no access to global information and, therefore, global optima cannot be guaranteed [Csáji et al., 2003]. Empowering the agents with learning skills creates the opportunity to improve the performance of the whole system [Kádár et al., 2003].

The fundamental aim of the paper is to outline the importance of adaptation and learning abilities in distributed manufacturing systems. A novel concept for adaptation and learning in multi-agent production control is suggested and both centralised and decentralised learning approaches are demonstrated. Special emphasis is given on a neurodynamic-based solution

with a three-level learning structure. The new prototype of the scheduler is integrated in a simulation environment which emulates a real manufacturing system. Unexpected events in the simulation will trigger rescheduling processes. The two integrated modules, the agent-based scheduler and the simulation, create a dynamic scheduling system which can be applied parallel to the shop-floor. The results of experimental runs illustrated at the end of the paper shows the applicability of the proposed approach.

2 Multi-agent manufacturing control

In order to overcome the drawbacks associated with hierarchical control, several researchers, e.g. [Hatvany, 1985] and [Duffie and Pipper, 1986], proposed the *heterarchical approach*. Heterarchical control is a highly distributed form of control, implemented by a system of independent co-operating processes or agents without centralised or explicit direct control. Control decisions are reached through mutual agreement and information is exchanged freely among the participating agents. The heterarchical control architecture is characterised by a flat structure that divides control responsibilities among co-operating controllers.

Since factories are inherently distributed, the need for distributed decision making arose quite naturally. *Distribute artificial intelligence (DAI)* architectures enable a system to deal with the right information at the time and place needed. According to [Bussmann, 1998)] "multi-agent systems can be best characterised as a software technology that is able to model and implement individual and social behaviour in distributed systems". Multi-agent technology has been considered as an important approach for developing distributed intelligent manufacturing system. Often these complexes are named *multi-agent manufacturing systems*.

In these manufacturing systems agents can represent manufacturing resources such as cells, machines, workers, parts, transportation units and/or operations. They can be also applied to model the functional components of a manufacturing system such as a scheduler or dispatcher. Consequently, they must also be capable of interpreting perceptions, reasoning, and choosing actions autonomously and in ways suited to achieving their intended goals. In order to operate reliably in unknown, partially known, and dynamic environments, they must also possess mechanisms to learn and adapt their interactions with the environments. In some applications they are required to be mobile and move or access different places or parts of their operating environments. Moreover, agents may be expected to be persistent, rational, etc., and in order to work in groups, able to communicate and collaborate. The paper concentrates on distributed, agent-based approaches and suggests learning and adaptive agents.

A relative novel approach for co-ordination in multi-agent systems is stigmergy that belongs to mechanisms which mimic animal-animal interactions. Stigmergy is an indirect co-ordination tool within an insect society where parts of global information is made available locally by pheromones, e.g., in the case of ant colonies [Valckenaers et al., 2001]. This way, individual ants are not exposed to the complexity and dynamics of the situation, and the communication burden in the computer realisation is significantly lower, compared to market-based solutions.

Micro- and macro-level interactions in multi-agent manufacturing systems are distinguished in [Bochmann et al., 2003]. The micro-level interactions are for the co-ordination of a small number of agents to solve particular sub-problems, e.g., the modelling of the behaviour of other agents by Bayesian networks. The macro-level interactions co-ordinate partial solutions from the micro level to ensure that the individual decision making of the agents is synergistic rather than destructive.

3 Learning agents

Throughout the paper the general job-shop scheduling (JSP) problem is formulated as follows: we have n jobs $J = \{J_1, J_2, ..., J_n\}$ to be processed through m machines $M = \{M_1, M_2, ..., M_m\}$. The processing of a job on a machine is called operation or task. The operations are non-preemptive (they may not be interrupted) and each machine can process at most one operation at a time (capacity constraint). Each job may be processed by at most one machine at a time (disjunctive constraint). O denotes the set of all operations. Every job has a set of operation sequences: the possible process plans, $o: J \rightarrow P(O^*)$. These sequences give the precedence constraints of the operations. The processing time of an operation on a machine is given by p: $M \times O \rightarrow R^+$ which is a partial function. Due dates are given for every job: d: $J \rightarrow R^+$, $d(J_i)$ is the time by which we would like to have J_i completed ideally. Objectives in scheduling are complex, and often conflicting. The objective is to produce a schedule that minimises (or maximises) a performance measure f of regular type, which is usually a function of job completion times (e.g.: maximum completion time, mean flow time, mean tardiness, number of tardy jobs, etc.), i.e., JSP is an optimisation problem. Except for some strongly restricted special cases, JSP is an NP-hard optimisation problem, i.e., no polynomial time algorithm exists that always provides us the exact optimal schedule, unless P=NP [Williamson et al., 1997]. Moreover, there is no good polynomial time approximation of the optimal scheduling algorithm.

The paper suggests a solution in which the agents should not investigate every potential schedule, because it is extremely time consuming. If an agent wants to bid for an operation sequence and it needs information about the production costs of that part of the job, which it cannot make, it should not announce it for every resource agent. It should make only a restricted tendering among those agents that will give a presumably good bid. In the suggested solution the agents use reinforcement learning, such as TD(λ), to learn for every operation sequence the presumably good bidders in a given time. Every agent learns independently. The states of the used reinforcement learning for deciding which action is to be taken is an operation sequence with its earliest start time. An action is the announcement of the sequence to a resource agent whose machine can do the next operation of the job. The rewards are computed from the given bids of the invited agents.

4 The prototype of the agent-based scheduler

In the developed system every job is managed by a dedicated order agent. The first investigated level constitutes the level of *temporal difference learning*. A state s of the reinforcement learning is the remaining operation sequence of the job in question, at a given time. The Markov property [Sutton and Barto, 1998] can be considered fulfilled. In every stage of the operation sequence of a given job, mobile agents visit (thin black arrows, Figure 1) prospective resources where other mobile agents are instantiated to find the next appropriate and available resource, etc., until reaching the system output. Naturally, because of complexity problems, all of the prospective resources cannot be visited. The possible actions are the selection of the resources appropriate for the next task. The approximation of the value function is realised by *artificial neural networks (ANNs)* stored locally in the resource agents. The bottom level of learning proceeds by these ANNs.

A mobile agent continues its way from one resource agent to another with $\pi(s, a)$ probability, where s is a state and a is an action. The randomisation of the routing is not only needed for exploration but it also helps us to avoid pathologic behaviour. Every $\pi(s, a)$ is treated as independent probability and it is possible to take two or more actions (called branching) in a state. Thus, several schedules can be parallel explored. From the approximated value function, which stores the performance measure estimations, the sending probabilities of the

mobile agents are computed with a Boltzmann formula. The original formula was modified to let the system have higher (than one) expected branching factor (1). Suppose that an estimation is available for the resource agent r for the expected return value if it sends a job with the remaining task sequence γ to the resource agent p at time t. Let us denote that by $V_p(\gamma, t)$, and the branching factor by β. If our aim is to maximise the achieved return values, the probability of the action a can be computed, which is the sending of the mobile agent to the resource agent p with the following formula:

$$\pi(s,a) = \min\left\{1,\ \left(\beta \cdot e^{V_p(\gamma,t)/\tau}\right) \Big/ \left(\sum_q e^{V_q(\gamma,t)/\tau}\right)\right\}, \quad (1)$$

where τ is the so-called Boltzmann temperature. High values cause the actions to be (nearly) equiprobable, low ones cause a greater difference in selection probability for actions that differ in their value estimations.

When a mobile agent virtually investigated a schedule (its remaining operation sequence is empty), it starts to move backwards and supports feedback information. The rewards for the *TD*(λ) RL mechanism are computed from the achieved performance measures. If agents return to a resource agent where there was previously a branching, than only the mobile agent with the best schedule continues its travelling backwards (bold black arrows, Figure 1), the others terminate. The resource agent uses the reward brought by the mobile agent to learn the optimal scheduling routes of the current system. Similarly to [Hadeli et al., 2004], it is assumed that the agents work much faster than the ironware they control, consequently, they can repeat this process several times before the system changes.

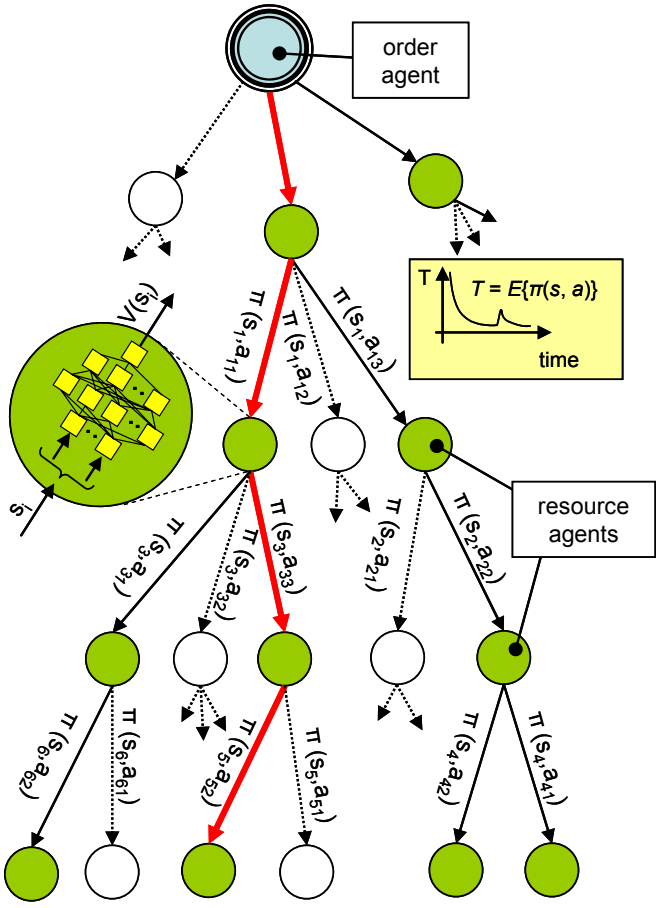

Figure 1: Outline of the neurodynamic multi-agent scheduling system

The top level of adaptation refers to the setting of the branching factor of mobile agents. A *simulated annealing* mechanism [Kirkpatrick et al., 1983] controls the "temperature" T of the system, which is the expected branching number of agents at a resource agent $T = E\{\pi(s, a)\}$, note that it is not necessarily an integer. If the system changes, T is raised to force the system to explore the new situation. If the system stabilises, it will be cooled down by lowering T to let the agents exploit the information they have gathered. Note that the Boltzmann temperature can be changed as well to let the system do more explorations or exploitations.

5 The integration of the agent-based scheduler and the simulation

Simulation is a powerful tool that is often applied to the design and analysis of complex systems. Decisions can be made about the system by constructing computer models of it and conducting experiments on the model. For constructing valid models of complex systems (e.g. manufacturing, transport, service systems etc.) and their processes the models should represent the discrete event evolution of the system, as well as features of the underlying continuous processes. The execution of a simulation study is a cyclical and evolutionary process. The first draft of the model will frequently be altered to make use of in-between results and in general the final model can only be achieved after several cycles.

The features provided by the new generation of simulation software facilitate the integration of these tools with the production planning and scheduling systems. Additionally, if the simulation system is combined with the production database of the enterprise it is possible to instantly update the parameters in the model and use the simulation parallel to the real manufacturing system supporting and/or reinforcing the decisions on the shop-floor.

In the proposed architecture the simulation model replaces a real production environment, including both the *manufacturing execution system (MES)* and the model of the real factory. It creates the uncertain environment for the scheduling and rescheduling actions capturing those relevant aspects of the problem that cannot be represented in a deterministic optimisation model.

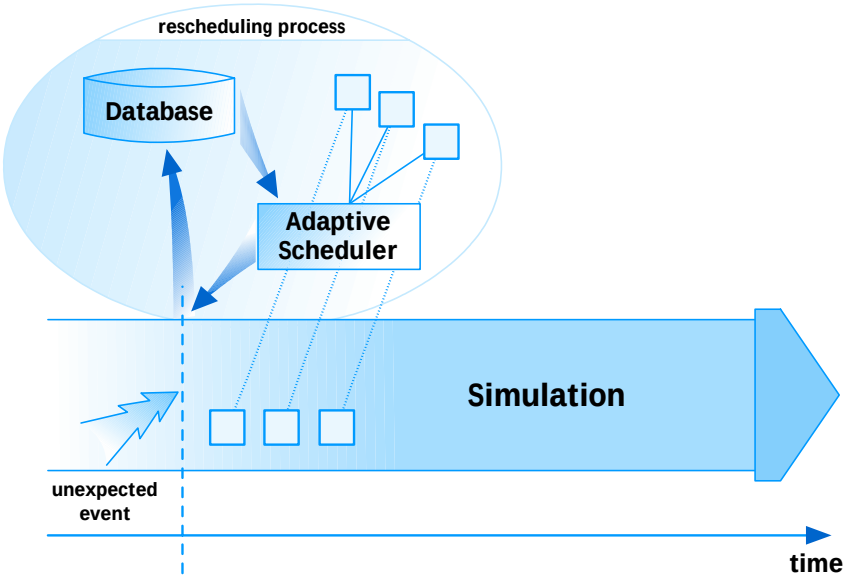

Figure 2. The rescheduling process initiated form the simulation side

The outline of the developed architecture is presented in Figure 2. The simulation model is coupled with the production database. On the base of the resources table, the whole model is generated automatically during the data preparation phase. This is combined with the weekly

calendar of the resources. The simulation model also offers a user interface through which the execution of the model can be monitored. Rescheduling action can be initiated in the case of an unexpected event occurs or if a main performance measure bypasses a permissible threshold.

6 Experimental results

The performance of the agent-based scheduler presented in the previous section was tested in conjunction of the simulation model. The results of the evaluation are presented in the following section.

6.1 Comparison of the adaptive algorithm with the optimal solution

In order to verify the adaptive algorithm, experiments were initiated and carried out. As a performance measure, the minimisation of the maximum completion time (C_{max}) was selected, however, without exploiting its special properties, in order to test the performance of the approach for an arbitrary regular measure. First, an advance schedule was generated in order to start with a non-empty schedule, by using random orders and simple dispatching rules. Then, the performance of the proposed solution was investigated, i.e., for finding good (or optimal) schedules with different parameter values.

The results were compared with other scheduling algorithms, such as branch and bound (B&B). The exploration / exploitation features of the algorithm were also tested. Since the proposed solution is a randomised algorithm, the results of several (usually a hundred) runtime outcomes were averaged (Table 1).

The number of operations in the new job									
	4	5	6	7	8	9	10	11	12
Steps of NDP(2.1)	$5{,}6 * 10^2$	$6{,}3 * 10^2$	$1{,}2 * 10^3$	$2{,}9 * 10^3$	$4{,}8 * 10^3$	$6{,}8 * 10^3$	$1{,}1 * 10^4$	$1{,}5 * 10^4$	$1{,}9 * 10^4$
Standard deviation	8,5	7,6	7,1	6,5	5,9	5,1	3,9	3,7	3,3
Steps of B&B	$4{,}9 * 10^2$	$1{,}2 * 10^3$	$5{,}5 * 10^3$	$1{,}5 * 10^4$	$5{,}5 * 10^4$	$2{,}1 * 10^5$	$6{,}5 * 10^5$	$2{,}6 * 10^6$	$1{,}2 * 10^7$
Ratio in %	113	49,5	22,4	18,8	8,6	3,1	1,6	0,5	0,1

Table 1: Computation costs as performance indicator

The proposed, neurodynamic-based approach is compared with the classical B&B algorithm in Table 1. It illustrates how much computation (measured in steps, i.e., virtually putting an operation on a machine) is required if a new job arrives at the system and it is scheduled without rescheduling the whole system. The B&B algorithm always finds the optimal schedule, however, asymptotically it needs much more computation than the distributed, neurodynamic-based solution. The NDP(X) row and the row below show the data achieved by using the proposed algorithm with branching factor X. It demonstrates how fast it could reach a solution in average, with a performance measure that differs at most 5% from the global optimum. The data were generated with 16 partially interchangeable machines, which were previously scheduled with jobs, and a random new job with 4 to 12 random operations. It is worth mentioning that reduction in computation costs grows rapidly with the complexity of the incoming new jobs.

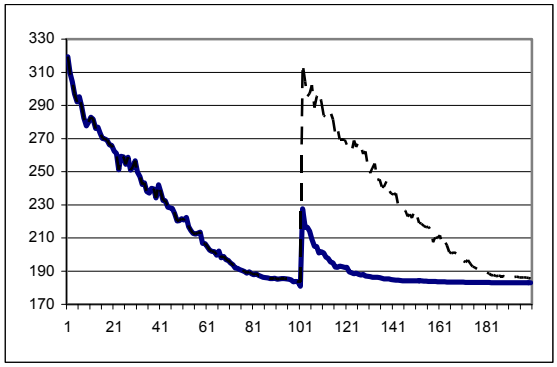

Figure 3. Machine breakdown *(t = 100)*

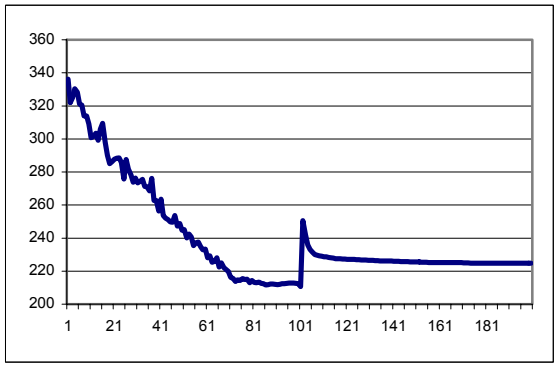

Figure 4. New job enters the system *(t = 100)*

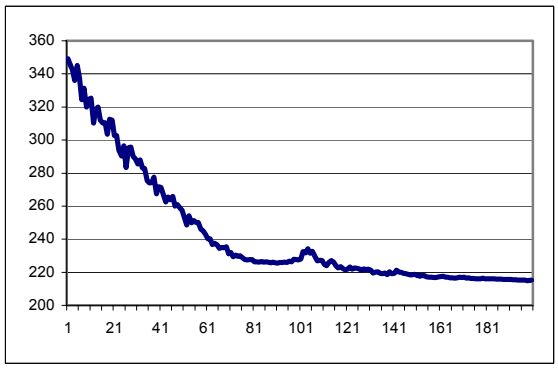

Figure 5. New machine is available *(t = 100)*

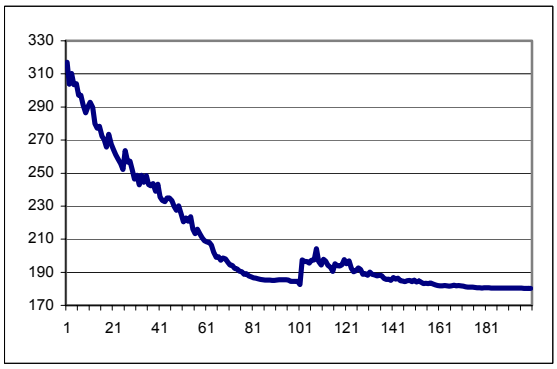

Figure 6. A job is cancelled *(t = 100)*

6.2 Test results of the dynamic-scheduling set-up

In this scenario the aim of scheduling was also to minimize the maximum completion time (Cmax). The adaptive features of our solution were tested by confronting it with unexpected events. In figure 3-6 there are four types of unexpected events: *machine breakdown*, *new machine, new job* and *job cancellation* respectively. The x-axis shows time while the y-axis shows the achieved performance measure (which is the total production time to be minimised). The figures presented here were made by averaging several (at least hundred) random runtime results. In these tests 20 machines with a few dozen of jobs were evaluated. At all cases at time $t = 100$ different unexpected events were initiated. The results show that the system is adaptive because it did not re-compute the whole schedule from scratch, but it tried to use as many information from the past as possible. The dashed line in figure 3 represents the performance measure which would arise in the case of a full recalculation of the whole schedule.

7 Conclusions

In the paper a market-based distributed production control system was described with learning and cooperative agents. Management of changes and disturbances and computational feasibility were regarded as key driving factors. In the proposed solution the learning is done by a three-level learning mechanism. The top level of learning consists of a simulated annealing algorithm, the middle (and the most important) level contained a reinforcement learning system, while the bottom level was done by numerical function approximators, like artificial neural networks.

The developed system can be used to solve general dynamic job-shop scheduling problems in a distributed, iterative and robust way. The agent-based scheduler was integrated in a simulation environment. The agents in the scheduler identify all the resources of the simulation model. The applicability of the integrated system was demonstrated by the results of experimental runs.

Acknowledgements

The research was supported partially by the project "Digital enterprises, production networks" in the frame of the National Research and Development Programme by the Ministry of Education (Grant No. 2/040/2001). A part of the work was covered by the National Research Fund, Hungary, Grant Nos. T034632 and T043547. Botond Kádár greatly acknowledges the support of Bolyai János Scholarship of the Hungarian Academy of Sciences significantly easing the contribution.

References

Bochmann, O., Van Brussel, H., Valckenaers, P., (2003), Micro- and Macro-Level Interactions in Multi-Agent Manufacturing Systems, Proc. of 36th CIRP ISMS, June 3-5, Saarbrücken, Germany pp: 61-67.

Bussmann, S. (1998), "An Agent-Oriented Architecture for Holonic Manufacturing Control", Proc. of 1st Int. Workshop on Intelligent Manufacturing Systems, EPFL, Lausanne, Switzerland, pp. 1 - 12.

Csáji, B.Cs.; Kádár, B.; Monostori, L. (2003), Improving multi-agent-based scheduling by neurodynamic programming, Lecture Notes in Computer Science; 2744: Lecture Notes in Artificial Intelligence, Holonic and Multi-Agent Systems for Manufacturing, Springer, pp. 110-123.

Duffie, N. A.; Piper, R. S., (1986), "Non-hierarchical Control of Manufacturing Systems", Journal of Manufacturing Systems, Vol. 5, No. 2.

Hadeli, Valckenaers, P., Kollingbaum, M., Van Brussel, H., (2004), Multi-Agent Coordination and Control Using Stigmergy, Computers in Industry, Vol 53, pp: 75-96.

Hatvany, J. (1985), "Intelligence and cooperation in heterarchic manufacturing systems", Robotics & Computer-Integrated Manufacturing, Vol. 2, No. 2, pp. 101-104.

Kádár, B.; Monostori, L. (2001), Approaches to increase the performance of agent-based production systems, Lecture Notes in Computer Science; 2070: Lecture Notes in Artificial Intelligence, Engineering of Intelligent Systems, Springer, pp. 612-621.

Kádár, B.; Monostori, L.; Csáji, B. (2003), Adaptive Approaches to Increase the Performance of Production Control Systems, Proc. of 36th CIRP ISMS, June 3-5, Saarbrücken, Germany pp:305-312.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983), Optimisation by Simulated Annealing, Science, 4598 pp: 671-681.

Monostori, L., Márkus, A., Van Brussel, H., Westkämper, E. (1996), Machine Learning Approaches to Manufacturing, Annals of the CIRP, Vol 45/2 pp: 675-712.

Sutton, R. S., Barto, A. G. (1998), Reinforcement Learning. The MIT Press.

Valckenaers, P., Van Brussel, H., Kollingbaum, M., and Bochmann O. (2001), Multi-Agent Coordination and Control Using Stigmergy Applied to Manufacturing Control. European Agent Systems Summer School, Prague, pp: 317-334.

Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevastjanov, S.V., Shmoys, D.B., (1997),Short Shop Schedules, Operations Research, Vol 45/2, pp: 288-294.