Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 79 (2019) 385–390

www.elsevier.com/locate/procedia

12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy

Coupling of centralized and decentralized scheduling for robust production in agile production systems

Fabio Echsler Minguillon[a,]*, Gisela Lanza[a]

[a] *wbk Institute of Production Science, Karlsruhe Institute of Technology (KIT), Kaiserstr. 12, 76131 Karlsruhe, Germany*

* Corresponding author. Tel.: +49-721-608-44153 *E-mail address:* fabio.echsler@kit.edu

Abstract

Individualized products and timely delivery require agile just-in-time manufacturing operations. Scheduling needs to deliver a robust performance with high and stable results even when facing disruptions such as machine failures. Existing approaches often generate predictive schedules and adjust them reactively as disturbances occur. However, the effectiveness of rescheduling approaches highly depends on the available degrees of freedom in the predictive schedule. In the proposed approach, a centralized robust scheduling procedure is coupled with a decentralized reinforcement learning algorithm in order to adjust the required degrees of freedom for a maximally efficient production control in real-time.

Keywords: Scheduling; Robustness; Reinforcement Learning; Agile Production

1. Introduction

Manufacturing today is facing an increasingly volatile and fast-paced environment. Product lifecycles are shortening while variant diversity is growing. Additionally, customer demand is more volatile and harder to predict than before [1]. As a result, agility is gaining importance as an evaluation factor of manufacturing operations [2].

Recent research efforts have been focused on the design of agile production systems such as matrix-structured manufacturing systems (MMS) [3]. These systems allow for agile production of many variants with a lot size of one. Key for efficient operation of such a production system is a suitable method for scheduling, i.e. assigning jobs to resources over time in order to meet some goal criterion [4]. The scheduling environment of an MMS corresponds to a flexible job shop scheduling problem (FJSSP) [4]. In make-to-order settings, jobs often have to meet due dates e.g. as to adhere to just-in-sequence (JIS) requirements [5]. In order to calculate the completion dates for performance evaluation, all information about jobs and resources involved is gathered centrally and scheduled with respect to global optimality, albeit it with long computation times [6]. However, unforeseen disruptions on the shop floor such as machine failures can deteriorate the performance of central schedules. Therefore, a reactive rescheduling method is needed in order to handle disruptions appropriately and maintain the performance level. In order to be able to react in a timely manner, local information is often used decentrally and rescheduling is performed with respect to local optimality in short computation times [6].

In this paper, an approach for scheduling and rescheduling for FJSSP settings found in MMS is presented. The aim of the approach is to deliver a robust performance with respect to the JIS criterion [7]. The approach consists of a centralized robust scheduling system, a decentralized reinforcement learning system for partial rescheduling and a coupling mechanism for adjusting the robustness requirement in centralized planning according to learning progress achieved in decentralized rescheduling.

2. State of the art

2.1. Dynamic scheduling

As most manufacturing systems operate in an uncertain environment, disruptions such as machine failures can render predictively created schedules infeasible. Scheduling therefore needs to also consider the presence of real-time events. This is defined as dynamic scheduling [8]. Approaches for solving dynamic scheduling problems are categorized as completely reactive, predictive-reactive, and robust pro-active [9].

- In completely reactive scheduling, no predetermined schedule is created. Only local dispatching is performed and subsequent machines are selected for processing jobs only after they have finished processing on a previous machine. These approaches inherently consider the current situation of the job shop. However, they are unsuitable for JIS settings because of the lack of a predetermined schedule. [8]
- Predictive-reactive scheduling approaches generate a predictive schedule off-line (before production has started) and correct that schedule on-line (during production) if necessary due to unforeseen disruptions. Predictive schedules are usually created centralized with all information available, while rescheduling can be performed decentralized with only local information available. This can decrease calculation times considerably. Rescheduling can either be performed as right-shift rescheduling (shifting operations in time as necessary), partial rescheduling (regenerating part of the predictive schedule) or complete regeneration (regenerate the entire remaining schedule) [9]. Predictive-reactive scheduling can be used for both adhering to JIS requirements (with predictive scheduling) as well as maintaining this plan in the face of disruptions (with reactive rescheduling). [8]
- Robust pro-active scheduling seeks to consider all uncertainties e.g. with respect to station availabilities during initial scheduling. No rescheduling procedure is needed in this case. However, this considers a worst-case approach and can yield poor performance in case of less than expected disturbances. [8]

2.2. Robustness and its application to scheduling

Robustness in the context of manufacturing can be defined as the ability to maintain a high performance even when facing unknown disruptions [10]. In order to evaluate the robustness in planning, the most commonly utilized criteria described in [11] are:

- Solution robustness: The result of a plan should not deviate from the planned outcome under different disruptions.
- Optimality robustness: In addition to a solution robust plan, the result of the plan should not deviate from the optimal outcome, i.e. represent a "perfect plan".
- Feasibility robustness: Disruptions should not have any effect on planned tasks.

For further details on information robustness, planning robustness and evaluation robustness, the reader is referred to [11]. In scheduling literature, solution robustness and optimality robustness are often summarized under the term robustness, whereas feasibility robustness is described as stability [12]. It is desirable to create both robust and stable plans, however this is not possible in most cases.

Robustness and stability criteria are operationalized in robust optimization (RO) models. An approach for RO is presented in [13]. Instead of optimizing a certain goal criterion (e.g. makespan, earliness and tardiness) directly, a robustness measure is introduced in the objective function and optimized instead. By integrating the original goal criterion into the robustness measure, robust and good solutions (with respect to the original goal criterion) can be obtained. This is demonstrated in the following example:

A disjunctive linear program (LP) formulation of the classical job shop scheduling problem (JSSP) with makespan optimization [14] can be written as:

$$
\begin{array}{lll}
\min & C_{\max}(S_p) & \\
s.t. & y_{h,j} - y_{i,j} \ge p_{i,j} & (i,j) \to (h,j) \in A \\
 & C_{\max} - y_{i,h} \ge p_{i,j} & (i,j) \in N \\
 & y_{i,j} - y_{i,h} \ge p_{i,h} \text{ or} & (i,h),(i,j), i = 1,\dots,m \\
 & y_{i,j} - y_{i,h} \ge p_{i,k} & \\
 & y_{i,j} - y_{i,h} \ge 0 & (i,j) \in N \\
 & C_{\max}(S_p) \ge 0 & (i,j) \in N
\end{array}
\tag{1}
$$

The formulation as a RO problem based on [13] is as follows:

$$
\begin{array}{lll}
\min & \upsilon & \\
s.t. & C_{\max}(S_p) \le \upsilon & \\
 & y_{h,j} - y_{i,j} \ge p_{i,j} & (i,j) \to (h,j) \in A \\
 & C_{\max} - y_{i,h} \ge p_{i,j} & (i,j) \in N \\
 & y_{i,j} - y_{i,h} \ge p_{i,h} \text{ or} & (i,h),(i,j), i = 1,\dots,m \\
 & y_{i,j} - y_{i,h} \ge p_{i,k} & \\
 & y_{i,j} - y_{i,h} \ge 0 & (i,j) \in N \\
 & C_{\max}(S_p) \ge 0 & (i,j) \in N
\end{array}
\tag{2}
$$

The main difference between both models lies in the way the optimization criterion is integrated. In the first case, makespan $C_{\max}$ is to be minimized, in the second case, a robustness criterion υ is to be minimized. In the example, the robustness criterion is a maximum makespan that is to not be exceeded by any schedule. However, this is not necessarily so.

The result largely depends on the robustness criterion chosen. Extensive research has been done on robustness measures for scheduling problems. Robustness measures can be categorized in two different classes:

- Scenario-based measures evaluate robustness as the deviation of the planned schedule from a realized schedule over scenarios. A low deviation indicates a high robustness, see e.g. [12], [15], [16], [17]. Computational studies have shown that scenario-based measures are capable of increasing robustness [16]. However, they are computationally intensive, as many scenarios need to be created and evaluated. This quickly becomes an intractable problem for many practical applications.
- Surrogate measures evaluate robustness as the approximated deviation of the planned schedule with mostly slack-based scenario-independent indicators, see e.g. [16], [18], [19], [20]. The amount and distribution of idle times (slack) in a schedule is used in many surrogate measures.

2.3. Reinforcement learning in scheduling

Reinforcement learning (RL) is described as "learning what to do […] so as to maximize a numerical reward signal" [21]. It describes both a problem and a class of solution methods within the domain of artificial intelligence. RL has found numerous applications within the domain multi-agent systems (MAS) in recent years. MAS consist of at least two agents deciding on some task. [22] Usually, a larger task (e.g. scheduling an entire shift) is decomposed into smaller decision problems that are then solved decentrally. Besides many rule-based approaches for negotiation (e.g. Contract Net Protocol [23]), RL has become one of the most widely applied methods from artificial intelligence in MAS. A thorough survey of RL in MAS can be found in [24]. For the purpose of this paper, only RL approaches in MAS with applications to scheduling are considered. The approaches transform scheduling into a sequential decision problem which is modeled with the help of decentralized Markov decision processes (DEC-MDP) [25].

A DEC-MDP is defined by a tuple $\langle Ag, S, A, P, R, \Omega, O \rangle$ [26] with

- $Ag = \{1,...,m\}$ as the set of agents; in a scheduling settings, these are m stations deciding on which of the waiting jobs to process next
- S as world states that can be factored into m individual components $S = S_1 \times ... \times S_m$; in scheduling, these are e.g. which stations are available, which jobs are in the system and what is their status etc.
- $A = A_1 \times ... \times A_m$ as a set of joint actions performed by the m agents; between two time steps this would be all jobs that have been picked for further processing

- P as a transition function that shows how the system state changes in response to an action; this could entail finished jobs or jobs whose state has been changed
- R as a reward function for executing a certain action in a certain state; in scheduling, the makespan is to be minimized often. A reward function for this case could reward every action by -1 for each time step required [25].

In [26], a decentralised policy search algorithm is proposed building upon the DEC-MDP formulation for scheduling. Policy updates are performed with a gradient estimation approach. After enough training examples have been collected, the policies of the agents are updated according to the gradient estimation and therefore gradually adjusted to a locally optimal policy. This is performed for all agents individually, enabling decentralised learning of an implicit coordination to solve a scheduling problem.

2.4. Summary and deficit

RO can be useful for generating robust schedules and is a feasible method for centralized scheduling of FJSSP as found in MMS. However, special care should be devoted to suitable robustness measures, as the obtained results largely depend on those measures. Surrogate measure are a promising and on-going research effort, as they require relatively little computational power and can deliver very good results.

RL within MAS represent a useful framework for both generating and regenerating schedules. Due to its relatively low computational requirements and strong decentralization of the scheduling problem, it represents a natural approach for rescheduling.

From a robustness and stability perspective, schedule repair with a partial rescheduling approach is desirable, as deviations from the centralized schedule are minimized. However, to the author's knowledge, no existing approaches explicitly treat the robustness required to even be able to reschedule within a certain time successfully as a variable. This is even more interesting for the relatively new applications of RL in scheduling have the potential to adapt to agile production systems like MMS.

3. Approach

The proposed approach for coupling of centralized and decentralized scheduling consists of three distinct systems that are connected within a loop.

The centralized scheduling with RO seeks to incorporate some knowledge about possible machine failures into schedule generation. The goal is to predictively calculate completion times for jobs within the bounds of the due dates as to allow for JIS production. The decentralized rescheduling with RL becomes active any time infeasibilities arise in the centrally created schedule. Its task is to then perform partial schedule regeneration within a user-defined time frame. The coupling mechanism incorporates scheduling knowledge acquired through repeated decentralized rescheduling as a lower robustness requirement for centralized scheduling. As the reactivity to disruptions becomes better over time, less robustness is required in centralized planning, leading to a generally better performance with respect to the JIS criterion. However, depending on the specific setup, at some point an equilibrium for the robustness requirement is reached. Without any robustness, no rescheduling within a certain time frame is possible. In other words, some form of slack is always required in the schedule or else rescheduling with partial regeneration is not possible.

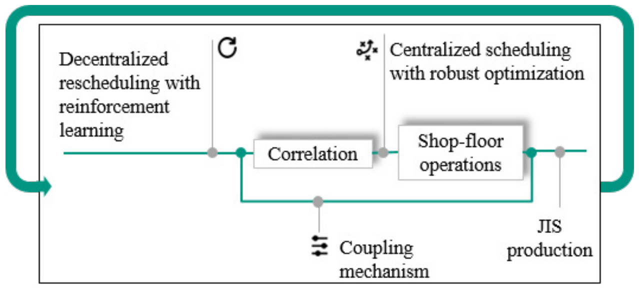
Decentralized rescheduling with reinforcement learning
Centralized scheduling with robust optimization
Correlation
Shop-floor operations
Coupling mechanism
JIS production

Fig. 1: Overview of the approach.

The approach is embedded in a discrete event simulation of the production system whose production control is to be optimized. It has to be trained separately for each new MMS, because the following system-specific factors have a great impact on its performance:

- Redundant stations: In MMS, identical stations are often multiplied e.g. in order to provide the capacity needed. Redundant stations provide flexibility for decentralized rescheduling, as they potentially allow for re-routing of jobs in case of machine failures.
- Station availabilities: MMS with very high station availabilities are less prone to disruptions caused by machine failures. In such settings, centralized scheduling is close to the JIS criterion in deterministic environments.
- Buffer size: Robustness can be created with work-in-process (WIP). By providing enough WIP at each station, the system is much less prone to starvation and can continue production when facing disruptions.
- Production programs: A tight production program can lead to a very high station utilization. In cases where all stations are utilized near their maximum capacity, rescheduling cannot be performed decentralized, because cascading effects inevitably require complete regeneration.
- Precedence graphs of products: Whether products have flexibility in their precedence graphs or not plays an important role in the ability to reschedule. Besides using redundant stations for a task, switching the order of some tasks allows for additional options for rescheduling.
- Time frame for schedule regeneration: The smaller the time frame is, the more difficult it becomes to reschedule within those boundaries. Rescheduling required idle times for compensating disruptions. Along a time line, the amount of idle times on all concerned stations needs to be at least as much as the duration of the disruption. This is more likely to be fulfilled for longer time frames.

3.1. Centralized scheduling with RO

Centralized scheduling with RO generates robust predictive schedules with committed completion time. The goal is to provide a schedule with a large robustness without any impact on performance. For centralized scheduling, a RO model based on [13] is utilized with a standard formulation of the FJSSP. The robustness measure has been developed to focus solution robustness only. Optimality can rarely be obtained in large scheduling settings; feasibility does not need to be obtained with RO because of the coupling with decentralized rescheduling. Therefore, the utilized robustness measure has a focus on practical applicability and considers specific station availabilities. As disruptions are unknown, all jobs should behave equally robust. In JIS settings, robustness can be considered as the slack time between planned completion time and due date. This will be denoted as unweighted slack and can be defined as follows:

$$us_i = dd_i - pct_i \tag{3}$$

Unweighted slack for a job i us_i is the difference between that job's due date dd_i and its planned completion time pct_i A weighting factor corresponding to the station availabilities is added to acquire weighted slack ws_i :

$$ws_i = us_i \prod_{k=1}^{o_i} a_k \tag{4}$$

The weight is obtained from the availabilities a_k of all k utilized stations along its precedence graph. Jobs utilizing stations with a low availability therefore have a much lower weighted slack than jobs on stations with a high availability. This can be used in robust scheduling to ensure weighted slack is evenly distributed amongst all jobs to be scheduled. The robustness measure used is:

$$\upsilon = \min_{\forall i}\{ws_i\} \tag{5}$$

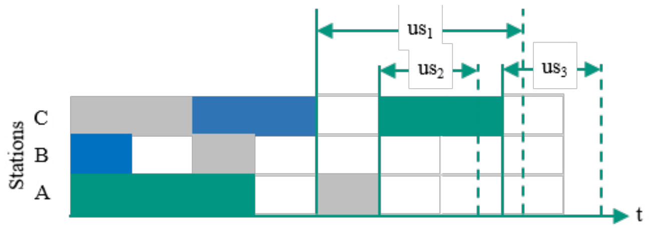

Fig. 2: Example for the slack-based robustness measure.

In Fig. 2, a simple example of a schedule with 3 jobs on 3 stations is shown. The us values correspond to the different unweighted slack values between completion time and due date. Weighted slack would have to incorporate the station availabilities of all used stations A, B, or C.

The optimization model used corresponds to the one outlined in [13] for FJSSP. Additionally, buffer constraints for physical buffers were added to model realistic scenario with WIP limits (organizational or physical limits).

In order to obtain the minimal robustness that can always be integrated in the central schedule without any performance deterioration, the following optimizations are performed:

1. Determine a predictive schedule with respect to the JIS criterion (i.e. making sure the deviation between

completion times and due dates are as low as possible, also: delivery reliability)

2. Add the achieved delivery reliability as an additional constraint to the RO model and determine a robust schedule with respect to the weighted slack robustness measure

This allows obtaining a maximally robust schedule that still allows for the best possible performance with respect to the JIS criterion. Requiring any additional robustness would inevitably lead to decreased performance. The robustness value obtained in 2. (r*) is the minimum robustness level possible to be manipulated further by the coupling mechanism.

3.2. Decentralized rescheduling with RL

Decentralized rescheduling with RL is used to regenerate a schedule partially after a disruption in order to continue with the robust schedule soon thereafter. The goal is to perform partial regeneration as fast as possible with as little robustness in the central plan as possible. A rescheduling strategy needs the following information to be able to reschedule:

- When should rescheduling take place? In this case, only machine failures trigger rescheduling.
- How long should rescheduling be performed? Here, this is a user parameter setting an upper bound.
- What should be rescheduled? All tasks belonging to jobs that were originally scheduled within the determined time.

Rescheduling is performed with the RL approach outlined in [26]. Stations are modeled as agents and decide independently which job to process next. Their reward function awards every agent a reward of -1 for each time step. This way, makespan for the tasks to be rescheduled is optimized and return to the robust schedule in less time than specified by the user is possible. The policy update function updates the agent's preferences over time and generates scheduling knowledge with each rescheduling after a disruption.

3.3. Coupling mechanism

The coupling mechanism is used during training of the approach to adjust the robustness requirement to the centralized planning. The goal is to reduce the robustness requirement as close as possible to the level obtained in 3.1.
The control loop is implemented in a simulation and allows for training in different production systems and different setting (e.g. time frames for schedule regeneration). The following steps are performed iteratively:

1. A centralized robust schedule is generated with a large robustness value (leading to a purposely low performance).
2. The production simulation is started and machine failures are generated randomly according to the real availabilities
3. The performance of decentralized rescheduling with RL is evaluated by post-optimization of each disruption. This way, the makespans can be compared and an optimality gap can be calculated.
4. If for multiple shifts a trend in a decreasing optimality gap is seen, the robustness value for centralized schedule is decreased gradually and steps 1-4 are repeated.

The control loop is in equilibrium if no learning progress in the decentralized rescheduling with RL is possible anymore. In this case, the robustness requirement for centralized scheduling converges towards r*.

4. Preliminary results

4.1. Experimental setting

In the experimental setting, a future body shop is simulated as an MMS. On-time delivery of sub-assemblies of a car body is of greatest importance in that settings. Thus, the JIS criterion should be optimized. However, different availabilities exist, leading to disruptions.

4.2. First results

The RO model was tested thoroughly for different physical buffer sizes. It was shown that for small buffer sizes, no feasible schedules could be found and that there is a tendency of decreased makespan and increased robustness as buffers get larger. The main finding however is that while the makespan-oriented optimization and the RO model generate almost identical makespans, the robustness in the RO model is significantly higher.

4.3. Outlook

Modelling of the decentralized rescheduling with RL and the coupling mechanism are an ongoing implementation effort and currently only exist is simplified versions. In the next step, the decentralized rescheduling with RL will be tested in a simulation with different robustness requirements to the centralized planning and different time frames for rescheduling. This should confirm the utility of a coupling in an experimental setting.

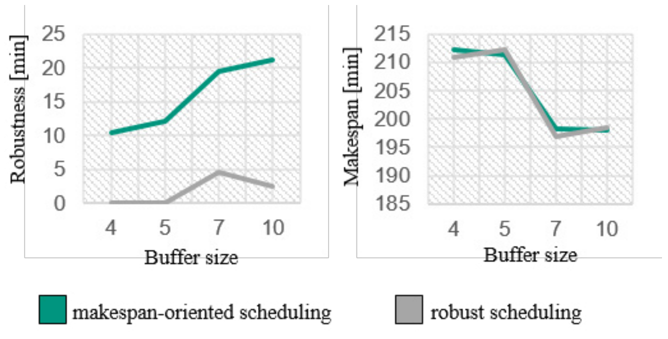

Fig. 3: Robustness and makespan for makespan-oriented and robust scheduling with different buffer sizes.

Acknowledgements

The authors would like to acknowledge the funding support of the German Federal Ministry of Economics and Technology (BMWi) within the joint project “SmartBodySynergy” (01MX15007) and the funding program “ELEKTRO POWER II”.

References

[1] Feldmann K, Slama S. Highly flexible Assembly – Scope and Justification, CIRP Annals - Manufacturing Technology, 50(2), 2001, p. 489–498

[2] Ramsauer C, Rabitsch C. Agile Produktion - Ein Produktionskonzept für gesteigerten Unternehmenserfolg in volatilen Zeiten,Industrial Engineering und Management. Beiträge des Techno-Ökonomie-Forums der TU Austria, Hrsg. H. Biedermann, Springer Gabler, Wiesbaden, 2016, p. 63–81

[3] Schönemann M, Herrmann C, Greschke P, Thiede S. Simulation of matrix-structured manufacturing systems, Journal of Manufacturing Systems, 37, 2015, p. 104–112

[4] Pinedo ML. Scheduling. Theory, Algorithms, and Systems, Springer International Publishing; Imprint: Springer, Cham, 2016

[5] Baker KR, Scudder GD. Sequencing with Earliness and Tardiness Penalties: A Review, Operations Research, 38(1), 1990, p. 22–36

[6] Lödding H. Handbook of manufacturing control. Fundamentals, description, configuration, Springer, Heidelberg, 2013

[7] Wu Z, Weng MX. Multiagent Scheduling Method With Earliness and Tardiness Objectives in Flexible Job Shops, IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 35(2), 2005, p. 293–301

[8] Ouelhadj D, Petrovic S. A survey of dynamic scheduling in manufacturing systems, Journal of Scheduling, 12(4), 2009, p. 417–431

[9] Vieira G, Herrmann, J, Lin E. Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods, Journal of Scheduling 6, 2003, p. 39-62

[10] Stricker N, Lanza G. The Concept of Robustness in Production Systems and its Correlation to Disturbances, Procedia CIRP, 19, 2014, p. 87–92

[11] Scholl A. Robuste Planung und Optimierung. Grundlagen - Konzepte und Methoden - experimentelle Untersuchungen ; Zugl.: Darmstadt, Techn. Univ., Habil.-Schr., 2000, Physica-Verl., Heidelberg

[12] Goren S, Sabuncuoglu I. Robustness and stability measures for scheduling: single-machine environment“, IIE Transactions, 40(1), 2008, p. 66–83

[13] Kouvelis P, Yu G. Robust Discrete Optimization and Its Applications, Springer, Boston, MA, 1997

[14] Manne AS. On the Job-Shop Scheduling Problem, Operations Research, 8(2), 1960, p. 219–223

[15] Jorge Leon V, David Wu S, Storer RH. Robustness measures and robust scheduling for job shops, IIE Transactions, 26(5), 1994, p. 32–43

[16] Xiong, J, Xing LN, Chen YW: Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns, International Journal of Production Economics, 141(1), 2013, p. 112–126

[17] Shen XN, Han Y, Fu JZ: Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems, Soft Computing, 21(21), 2017, p. 6531–6554

[18] Hazır Ö, Haouari M, Erel E. Robust scheduling and robustness measures for the discrete time/cost trade-off problem, European Journal of Operational Research, 207(2), 2010, p. 633–643

[19] Van de Vonder S, Demeulemeester E, Herroelen W, Leus R. The trade-off between stability and makespan in resource-constrained project scheduling, International Journal of Production Research, 44(2), 2006, p. 215–236

[20] Mehta SV. Predictable scheduling of a single machine subject to breakdowns“, International Journal of Computer Integrated Manufacturing, 12(1), 1999, p. 15–38

[21] Sutton RS, Barto A. Reinforcement learning. An introduction, The MIT Press, Cambridge, Massaschusetts, London, 1998

[22] Weiss G. Multiagent systems, MIT Press, Cambridge, Mass., 2013

[23] FIPA: FIPA Contract Net Interaction Protocol Specification, http://www.fipa.org/specs/fipa00029/SC00029H.html, 2002

[24] Busoniu L, Babuska R, Schutter B. A Comprehensive Survey of Multiagent Reinforcement Learning, IEEE Transactions on Systems, Man, and Cybernetics, Part C, 38(2), 2008, p. 156–172.

[25] Gabel T. Learning in Cooperative Multi-Agent Systems. Distributed Reinforcement Learning Algorithms and their Application to Scheduling Problems, Suedwestdeutscher Verlag fuer Hochschulschriften, Saarbrücken, 2010

[26] Gabel T, Riedmiller M. Distributed policy search reinforcement learning for job-shop scheduling tasks“, International Journal of Production Research, 50(1), 2012, p. 41–61