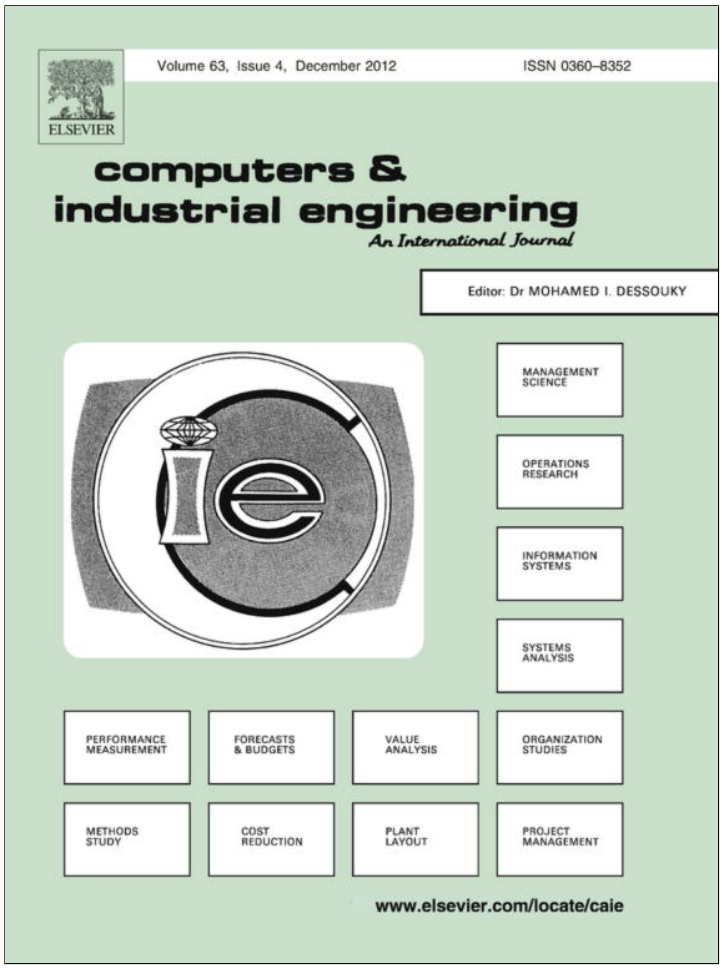

Computers & Industrial Engineering 63 (2012) 901–925

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Agent learning in autonomic manufacturing execution systems for enterprise networking

Milagros Rolón, Ernesto Martínez *

INGAR (CONICET—UTN), Avellaneda 3657, Santa Fe S3002 GJC, Argentina

ARTICLE INFO

Article history:
Received 7 April 2011
Received in revised form 11 June 2012
Accepted 13 June 2012
Available online 20 June 2012

Keywords:
Autonomic systems
Agent-based simulation
Multi-agent learning
Distributed production control
Manufacturing execution systems
Enterprise networking

ABSTRACT

In enterprise networks, companies interact on a temporal basis through *client–server* relationships between order agents (clients) and resource agents (servers) acting as autonomic managers. In this work, the autonomic MES (@MES) proposed by Rolón and Martinez (2012) has been extended to allow selfish behavior and adaptive decision-making in distributed execution control and emergent scheduling. Agent learning in the @MES is addressed by rewarding order agents in order to continuously optimize their processing routes based on cost and reliability of alternative resource agents (servers). Service providers are rewarded so as to learn the quality level corresponding to each task which is used to define the processing time and cost for each client request. Two reinforcement learning algorithms have been implemented to simulate learning curves of client–server relationships in the @MES. Emerging behaviors obtained through generative simulation in a case study show that despite selfish behavior and policy adaptation in order and resource agents, the autonomic MES is able to reject significant disturbances and handle unplanned events successfully.

1. Introduction

Global supply chains, market fragmentation, mass customization and shorter product life cycles have scaled up competition among companies which give rise to the need for introducing cognitive abilities through flexible and easily reconfigurable production systems (ElMaraghy, 2009). To concentrate in their core competences and strengths, networking is the alternative of choice for survival and prosperity of most small and medium enterprises (Westkamper, Huser, & Van Briel, 2000). Moreover, ubiquitous computing and communication technologies readily allow creating virtual integration and temporal associations among enterprises (Ueda, 1992; Van Brussel, Wyns, Valckenaers, & Bongaerts, 1998; Warnecke, 1993; Wiendahl et al., 2007; Xu, Wei, & Fan, 2002). As a result, there is an increasing trend towards inter-firm integration through virtual enterprise networking using *client–server relationships* between order managers (clients) that pose processing requirements for orders, and services providers (servers) that manage resources that are needed to carry out tasks (Canavesio & Martinez, 2007; Vernadata, 2010).

Despite the above mentioned migration towards more integrated and complex enterprise networks operating across multiple companies and regions, still remains the issue of the natural tendency of individual enterprises to act in a selfish way when choosing suppliers or offering services to clients without revealing their decision-making strategies over time. A manufacturing execution system (MES) for enterprise networking must account for this type of strategic interactions when developing distributed production control. Therefore, requirements for distributed MESs in enterprise networking include task execution control performed by autonomous decisional entities that reflect each company's own interest, autonomous local rescheduling triggered by disruptive events, private information protection and correct matches between clients and servers based on capability, cost and reliability issues. In this context, the development of an autonomic system for scheduling and execution control in which multiple entities interact while attempting to achieve their individual goals represents a major challenge from the perspective of designing multi-agent systems.

As guidelines for designing decentralized MESs in a single company, several multi-agent architectures have been proposed. An almost pure distributed architecture for manufacturing control is PROSA (Van Brussel et al., 1998). This architecture was created for holonic manufacturing systems and consists of three types of basic holons (agents) that cover all aspects of heterarchical control and are structured using the object-oriented concepts of aggregation and specialization: order holons, product holons, and resource holons. Although in principle the PROSA definition can be used in multiple-firm environments, the collaborative characteristics of holons make no room for reflecting companieś selfish behavior. Also, the concept of staff holons demands resorting to centralized

* Corresponding author. Tel.: +54 342 4534451; fax: +54 342 4553439.
E-mail address: ecmarti@santafe-conicet.gob.ar (E. Martínez).

0360-8352/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2012.06.004

decision-making elements which are not ideally suited for strategic interactions between enterprises. The holonic MES design proposed by Hadeli, Valckenaers, Kollingbaumb, and Van Brussel (2004), Valckenaers and Van Brussel (2005) and Valckenaers, Van Brussel, Verstraete, Saint Germain, and Hadeli (2007) is based on the PROSA architecture augmented with coordination and control mechanisms. In a holonic MES, the information sharing mechanism is inspired in natural systems based on stigmergy and future state prediction of resource usage conflicts and availability. Unfortunately this design concept is effective for a single company, but not for enterprise networking where each company does not reveal its strategy for action selection to other companies.

Besides PROSA, there have been other heterarchical control architectures for supporting MES design. For example, the ADACOR architecture for production control (Leitão, Colombo, & Restivo, 2005) alternates between transient situations, which are due to the occurrence of disturbances and unplanned events, to stationary states where production control relies mostly on supervisors and coordinator levels. ADACOR is not a purely decentralized structure and thus is not the best option for autonomic execution control based on client–server relationships in enterprise networks. Other academic proposals related to the holonic control concept through purely cooperative interactions are those in Chirn and McFarlane (2000), Blanc, Demongodin, and Castagna (2008) and Covanich and McFarlane (2009), where by design interacting holons are not able to accommodate selfish behavior and strategic interactions.

Designing MESs specifically prepared for enterprise networking is still an open problem, which have been hardly addressed in the existing literature. In Camarinha-Matos (2001) an architectural design for an MES specifically geared towards a virtual enterprise was proposed. However, in this proposal different parts of the manufacturing process are assigned by the coordination system to different companies according to their core skills and roles while direct interactions among enterprises are neither allowed nor encouraged. A framework for production control of virtual enterprises with the holonic paradigm was presented by Huang et al. (2002). It consists of six types of holons and has an upper (global) virtual enterprise echelon and local member enterprise views. In their proposal, a scheduling holon generates resource-task assignments in a centralized way inside each enterprise. In the distributed resource allocation problem for enterprise networking of Anussornnitisarn, Nof, and Etzion (2005), autonomous agents maximize their own decision criteria based on limited information, but they lack automatic rescheduling and execution control capabilities. A holonic manufacturing execution system for enterprise networking along with an industrial case study was presented by Valckenaers, Van Brussel, Saint Germain, and Van Belle (2010). The holonic MES is based on the PROSA architecture, hence it is not suitable for modeling selfish behavior and adaptive policies in the establishment of client–server relationships. Therefore, the development of an autonomic MES design which allows agent-based interactions and selfish behavior for enterprise networking is still a challenging problem to be addressed.

A distributed design for an MES based on truly autonomic units was proposed by Rolón and Martinez (2010) to fill the gap between production planning and shop-floor control. In this regard, it was argued that to make the most of reconfigurable and flexible shop-floors autonomic MESs do not need to rely on a supplied schedule. To this aim, agents using previously defined properties of autonomic computing (IBM Corporation, 2006; Kephart & Chess, 2003) implement a total integration of distributed schedule generation and local execution control. A detailed interaction mechanism of the @MES was modeled and simulated by Rolón & Martinez, 2012. Results obtained for different simulation runs demonstrated that the proposed interaction mechanism is stable and does have a robust disturbance rejection capability. This agent-based design of an autonomic MES readily makes room for introducing selfish behavior and policy adaptation in execution control of multiple servers that concurrently process orders (for clients) and therefore is used here as a basis for the autonomic MES that includes learning capabilities in enterprise networking. In this context, both shop-floor execution control and rescheduling are emergencies from the actions and interactions among selfish agents (Kaihara, Fujii, Toide, Ishibashi, & Nakano, 2010; Wang & Usher, 2004) that learn and adapt continuously their decision-making policies as the behavior of other agents and the environment changes.

Learning has been previously addressed in the context of multiple-firm interactions. Chaharsoogui, Heydari, and Zegorki (2008) proposed a reinforcement learning model for a supply chain based on decisions that consider the entire supply chain. In their work, agents cooperate with each other and their rewards are obtained based on the fulfillment of common goals, not on their individual self interests. In Valluri, North, and Macal (2009) three algorithms for reinforcement learning (Sutton & Barto, 1998) are compared in the context of a distributed supply chain with selfish behavior. However, this comparison does not include algorithms that take into account asynchronous learning in multi-agent supply chains. Selfish behavior was introduced in the proposal of Valluri and Croson (2005) for supplier selection inside a heterogeneous group of sellers. Learning in this case results in a clear-cut separation of sellers capable of producing high-quality goods from those incapable of doing so, but rewards are given only to resource servers and not to the clients.

In this work, an agent-based simulation model of an autonomic MES (@MES) for enterprise networking with learning capabilities is proposed. The interaction mechanism in the @MES (Rolón & Martinez, 2012) has been extended to allow adaptive decision making and selfish agent behavior, typical of integrated manufacturing environments. Thus, dimensions of processing quality, equipment reliability and resources utilization costs have been incorporated into the policy used by order and resource agents. To this aim, learning algorithms for both single-agent (Sutton & Barto, 1998) and multi-agent environments (Leslie & Collins, 2005) are implemented and their performances are compared via generative simulation. Asynchronous learning in multi-agent systems gives rise to emerging behaviors which are difficult to anticipate. Simulation results show that in the proposed autonomic MES, intelligent agents are able to adapt their policies and behave selfishly without compromising the stability and performance of the enterprise network. Thus, introducing learning from interactions in the @MES is the best avenue for matching order and resource agents in virtual markets of clients demanding resource usage for processing tasks and service providers having different capabilities to carry out tasks. Results obtained highlight the importance of agent-based modeling and simulation as a key tool for distributed MESs design and verification.

2. MES for enterprise networks

2.1. Enterprise networking

Some of the strategic objectives that mostly motivate small and medium enterprises to integrate with each other are: maximizing the flexibility and adaptability to changes in its business environment, accessing to new markets, increasing competitiveness due to the possibility of offering a broader product portfolio, allowing access to technology or skills that are not present in a single company, and improving effectiveness and efficiency (Chung, Yam, & Chan, 2004; McClellan, 2003).

Enterprise networks (Canavesio & Martinez, 2007; Vernadata, 2010), extended enterprises (Grossmann, 2009; Kinder, 2003) and virtual enterprises (Valckenaers et al., 2010; Wu & Su, 2005) are different strategies for establishing alliances and temporal integration among enterprises. In this paper, we refer specifically to MESs for enterprise networks, which are groups of small and medium-sized companies that have the characteristics described below (Canavesio & Martinez, 2007; Mallidi, Praskevopoulos, & Paganelli, 1999; Mezgár, Kovács, & Paganelli, 2000):

- Each node is an independent company that associates with peers to achieve common objectives (for example, produce and sell certain product together).
- Each node has equal rights to serve production tasks to fulfill an order.
- Among the nodes, long term relationships are established which give rise to a stable network structure over time.
- Nodes have freedom to decide when to join or leave the network.
- There exists a coordinating node that centralizes information about the actual state of the network nodes in order to know each one skills and availability.

The increasing trend towards inter-firm integration mentioned above emphasizes the need to manage simultaneously multiple shop-floors for order processing in enterprise networks through local execution control and emergent rescheduling. Within enterprise networking, the need to have distributed MESs conformed by autonomous decisional entities is evident due to a mandatory heterarchical organization since there does not exist an enterprise in a network having complete information or overall control over the rest. On the contrary, firms belonging to a network take their decisions based on their own interests and using order- or resource-specific information.

Agents or actors inside each enterprise can have a *client* role, being responsible for a certain order type, or a *server* role, being responsible for task processing or service provision. Therefore, client–server relationships are established among agents as temporal associations between a client that makes a request for processing a task and a server being responsible for meeting on time and with the required quality all committed tasks. Accordingly, proper matches should be formed between a resource (server) with an appropriate quality level and a high reliability rate with a client demanding service or processing. In this context, where multiple companies interact and negotiate to jointly produce certain goods or services, an effective synchronization of activities among the companies is mandatory. Thus, client–server interactions demand from a distributed MES the property of keeping the most appropriate matches regarding quality levels, reliability and prices through local execution control and emergent rescheduling.

2.2. Manufacturing execution systems

An MES is the main software production management tool in-between production planning at the company management level and control/automation systems at the shop-floor level (Blumenthal, 2004; Harjunkoski, Nyström, & Horch, 2009). MESs are complex IT installations which, depending on the form they take, can affect a large number of functional areas in a manufacturing company (Kletti, 2007). Possibilities for using MESs range from quality assurance or personnel management to complex detailed scheduling control systems (Meyer, Fuchs, & Thiel, 2009), but the main role that should be deployed by an MES is production control (Valckenaers & Van Brussel, 2005). If production control is decentralized, decisional activities can be seen as local control activities (Trentesaux, 2009).

MESs are interfaced to local production processes and capture machine data through sensors and actuators that made up automation and control systems such as PLCs (programmable logical controllers), DCSs (distributed control systems), and other intelligent devices. All MESs are also necessarily linked to a production planning system for filling in missing schedule details and coordinating what is being done where and what remains to be done (McClellan, 2000). In this manner, MESs bridge the gap between the high level management information of the enterprise network ERP and online production information from local shop-floors that are managed in their control systems (Bauer & Stoeter, 2010). Typically, an ERP system trickles data for planned orders – including sizes, product mixes, and due dates – down to an MES for schedule generation and execution control. Accordingly, distributed MESs should include shop-floor planning and scheduling tools for transforming planned orders into detailed, executable schedules for verification and tracking purposes of production orders. More importantly, an MES should be able to reschedule in real-time the execution of tasks as soon as unplanned events and disruptions occur (Meyer et al., 2009).

A distributed MES design based on autonomic units with action retries and rollbacks for automatic rescheduling was recently proposed by Rolón and Martinez (2012). The autonomic idea was used in this approach to embrace a total integration of distributed schedule generation and local execution control for agility and responsiveness. To carry out this integration in an autonomic MES the concept of order agents (OAs) and resource agents (RAs) were proposed using previously defined properties of autonomic computing (IBM Corporation, 2006; Kephart & Chess, 2003). Each OA is responsible for completing all orders of a given type as required by its attributes (due date, product mix, required quality level and size) which determines the tasks required for processing the order at the shop-floor. The OA chooses the best route through server selection and follows up execution of a given order while it is being processed. Each RA manages the specific schedule of its resource (e.g., a machine, a production line or local shop-floor) and registers its usage commitments and failure state, if any. Each RA is responsible for the execution of similar tasks for different orders. Selfish behavior and decision policy adaptation will be introduced in the autonomic MES to make it capable for working in an enterprise network environment.

In Fig. 1, information and decisions exchanged between the autonomic MES for enterprise networking with (lower) shop-floor sensors/actuators and with a (upper) production planning system at each server company is shown. In Fig. 1, the company A is the owner of certain resources but do not manage order requirements whereas the company C only has links with customers while receiving production orders. The company B has both the schedule execution (lower) loop that involves an MES and actuators/sensors at the shop-floor level and the other (upper) loop involving both an MES and an ERP system. An OA belonging to the upper loop follows up execution of the planned orders by a given client company's ERP system. RAs belong to the lower loop where the MES is interfaced with the data acquisition and control systems. OAs interact with RAs through direct contact (messages) among concerned agents and indirect interactions through a dynamic Gantt chart which is used as a single blackboard for information sharing about resource utilization and commitments in the enterprise network. The overall schedule is thus an emergence of such interactions and it is perceived in a dynamic Gantt chart.

The autonomic MES has the key role of translating production goals (planned orders) decided at the company B and the company C management levels into a detailed schedule to be executed and controlled within companies A and B based on hardware/software actuators at the shop-floor control level of each company. From bottom up, the state of resources at different shop-floors is

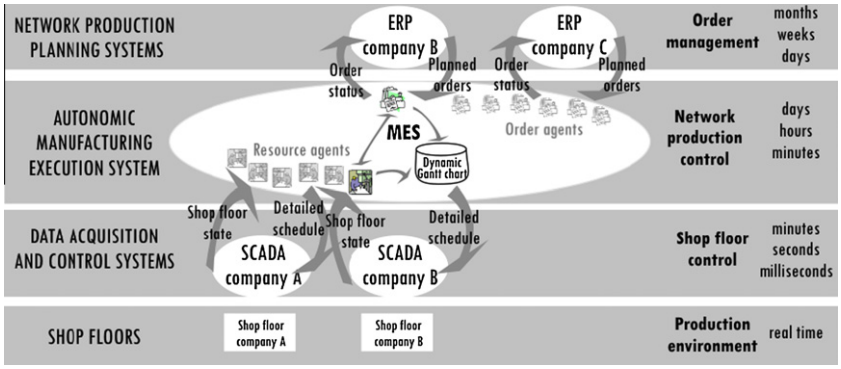

Fig. 1. Autonomic MES for enterprise networking.

abstracted by RAs using data gathered in their shop-floor control systems and fed into the autonomic MES for execution control and rescheduling. State of orders being processed are reported to companies B and C management levels through the concerned OAs which may affect the release of future orders and may change policies for establishing client–server relationships.

The interface between an MES and upper/lower management levels in Fig. 1 must be based on a proper design using increasingly abstract sensors and actuators to address the important differences in information scope and detail used. Data acquisition and control systems, at different shop-floors, generate highly disaggregated information with a very limited scope (e.g., a specific resource), whereas the network production planning system requires abstract information such as the status of an order being processed in different companies. Besides, both OAs and RAs should be designed with learning and reasoning capabilities to make the @MES sufficiently reconfigurable and cognitive to consolidate stable relationships among OAs and RAs and to successfully face highly dynamic and uncertain environments.

3. Agent learning

3.1. Learning in multi-agent systems

Most of the work in modern artificial intelligence is about learning from reinforcements or evaluative feedback received by an individual agent (Sutton & Barto, 1998), where the design focuses on an agent or controller that learns to behave successfully in an uncertain environment. A multi-agent viewpoint highlights that the overall system as a whole is made up of situated autonomic entities (Mannor & Shamma, 2007). When we consider learning in multi-agent systems, the environment includes the existence of other agents. Therefore, the problem is not only that other agent's learning will affect the environment perceived by a given agent, but also to what extent those environmental changes will depend partly on the actions taken by this agent (Shohan & Leyton-Brown, 2009). Consequently, individual and collective learning will be impacted also by each individual agent. This asynchronous learning implies that continuous update of agent policies gives rise to a complex adaptive system, where simple learning rules coupled with goal-oriented interactions typically lead to unpredictable emergent properties.

Because of complex adaptive behavior, the problem of designing distributed control systems is notoriously difficult (Mannor & Shamma, 2007), mainly due to the presence of self-interested agents who want to maximize her/his own profit by all means (Vlassis, 2007). In this context, the problem of designing interactions that are collectively robust and individually rational for each individual agent is the subject of mechanism design. The interaction mechanism of the @MES proposed by Rolón and Martinez (2012) was proven to be resilient and stable despite the total autonomy given to OAs and RAs when negotiating resource usage without resorting to *a priori* defined schedule. Nevertheless, their interacting agents were so far not capable of learning from evaluative feedback of decisions taken. The motivation behind introducing learning to the agents of the @MES model is that adaptive decision policies would increase MES realism and the robustness of an agent behavior under a variety of shop-floor conditions. Specifically for an OA, it is intended that through learning it would find the feasible processing route with the best price-reliability relation. To meet this objective the OAs would select those RAs that offer the most convenient price avoiding being served by RAs which are not reliable enough according to their own experience regarding delivery times. For the RAs, learning would make them able to accumulate more revenues in the long run by selectively processing only certain tasks. To this aim, the RAs should learn to adjust their processing quality to serve as many profitable tasks as possible. Also RAs must learn choosing a proper pricing strategy for services bearing in mind maximizing resource utilization.

Objectives defined above for introducing learning in OAs and RAs that interact through the @MES mechanism could be addressed in any or all of three following ways: as individual learning in which agents learn from their own experience; as evolutionary learning in which the population of agents learns because some agents "die" and are replaced by better agents, leading to improvements in the population average; and as social learning in which some agents imitate or are taught by other agents, leading to the sharing of experience gathered individually but distributed over the whole population (Gilbert, 2008). In the proposed @MES, learning is achieved mainly from evaluative feedback of each OA while traveling along a production route and by each RA while selecting quality levels and making service commitments for different tasks (individual learning). Additionally, OAs may also learn while sharing information about rewards gathered in alternative processing routes (social learning). Reinforcement learning (Sutton & Barto, 1998) is the approach used here for introducing learning to the @MES model. This computational approach for introducing learning in the @MES will focus on the problem faced by OAs and RAs that must learn through trial-and-error interactions. Action selection in turn affects the shop-floor state of enterprises acting as server providers which gives rise to a continuous update of decision-making strategies used by all OAs and RAs.

3.2. Reinforcement learning

When reinforcement learning is introduced in a multi-agent system, the environment is set up so that it provides rewards to all agents (positive or negative reinforcement signal), which are

directly linked to their roles and specific goals. The objective of a reinforcement learning agent is to maximize the total reward or return it receives over time. While seeking to maximize the return for a sequence of actions, each agent must find the optimal policy, that is, the optimal way of behaving in order to accumulate rewards despite other agents change their policies and environmental changes occur (Sutton & Barto, 1998).

The most important algorithm in the reinforcement learning field is *Q-learning* (Watkins, 1989). A simplified setting to understand Q-learning is called evaluative feedback (Sutton & Barto, 1998, chap. 2). In this simplified version of Q-learning, the issue is to choose among alternative actions as in multi-armed bandits. Each action a has a value $Q(a)$, that is, an expected or mean reward when action a is chosen. If the Q-value of each feasible action were known, then it would be trivial for the agents to behave optimally: they would always select the action with the highest Q-value. Nevertheless, the Q-values for alternative actions are not known in advance, but should be estimated by trying them and resorting to a learning rule which accounts for evaluative feedback. If value estimates for the action Q-values are kept, then there is at least one action a^* whose estimated value is the greatest. This is called the *greedy action*.

If an agent always selects the greedy action, it is said that the agent is exploiting his current knowledge of the Q-values for available actions. If instead the agent selects any of the non-greedy actions, it is said that it is exploring, which enables the agent improving its estimation of the Q-values for actions that seem inferior to the apparently optimal one (Sutton & Barto, 1998). Exploitation is the right thing to do in order to maximize the short-term expected reward, but exploration may produce a greater return in the long run. Therefore, exploration and exploitation should be balanced and there are many methods to do so. *Softmax* is one of those methods and has been widely used in different learning systems to handle the exploration/exploitation balance. It is based on the Boltzmann distribution:

$$\beta(Q_t)(a) = \frac{e^{Q(a)/\tau}}{\sum_{b\in A} e^{Q(b)/\tau}} \tag{1}$$

where τ is a positive parameter called the *temperature* that gives rise to differences in action selection probabilities $\beta(Q_t)$ depending on its value. When the temperature is "high," all actions will be (almost) equally probable and Q-values are almost irrelevant for defining action selection probabilities. As the temperature decreases, an action having a high Q-value will be selected with higher probability, which highlights exploitation rather than exploration (Sutton & Barto, 1998).

At each time step t, the agent takes the action a, gets the subsequent reward r_t, and the estimated Q-value for that action $Q(a)$ is then updated using an incremental learning rule as follows:

$$Q_{t+1}(a) = Q_t(a) + \alpha_{t+1}[r_t(a) - Q_t(a)] \quad \text{for each } a \in A \tag{2}$$

where A stands for the set of available actions and α is the learning rate. This single-agent learning rule is referred here as *Single-agent Q-learning*, where each agent estimates the value of taking a given action from the set A at each time step while using a policy that is a simple function of the Q-values estimates without taking into account the presence of other agents in its environment—the only information used is the reward received for each action taken.

To account for an agent environment populated by other agents an alternative is resorting to a Q-value update rule such as the *Individual Q-learning* proposed by Leslie and Collins (2005). These authors tested their algorithm in two strategic games for which most learning algorithms fails to converge. In the Individual Q-learning rule, each agent selects an action a, receives a reward r_t, and then updates the corresponding Q_{t+1} according to Eq. (3):

$$Q_{t+1}(a) = Q_t(a) + \alpha_{t+1}\frac{r_t - Q_t(a_t)}{\beta(Q_t)(a_t)} \quad \text{for each } a \in A \tag{3}$$

In Eq. (3) above, the reward prediction error $(r_t - Q_t(a))$ (Sutton & Barto, 1998) is divided by the probability for a given action a to be selected according to the *Softmax* criterion in Eq. (1) $(\beta(Q_t)(a))$. This can be viewed as compensating for the fact that actions chosen with low probability do not receive frequent updates of their Q-values. So, when such actions are selected any reward prediction error must have greater influence on the Q-value updates compared to updates for actions that are more frequently selected. Division by $\beta(Q_t)(a)$ results in a system that is closely related to the well-studied smooth *best response dynamics* (Hofbauer & Hopkins, 2005). In Eq. (3), $\{\alpha_t\}_{t\geqslant 1}$ is a deterministic sequence of learning rates that must satisfy:

$$\sum_{t\geqslant 1}\alpha_t = \infty, \sum_{t\geqslant 1}(\alpha_t)^2 < \infty \tag{4}$$

for Q-values to converge under stationary conditions. These restriction for bounded sequences in the learning rate values mean that standard theorems of stochastic approximation can be used (Benaïm & Hirsch, 1999) and must involve some strategy for decreasing the learning rate α as more interactions have been experienced by an agent.

3.3. Learning client–server relationships

Adapted from the Contract Net protocol (Smith, 1980), the basic mechanism for establishing client–server relationships in the enterprise network is shown in Fig. 2. In this section, the introduction of reinforcement learning in order to guarantee optimal matching between order and resources agents is initially presented using an idealized setting where each order type corresponds to only one task with given quality requirements. Firstly, the OA associated to a given task type makes public the need for a service provision. As a response, each RA make a bid based on selecting the

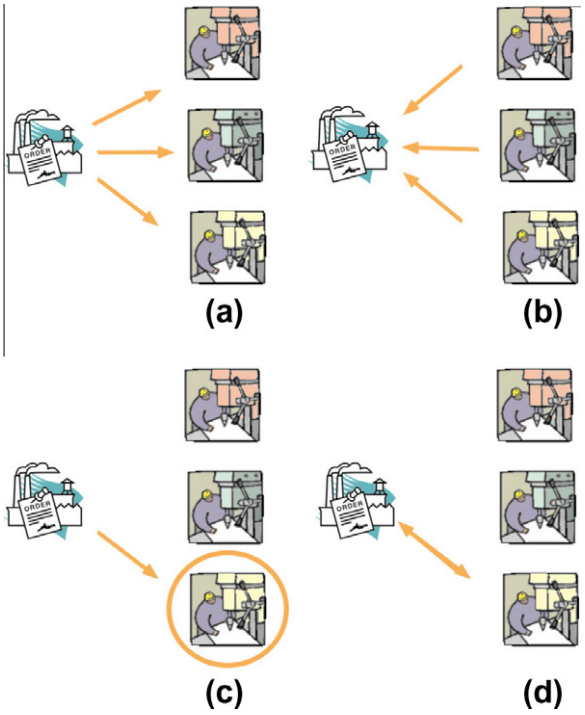

Fig. 2. Steps for establishing client–server relationships. (a) Announcement. (b) Bidding. (c) Awarding. (d) Establishment.

quality level for the task, which determines the price and processing time in the offer. Later on, in the awarding stage, the OA evaluates the received offers and selects a RA. This decision is based on both the offered price and the reliability perception in the fulfillment of the promised time by each RA. Finally, in the establishment stage, the contract is closed, the client–server relationship is established and the chosen RA begins to execute the task. On this basis the OA receives its reinforcement signal (benefit associated with the contract price). The OA updates the reliability index for the selected resource and also receives its reward.

Let us assume an OA_j has a service request (task) with a quality level which is not exactly known *a priori* by any of the servers (RAs). Task requirements can only be inferred through ongoing interactions with the OA_j. In this regard, although the execution quality of a task is a multifaceted concept, it is represented here by a real value between 0 and 1 (considering that processing quality is computed as the weighted sum of quality factors). It is assumed that any RA_i can process the task by choosing a quality level U_{R_i} from its available alternatives. The chosen quality by RA_i determines the offered time t_{a_i} and the price P_i quoted in the bid. The mismatch, if any, between the required quality level and the one provided can only be known by the OA_j when the task has been finished.

For the sake of simplicity, let us assume in remaining part of this section that the intrinsic processing time t_s is the same for all tasks in all resources; the offered processing time by RAs in Fig. 2b is given by $t_{a_i} = (1 + U_{R_i}) \times t_s$ and is related to the price P_i, asked to the OA_j for processing a type j task by:

$$P_i^j = k_r \times t_{a_i} \tag{5}$$

where k_r is the price per time unit of resource utilization. The reliability of the RA_i is perceived by the OA_j as determined by the reliability index R_{ij} which is updated once each client–server interaction between OA_j and RA_i had ended (as it has been pointed out by Tran (2010), agents do not trust reliabilities perceptions provided by other agents). Reliability rates are initialized to 1 (it is assumed that new RAs are totally reliable) and changes by ±0.1 depending on if the offered finalization time for the task has been achieved or not. OAs learn to choose the subset of RAs that offer the best price/reliability ratio. Thus, Q-values are based on rewards for each OA_j following an interaction with a RA_i and describe the corresponding price and reliability index for the chosen resource.

Whenever the required quality level U_{o_j} for the task is higher than the one chosen by RA_i for processing the task, U_{R_i}, a rework operation must be carried out in the same equipment item. The amount of rework is modeled here by $10 \times (U_{o_j} - U_{R_i})$ if $U_{o_j} \geqslant U_{R_i}$. Reprocessing time is also estimated as 25% of the original processing time which defines the actual processing time as follows:

$$t_{a_i}^{real} = \begin{cases} \left\lfloor 1 + 2.5(U_{0_j} - U_{R_i}) \right\rfloor \times t_a & \text{if } U_{o_j} \geqslant U_{R_i} \\ t_{a_i} & \text{if } U_{o_j} < U_{R_i} \end{cases} \tag{6}$$

The actual reward the resource agent RA_i gets is associated with the mismatch between the actual finalization time and the offered time. This reinforcement signal for the resource agent decreases in case of the offered finalization time is exceeded, namely there exists a $delay_i = t_{a_i}^{real} - t_{a_i}$ if $t_{a_i}^{real} \geqslant t_{a_i}$. Also, for each time step t where the bid made by a RA_i is not accepted, the resource agent receives a zero reward

$$r_t^i = \begin{cases} \frac{P_i}{1+delay_i} & \text{if } RA_i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Considering resource reliability as an index of proper operation for a specified time (Grant & Leavenworth, 1996), reliability of a processing route in an enterprise network will be considered as the reliability of the "weakest link" for such route. Therefore, OAs should learn to choose those processing routes with the best price/reliability relation, and rewards obtained for each OA_j will be associated with the overall price paid for processing a given order considering the n tasks in the recipe and the reliability of the least reliable resource in the chosen processing route as follows:

$$r_t^j - \frac{\sum_{i=1}^{n} P_i^j}{\min(R_{ij})} \tag{8}$$

3.4. Simulating learning curves

To test how client–server relationships can be learned using *Single-agent Q-learning* (i.e., plain Q-learning) and *Individual Q-learning*, before introducing them into the @MES, isolated simulations were performed with four OAs and four RAs. Table 1 contains the quality levels (actions) that can be chosen by the RAs while processing the first and the second task of each product (order) recipe, and also the required quality levels for each of the four orders, one for each OA.

In this experiment a standard processing time $t_s = 10$ and a unit processing price of $k_r = 1$ were considered for all tasks in all servers. At each time step t, a chosen order type is selected randomly. Q-values were initialized to zero for all agents. For both algorithms the learning rate α_τ was updated by Eq. (9), where t represents the number of time steps, $\alpha_0 = 0.3$ and $m = 12$ for RA_1 and OA_1, $m = 14$ for RA_2 and OA_2, $m = 16$ for RA_3 and OA_3 and $m = 18$ for RA_4 and OA_4.

$$\alpha_t = \frac{\alpha_0}{\left(1 + \frac{t}{m}\right)} \tag{9}$$

For both learning algorithms, a *Softmax* policy was used to balance exploration and exploitation. In the Individual Q-learning the temperature parameter τ is set to a value equal to 1. In the Single-agent Q-learning a temperature value that gradually decreases over time was used for reducing exploration and exploiting experience as time passes, according to Eq. (10), where parameters are set to: $\tau_0 = 2$ and $c = 50$.

$$\tau = \frac{\tau_0}{\left(1 + \frac{t}{c}\right)} \tag{10}$$

Optimal actions for the RAs are given in Table 2. This information is used as a reference for performance comparison between both learning algorithms. When there is only one resource capable of producing the required quality level, the optimal action is to

Table 1
Quality levels available to RAs and required product quality levels.

RA executing the first task	RA executing the second task	Required quality level
$U_{r_1T_1} = [0.0\ 0.1\ 0.2]$	$U_{r_1T_2} = [0.0\ 0.1\ 0.2]$	$U_{o1} = [0.1]$
$U_{r_2T_1} = [0.3\ 0.4\ 0.5]$	$U_{r_2T_2} = [0.3\ 0.4\ 0.5]$	$U_{o2} = [0.4]$
$U_{r_3T_1} = [0.6\ 0.7\ 0.8]$	$U_{r_3T_2} = [0.6\ 0.7\ 0.8]$	$U_{o3} = [1.0]$
$U_{r_4T_1} = [0.8\ 0.9\ 1.0]$	$U_{r_4T_2} = [0.8\ 0.9\ 1.0]$	$U_{o4} = [0.7]$

Table 2
Optimal actions for the RAs regarding quality levels.

Optimal action	RA involved	Product involved	Optimal quality level
$U_{r_1T_{1/2}}(3)$	RA_1	Product 1	0.2
$U_{r_2T_{1/2}}(3)$	RA_2	Product 2	0.5
$U_{r_3T_{1/2}}(2)$	RA_3	Product 4	0.7
$U_{r_4T_{1/2}}(3)$	RA_4	Product 3	1.0

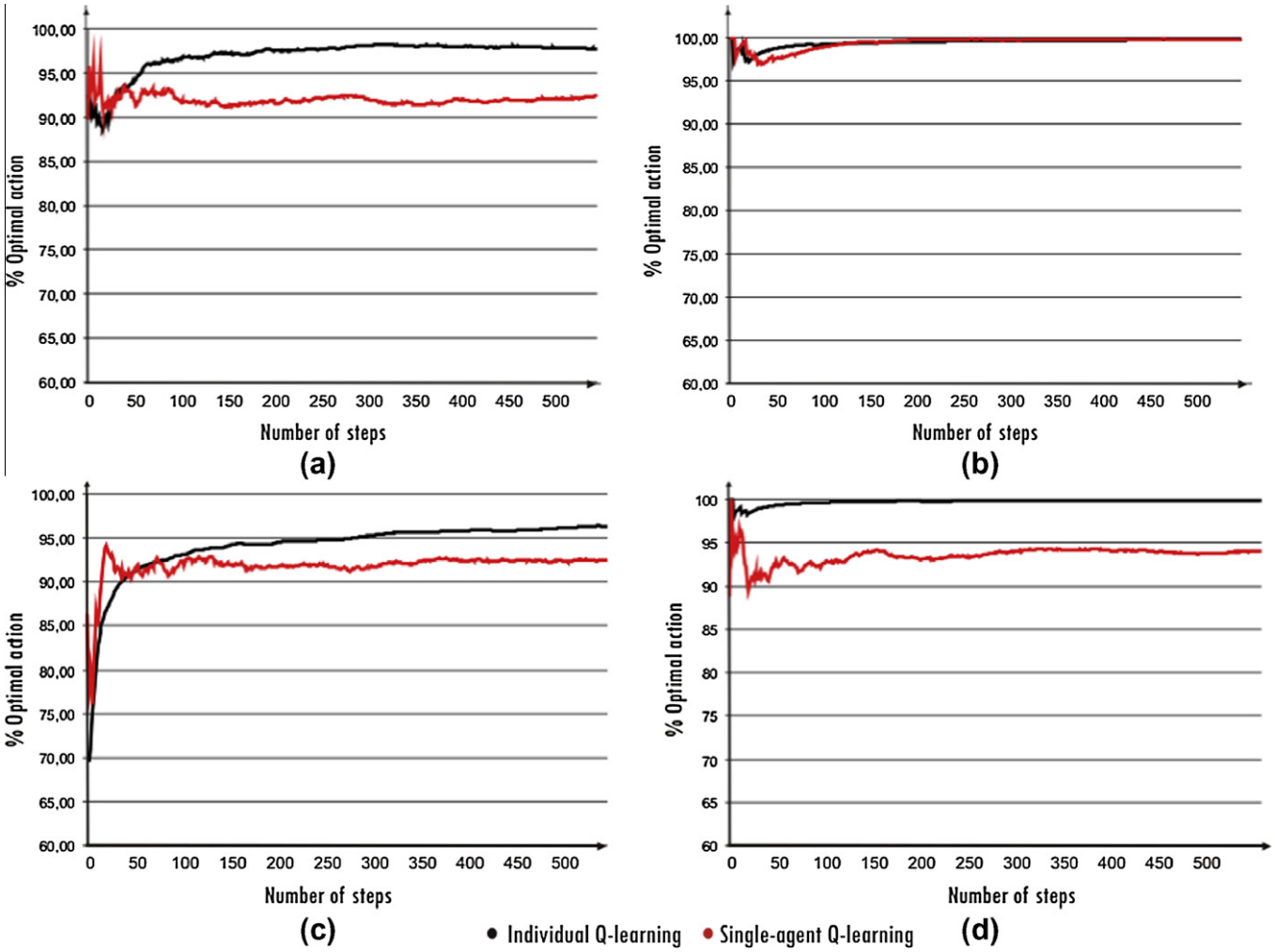

Fig. 3. Comparison of the optimal actions executed by RAs for task 1. (a) Actions executed by RA 1. (b) Actions executed by RA 2. (c) Actions executed by RA 3. (d) Actions executed by RA 4.

process the task at the maximum level of quality. However, if there exist competition among resource agents that can produce the task at the required quality level (e.g., both RA_3 and RA_4 for tasks of product type 4), it is more suitable for RA_3 to process tasks at the exactly required quality level rather than at the maximum one in order to guarantee that the client–server relationship with OA_4 is actually established.

In Fig. 3, a performance comparison for Individual *Q*-learning and Single-agent *Q*-learning is made regarding (optimal) action selection for task 1. Simulation results for task 2 reveal similar performance. To obtain learning curves, 500 steps were simulated in each learning episode and the results obtained in 10 episodes were averaged. Results obtained for task 1 and task 2 highlight that optimal matching is obtained when the Individual *Q*-learning rule is used.

Optimal action selection for an OA_j once all offers from RAs have been received is based on the highest *Q*-value for both first and second tasks, respectively. In Fig. 4, the comparison of Individual *Q*-learning and Single-agent *Q*-learning for the OAs is shown for the simulation study. Learning curves are based on simulating 500 decision steps in 10 independently generated episodes and averaging results obtained. Results for all OAs exhibit that the performance of the Individual *Q*-learning is higher than the Single-agent *Q*-learning performance for OA_3 and OA_4, and are similar for both algorithms when considering OA_1 and OA_2. Note that the rate of convergence towards the optimal policy is higher for the Individual *Q*-learning for all OAs. Also, it worth noting that after 20 time steps, actions selected using the Individual *Q*-learning correspond to the optimal action, which suggest that this learning rule is better in exploiting knowledge generated through evaluative feedback.

4. Interaction mechanism

The interaction mechanism in the @MES for enterprise networking has been designed using the Prometheus methodology integrated with Hermes (Cheong & Winikoff, 2006), which provides enough room for detailed specification of goal-oriented interactions in multi-agent systems, including descriptions for their roles and data needed in role deployment. An interaction-goal hierarchy, action maps, action sequence and action message diagrams were previously specified for the interaction mechanism of the @MES proposed by Rolón and Martinez (2012).

Regarding data stores, there are five databases in the @MES that make shop-floor execution control and emergent rescheduling feasible:

(1) The *process and services knowledge database,* containing information about different resources in the enterprise network, reconfigurable capabilities, operations and services

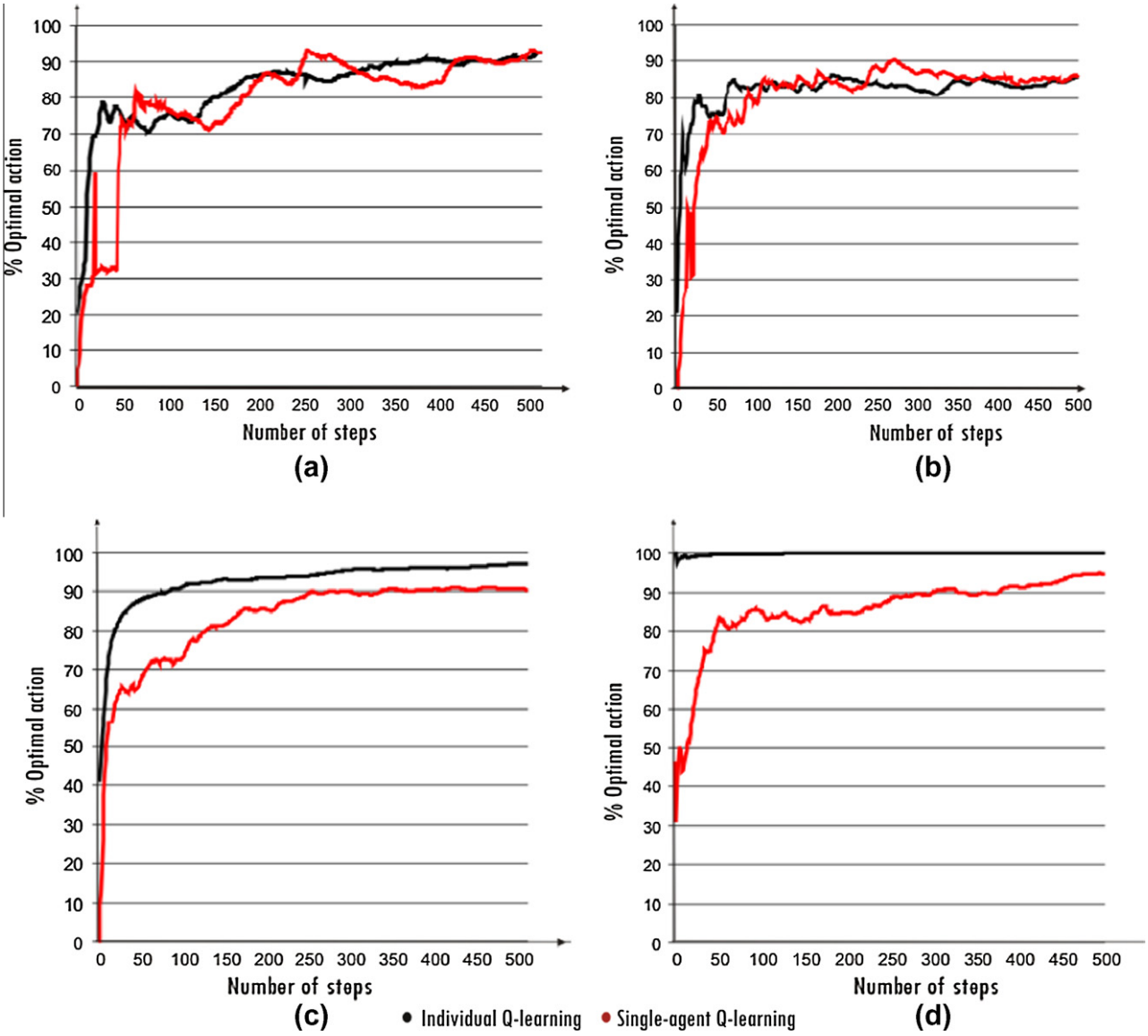

Fig. 4. Comparison of the optimal actions executed by OAs. (a) Actions executed by OA 1. (b) Actions executed by OA 2. (c) Actions executed by OA 3. (d) Actions executed by OA 4.

per resource along with required inputs and the work-plan per product type which determines the different production formulas or recipes.

(2) The *Gantt chart database,* holding the current schedule for each resource committed to a group of tasks for a certain time window. It contains also the 'broken' or 'in maintenance' indicators for certain resources according to their actual condition.

(3) The *executed orders database,* that has records related to orders that have been recently executed and have already left the network.

(4) The *pending orders database,* containing records associated with outstanding orders within the @MES.

(5) The *non feasible orders database,* holding records of rejected orders.

The generalized action maps for the interaction-goal hierarchy are shown in Figs. 5–8 for the following interaction-goals: Order Acceptance, Resource Commitment, Monitoring (orders), Monitoring (resources) and Rescheduling. The learning approach discussed in Section 3 involving the dimensions of processing quality, reliability and equipment costs is incorporated into the interaction mechanism so that optimal client–server relationships are established in the @MES based on *Q-values* for available actions to each agent type.

The developed action maps incorporate scenario variations using failure handling procedures in Hermes to address unplanned events. Termination and action retries for action failures, as well as rollbacks when faced with an interaction goal failure were implemented. For example, in Fig 7, an action retry is used when the concerned RA updates the Gantt chart to inform that the execution time of a task was actually larger than scheduled. If the change does not disturb other committed tasks, then order processing will continue according to the existing schedule. An example of a rollback for interaction-goal failures is shown Fig. 6, where if the OA cannot generate a solution, the Resource Commitment interaction-goal is rolled back to the Order Acceptance interaction-goal that will be achieved eventually in a different manner, and consequently the interaction logic leads to a different result. Figs. 6 and 7 show the interactions between an OA and several RAs, whereas

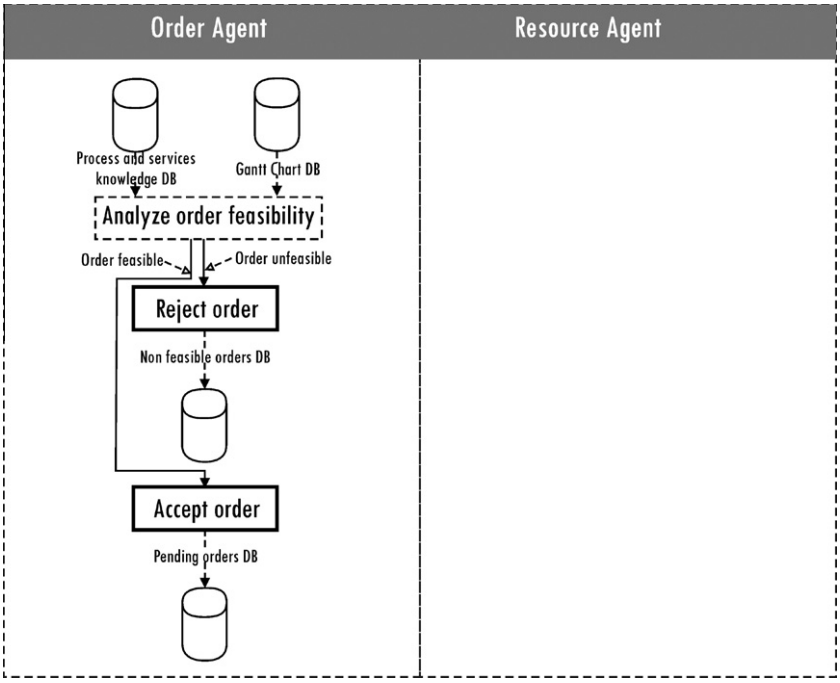

Fig. 5. Action Map for Order Acceptance Interaction-Goal.

Fig. 8 depicts strategic interactions among several RAs and several OAs. In these three action maps, the parallel split and merge symbols highlight multi-agent system parallelism where multiple instances of OAs or RAs are interacting asynchronously. This parallelism in casualty links are denoted by a "crow's feet" symbol and the multiplicity is shown beneath it. For example, in Fig. 8, when a failure of the resource item is detected or maintenance operations are planned, the concerned RA informs the fact through Gantt chart registration, and all OAs involved have to remove resource earmarking and roll back to the resource commitment interaction-goal to reschedule the affected tasks.

The action map of Fig. 5 describes all actions involved in the Order Acceptance interaction-goal. Particularly when a new order enters the enterprise network, the concerned OA which is managing the order execution analyzes its feasibility through Gantt chart exploration. If all tasks associated with the order can be virtually inserted into the existing Gantt chart, the order is accepted and continues to the next interaction goal. On the other hand, if the order is infeasible it will be rejected and returned to the production planning system (ERP) of the respective company. If for example, the due date of the rejected order is modified, then the order is re-submitted to the network production control level. However, it will enter in the @MES as a new order.

Following order acceptance, a concerned OA asks candidate RAs of each task in the production formula (see Fig. 6) a probable finalization time, price and a due date for complete task servicing. RAs decide task allocation to each resource usage plan according to local dispatching rules, which are quite insensitive to unexpected events and yield very robust behavior (Cicirello & Smith, 2004). In this regard, RAs follow different criteria to define resource scheduling such as SPT (shortest processing time), EDD (earliest due date) and FIFO (first in first out), which result in different design parameters for the @MES. Selected RAs provide different options for probable finalization times and processing prices of their resources according to their quality level selection based on their *Q*-value estimations (see Section 3). These options given by selected RAs allow the concerned OA to make a list of candidate solutions, of which the OA chooses the best one according to its *Q*-values for action selection. Therefore both agent types decide on their choices (task quality level for the RAs and processing route for the OAs) through a reinforcement learning algorithm (see Section 3). After selecting the solution for processing the order, the OA in charge of the order answers all offers received in due course so as to commit the chosen service providers. As a response to the demand for booking of the OA, selected RAs register the assigned time slot into their resource usage plan. After resource booking is finished, the OA secures all the inputs needed by the production formula for the product in the order.

As can be seen in Fig. 6, the option compatibility (no operation is scheduled to begin before the previous one has finished) and completeness (all the operations of the process route are scheduled) are checked through different activities of the action map before generating a list of solutions. If these conditions are not met, selected RAs are asked to get alternative task insertion options into their resource plans or to remove their resource earmarking.

Order monitoring, and the corresponding rescheduling activity due to disruptive events (Fig. 8), as well as resource monitoring and rescheduling (Fig. 9) are both described in only one activity map because they are tightly linked through action retries. Regarding order execution (Fig. 8), an OA requests moving raw materials and semi-finished products to the corresponding resource queue in order to be processed. Whenever the time to begin executing a task is reached, and the materials or semi-finished products have arrived to the equipment area, the concerned RA executes the task and eventually the OA evaluates the task outcome comparing the quality requirements for the task with the actual quality obtained. If quality requirements are satisfied, the next operation is executed until there are no pending operations in the product recipe. When order processing has finished, each RA receives its reward, associated with the price paid by the OA if the resource was selected, decreased in case a processing delay occurs (see Eq. (7)). Unselected RAs receive zero reward, as it is shown in Eq. (7). The concerned OA also receives its reinforcement signal, related to the price paid for processing all the tasks in the processing route and the reliability of the least reliable resource, as it is shown in Eq. (8). After all involved agents in processing the order receive their corresponding

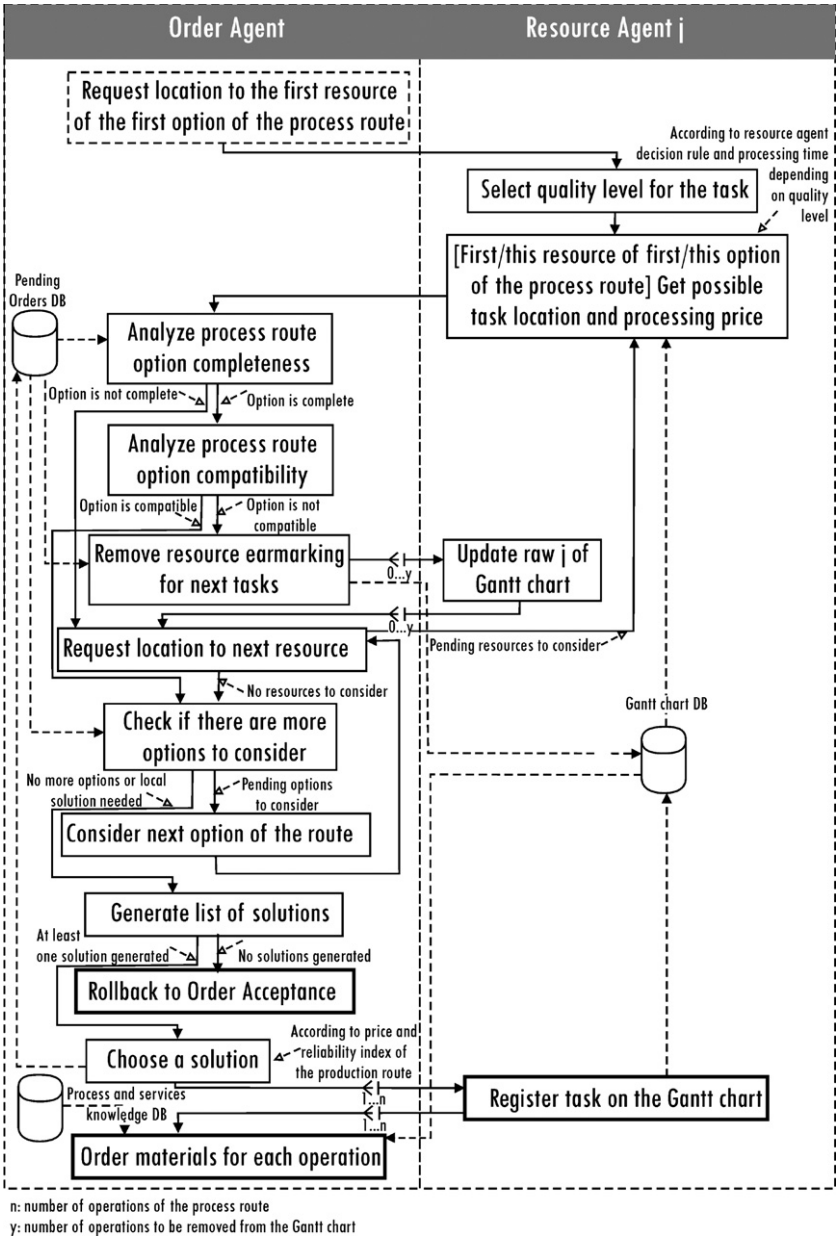

Fig. 6. Action Map for Resource Commitment Interaction-Goal.

rewards, the concerned OAs stores data about the completed order in the *executed orders* data base.

RAs in a given enterprise check their resource states based on the information received from their data acquisition and control systems in their corresponding enterprise shop-floors. The corresponding action map for monitoring resources and rescheduling interaction goals are shown in Fig. 8. If a RA receives a failure condition for its resource from the shop-floor/automation level, it diagnoses the failure type and probable duration of the unavailability. Should there exist planned maintenance operations, the concerned RAs have to inform through a Gantt chart registration event the time period for which the resource is not available so that all OAs can be aware of such circumstance when looking up for processing feasibility of future orders. In case there are tasks already scheduled over the duration of resource unavailability period, affected OAs will have to remove resource earmarking and roll back to the resource commitment interaction-goal to reschedule outstanding tasks.

5. Generative simulation

5.1. Generative modeling and simulation

When adopting a multi-agent view of MESs, it soon becomes apparent that decentralizing production control revolves around the design of an artificial society of agents where goal-oriented interactions are the leverage points for emergent behaviors (Azadegan & Dooley, 2010; Ferber, 1999). Moreover, if there exists simultaneous learning in the policies used by all actors involved in this multi-agent environment, emergent behaviors that are difficult to predict arise. In this context, generative modeling refers to the application of computational models to understand and influence emergence properties of complex social systems (Epstein & Axtell, 1996; Epstein, 2006). More specifically, generative modeling is able to discover in advance emergent connections between @MES components. Agent-based simulation of detailed micro-processes gives rise to system-level knowledge and archetype patterns

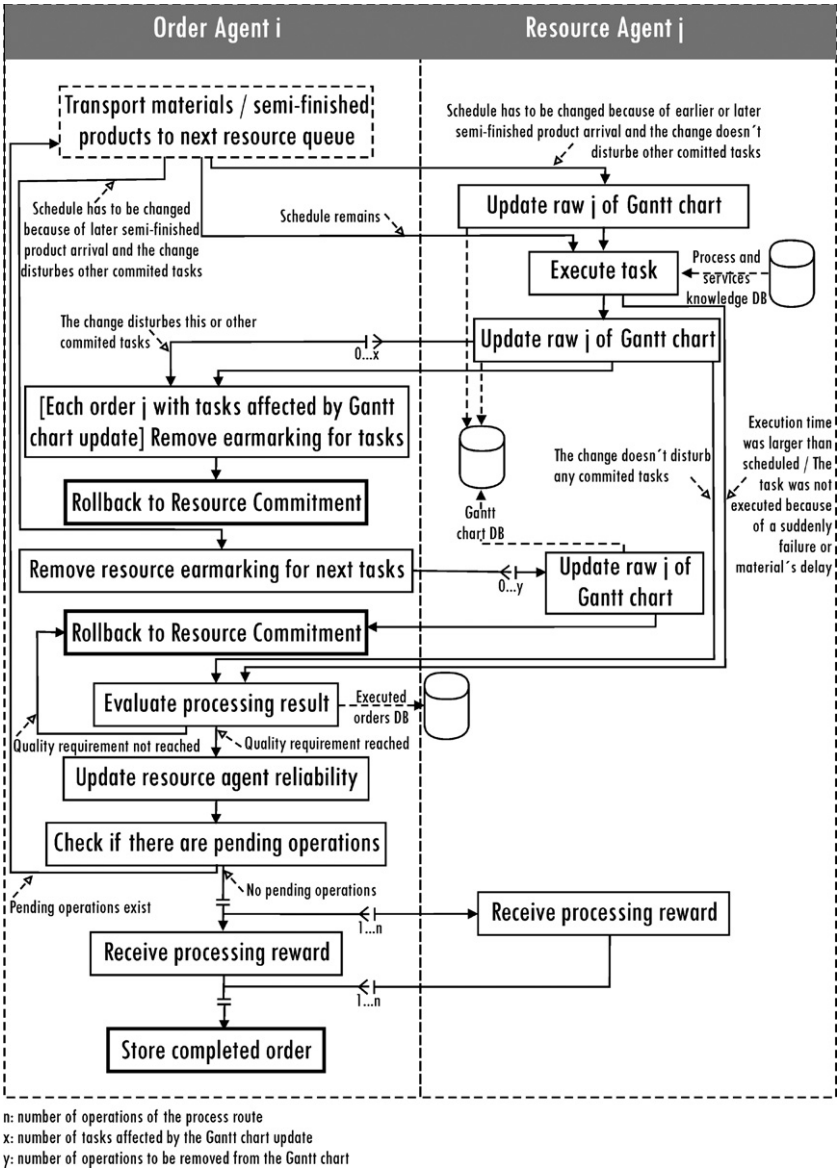

Fig. 7. Action Map for Monitoring (orders) and Rescheduling Interaction-Goals.

causing emergent behavior, and in this way it may identify possible emergences that are outside the range of analytical thinking (Epstein & Axtell, 1996; North & Macal, 2007). To this aim, generative modeling and simulation must follow an iterative model construction process.

The above process starts with an initial description of individual agent behaviors as well as interaction mechanisms along with supporting data and model parameters. This conceptual model is then converted to a generative model that can be used to test hypothesis (Schelling, 1978). The resulting agent-based model is then simulated and the initial results are examined. In such a way a controlled micro-world simulation is used to discover macro-level behaviors resulting from on-going interactions at the micro-scale. The agent internal decision-making policy and the interaction mechanism in the model are then updated based on goals that different agents must achieve, and the model is simulated again. Thus the model serves as a tool to validate the hypothesis (simulated experimentation) and as a guideline to improve the logic and detailed specifications of interaction mechanisms in multi-agent systems.

Generative modeling requires a focus on simple entities and their interactions. Models create solid and stable building blocks that encapsulate key parts of the real system's behavior. Their implications tend to be robust to significant changes in the underlying structure, they tend to produce "surprising" results that motivate new predictions, they can be easily communicated to others and they are fertile grounds for new applications and contexts (Epstein & Axtell, 1996; Miller & Page, 2007). As the structure of a model becomes more complicated, many of these desirable features are lost, and we move away from modeling towards simulation. In Fig. 9, the real world along with modeling and simulation are represented for the @MES. There, it is shown that within the @MES model, there is no intention to model any particular system, while when we move to the simulation field, a specific case study for the @MES must be defined.

In contrast to the traditional top-down approach, a bottom-up emergence in an artificial society of agents defines patterns that result from the chosen design for the interaction mechanism. Although decision-making policies used by all agents might be quite simple, the result of ongoing interactions among them can be very complex and difficult to predict, mainly when agent internals are heterogeneous and each individual agent learns over time based on its unique experience and evaluative feedback. In these

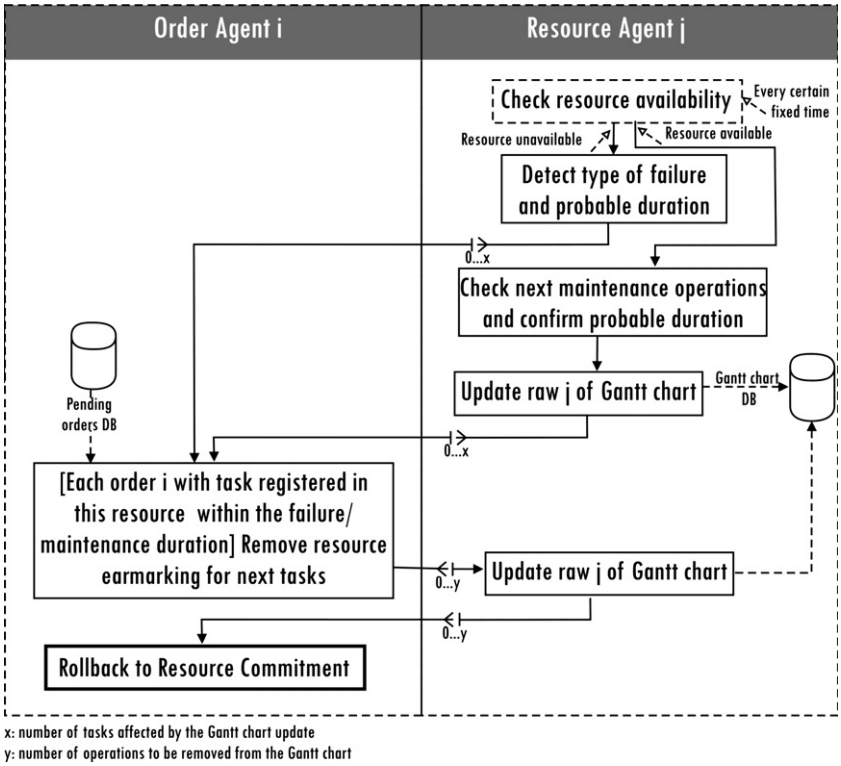

Fig. 8. Action Map for Monitoring (resources) and Rescheduling Interaction-Goals.

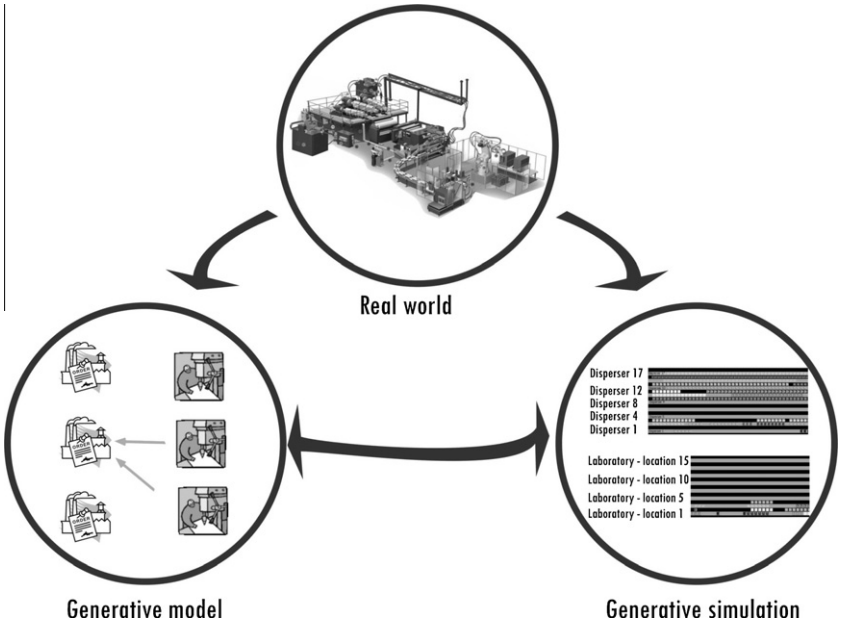

Fig. 9. Generative Modeling and Simulation.

multi-agent environments, actions and interactions of local controllers that are learning and adapting their policies as the behavior of other agents and the environment change lead to a complex global behavior.

The applicability of generative modeling is especially suited for industrial environments, since interacting parts are naturally distributed in both time and space, their properties change over time, there are objectives and constraints in conflict with each other and emergent phenomena can be exhibited at different levels of abstraction (Nilsson & Darley, 2006). Furthermore, there are some previous works that emphasize the results obtained by generative modeling for a given MES design. Vaario and Ueda (1998) proposed a modeling method for simulating dynamic scheduling where local attraction fields drive transporters carrying jobs to particular machines according to priority rules. The resulting emergence is generated by local interactions among agents through their force fields which cannot be anticipated without doing simulations. Ueda, Fujii, and Inoue (2004) introduced an emergent synthesis of an MES

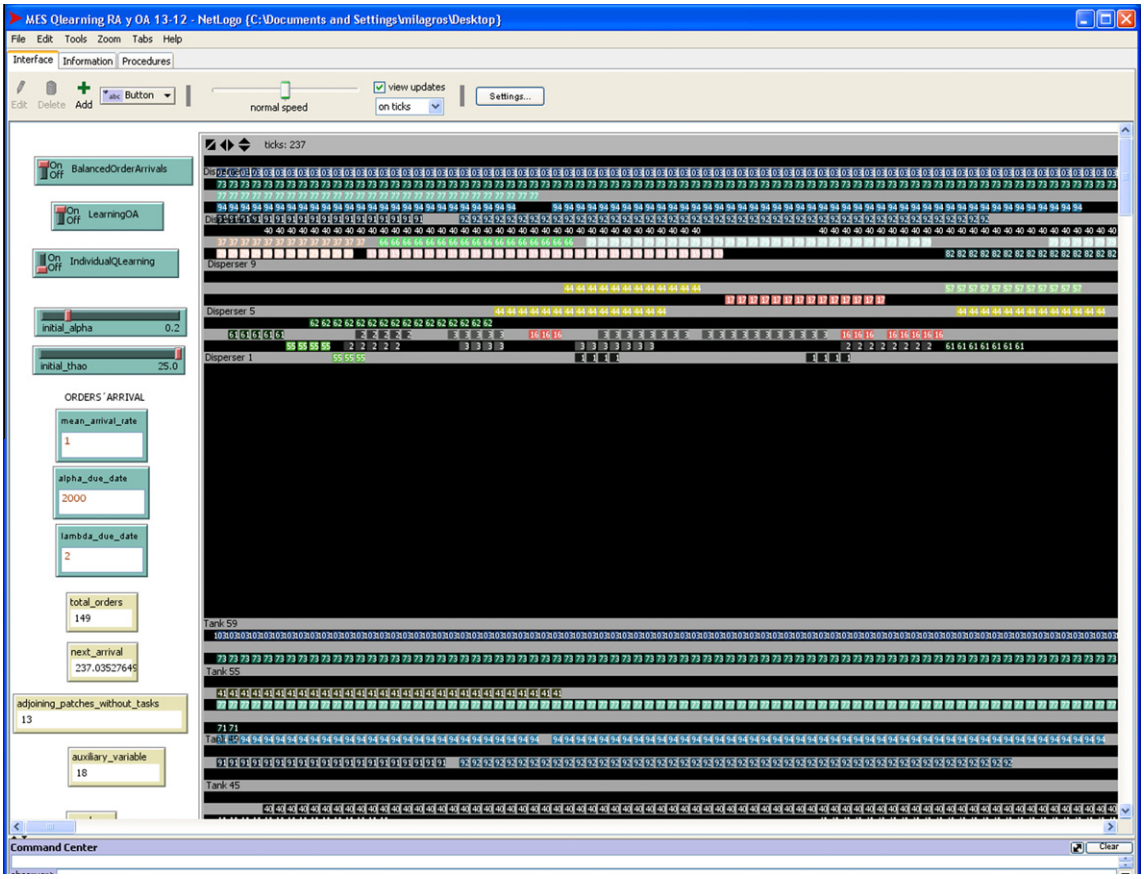

Fig. 10. Snapshot screen in the Netlogo prototype.

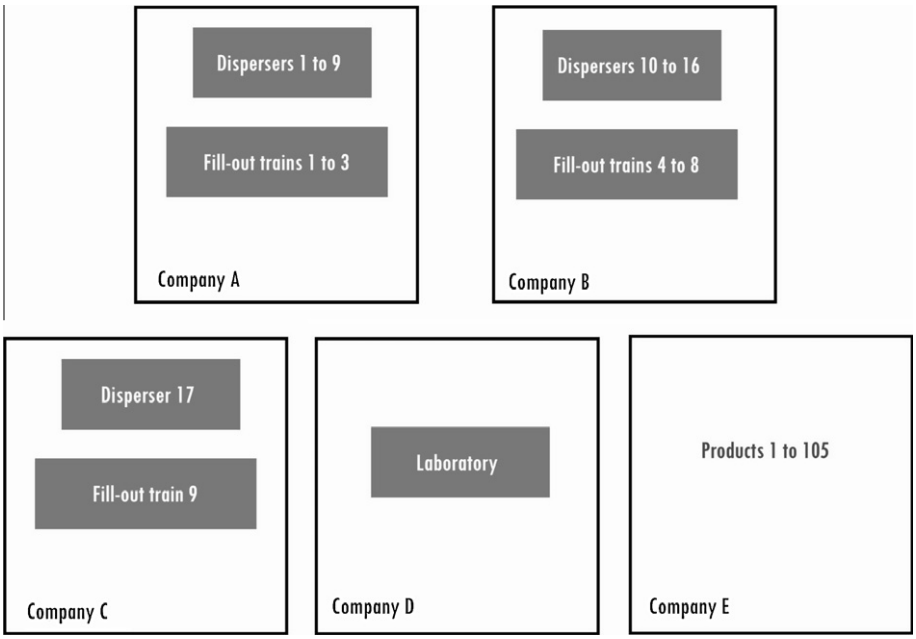

Fig. 11. Resource ownerships in the paint plant network.

decision-making policy based on a system model to predict the effect of alternative control and planning actions in *make-to-order* manufacturing environments. Emergence in this case was used to evaluate and control the range of time and cost constraints from customers that the system is able to fulfill. Generative simulation models are very natural means for understanding and designing complex adaptive systems such as the @MES, where the bottom-up design approach makes room for easy integration with the control/automation level. As a result, both scheduling and execution control in the @MES are bottom-up emergencies from interactions

among OAs and RAs in a mechanism designed to abstract relevant decisions to schedule and control a dynamic shop-floor environment (Rolón, Canavesio, & Martinez, 2009).

5.2. Prototype implementation in Netlogo

A computational model of the @MES was implemented in Netlogo® (Wilensky, 1999), a well-equipped software environment for modeling complex adaptive systems, or "synthetic worlds," and a useful tool in understanding emergent decentralized behaviors in multi-agent systems (Vidal, 2010). In Netlogo, a simulation model is made up of agents. Each agent develops a decision-making policy and deploys its role in an asynchronous manner regarding the behavior of other agents. In Netlogo worlds, there can be four types of agents: *turtles*, *patches*, *links*, and *the observer*. Turtles are agents that move around in the world. The world is typically two-dimensional (with an optional 3D application in recent versions of the language) and is divided up in a grid of patches. Each patch is a square piece of "ground" over which turtles can move. Links are agents that connect two turtles. The observer does not have a ground location, but it is constantly looking out over the world of turtles and patches.

To create the generative model in Netlogo a synthetic abstract world –where order and resource agents interact while learning to establish client–server relationships– has to be defined. Gantt charts are popular management tools and have found a role providing a quite useful interface using a time-phased dependent approach to production management in spite of dating back over a century (Wilson, 2003). Thus for the interaction mechanism in the @MES a dynamic Gantt chart was chosen to define the artificial environment, aptly named scheduling world, where OAs and RAs reflect their decisions and mutual influences (see Fig. 10). In the Gantt chart, each resource corresponds to a row and there is a dynamic temporal window with a fixed duration from the current

Table 3
Product types in the case study.

ID #: lot size (in liters)	Large family	Family	Pail type	Size (in liters)
1: 80, 2: 375	alkyds	varnishes	stin	0.25
3: 300, 4: 800, 5: 1300				0.5
6: 1000, 7: 1500				1
8: 800, 9: 1000, 10: 1800, 11: 2000, 12: 2600, 13: 3000, 14: 3500, 15: 3800				4
16: 150, 17: 800, 18: 1400, 19: 2400		enamels	plastic	1
20: 2000, 21: 2600, 22: 3000, 23: 3500, 24: 4000, 25: 4500, 26: 5000, 27: 5500				4
28: 2000, 29: 3000, 30: 4000				10
31: 2000, 32: 2600, 33: 3000, 34: 3500				20
35: 1300, 36: 1500, 37: 2000, 38: 2600, 39: 2800, 40: 3200		primers	tin	4
41: 2000, 42: 3000, 43: 4000				20
44: 540, 45: 1500		industrial coatings	plastic	1
46: 1000, 47: 2000, 48: 3000				4
49: 2400, 50: 4000, 51: 5500				10
52: 2400, 53: 3000, 54: 4000				20
55: 200, 56: 375, 57: 540, 58: 800, 59: 1000, 60: 1300			tin	0.25
61: 375, 62: 800, 63: 1300, 64: 1500				0,5
65: 1000, 66: 1500				1
67: 1000, 68: 2000, 69: 3000				4
70: 2400, 71: 3000, 72: 3500, 73: 4000				20
74: 375, 75: 540				1,5
76: 810, 77: 4050				2,7
78: 1300, 79: 2600				13
80: 2000, 81: 2400		pools	tin	4
82: 3200, 83: 3900				20
84: 3500, 85: 5500		demarcation	tin	20
86: 3000, 87: 4000				200
88: 48, 89: 150		anilines	envelope	0.06
90: 800, 91: 1000, 92: 1400, 93: 1800	latex	latex	tin	0,5
94: 1400, 95: 2600, 96: 3200, 97: 4000				1
98: 2000, 99: 2400, 100: 3000, 101: 3500				4
102: 2000, 103: 2400, 104: 3500, 105: 5000				20

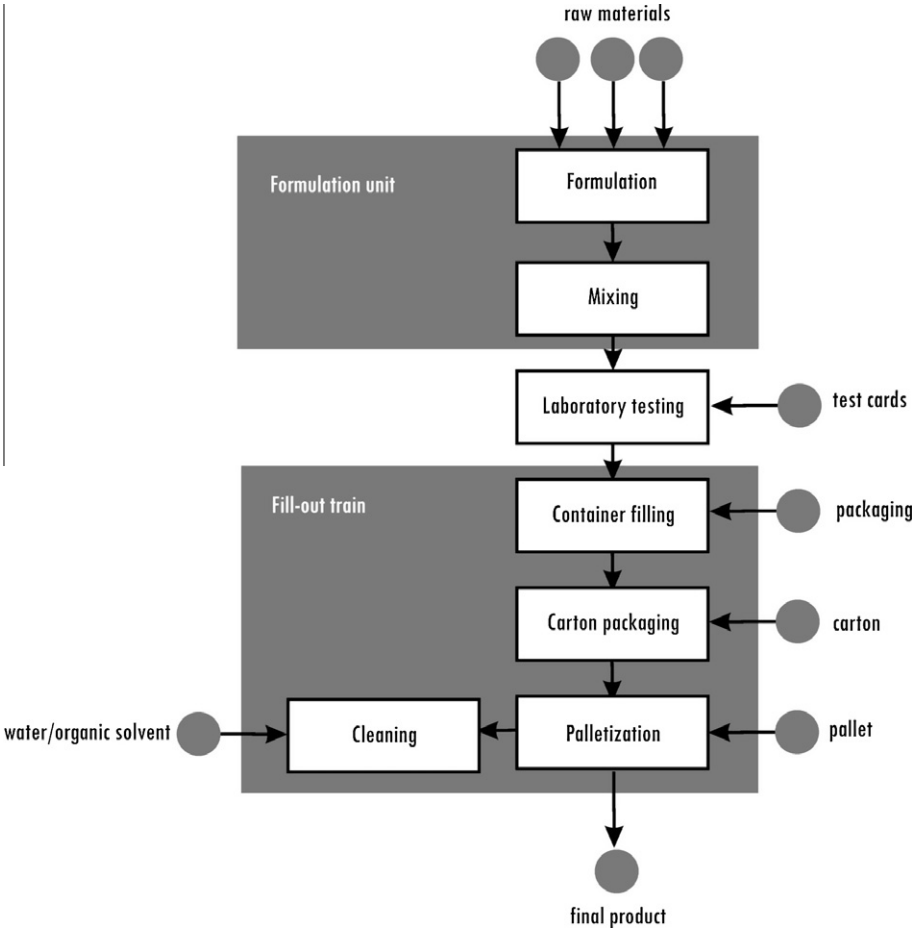

Fig. 12. Paint production process.

time into the future describing when the resource has been reserved for processing tasks. In the Gantt chart is also shown whether the resource is unavailable due an unplanned event or if it is in a *maintenance* state. OAs and RAs constantly watch over the Gantt chart to carry out their roles; an OA specifically looks up for an allocated processing route whereas each RA oversees already committed resource usage over the planning horizon.

Following a metaphor of communication used in biology and ethology where animals tend to behave collectively by using signals, Gantt charts are used in the @MES as a means to reflect the effects of interactions among OAs and RAs. In the simulated scheduling world interacting order and resource agents take decisions that update the Gantt chart to represent the state changes in shop-floor resources and orders as a result of usage commitments for planned orders and disruptions. The schedule is thus a result of such interactions and it is perceived in the dynamic Gantt chart. The above mentioned interactions between the managed elements (orders and resources) are realized through their autonomic managers, that is, order processing in each shop-floor environment is controlled in a decentralized way by interacting autonomic OAs and RAs acting locally and learning to achieve their specific goals.

For model construction, global variables were used whenever the information must be shared (available to every agent), and turtle variables and patch variables when they were solely private information for a single agent or type of agents. The different tasks involved in order processing gave rise to a number of mobile agents (turtles), one for each time unit, that move over a grid of stationary agents (patches) which represents resource usage times. Sets of agents (agentsets) were used to differentiate between order types based on the corresponding production formula and order attributes. Each order has different attributes such as product type, due date and required quality whereas its arrival times are stochastic. A batch (order) can then follow many different routes

Table 4
Capabilities in formulation units.

Formulation unit (tank) ID #	Capacity		Large family	Disperser ID #
	Minimum (in liters)	Maximum (in liters)		
1–4	40	200	Alkyds	1
5–9	100	500		2
10–14				3
15–18	500	1000		4
19–21				5
22–24				6
25–28				7
29–32				8
33–34				9
35–37	800	4000		10
38–40				11
41–44				12
45–47			Latex	13
48–49				14
50	2000	5000	Alkyds	15
51			Latex	14
52–53	4000	8000	Alkyds	15
54			Latex	16
55–56	6000	20000		
57–59			Alkyds	17

Table 5
Laboratory mean processing time for each product type.

Product ID #	Large family	Family	Mean (in min.)
1–15	Alkyds	Varnishes	10
16–34		Enamels	30
35–43		Primers	10
44–79		Industrial coatings	60
80–83		Pools	30
84–87		Demarcation	30
88–89		Anilines	30
90–105	Latex	Latex	20

Table 6
Capabilities for fill-out trains.

Fill-out train ID #	Capacity		Pail type
	Minimum (in liters)	Maximum (in liters)	
1	40	1000	Envelope
2	40	1000	Tin
3	40	1000	Plastic and tin
4	500	10000	Tin
5	500	10000	Plastic and tin
6	500	10000	Tin
7	40	4000	Tin
8	40	4000	Plastic and tin
9	10000	20000	Tin

Table 7
Products that can be processed by interconnected formulation units and fill-out trains #1 through #4.

Formulation unit (tank) ID #	Fill-out train ID #			
	1	2	3	4
1	88, 89			
2		1, 55		
3			1, 16, 55, 88, 89	
4		1, 55		
5	89	2, 3, 55, 56, 61, 74		
6				
7–8				
9–13			2, 3, 16, 55, 56, 61, 74, 89	
14				
15–21		4, 6, 8, 9, 57, 58, 59, 62, 75, 76		
22–24		4, 6, 8, 9, 57, 58, 59, 62, 75, 76	17, 44, 46	4, 6, 8, 9, 57, 58, 59, 62, 65, 67, 75, 76
25				
26–29		4, 6, 8, 9, 57, 58, 59, 62, 75, 76		
30–32				
33–34		90		
35–36				4–15, 35–40, 58–60, 62–70, 76–84
45–49				90–96, 98
52–53				41, 71, 72, 77, 85

Table 8
Products that can be processed by interconnected formulation units and fill-out trains #5 through #6.

Formulation unit (tank) ID #	Fill-out train ID #	
	5	6
35–41	4–15, 17–24, 28, 29, 31–40, 45–50, 52, 58–60, 62–70, 76, 78–84	11–15, 37–40, 68–70, 79–84
49		95–98
50		11–15, 37–41, 68–71, 77, 79–84
51		95, 96, 98
52	24–27, 29, 30, 41, 50, 51, 53, 71, 72, 77, 85	41, 71, 72, 77, 85
54		97, 99–103
55–56	30, 42, 54, 72	
57–58		97, 100–103

Table 9
Products that can be processed by interconnected formulation units and fill-out trains #7 through #9.

Formulation unit (tank) ID #	Fill-out train ID #		
	7	8	9
1–4	1, 55	1, 55	
5–14	2, 3, 55, 56, 61, 74	2, 3, 55, 56, 61, 74	
42–44	4–15, 37–40, 58, 59, 62, 68–70, 76, 79–84	4–6, 8, 9, 11–15, 19–24, 28, 29, 31–34, 37–40, 47–50, 52, 58, 59, 62, 68–70, 76, 79–84	
45–48	90, 95, 96, 98		
50	11–15, 37–40, 68–70, 79–84		
53		5, 24	
55–56			42, 43, 73, 86, 87
57–59			87, 104, 105

while using different pieces of equipment, and OAs and RAs have many possibilities for establishing client–server relationships.

The dynamic Gantt charts in Fig. 10 were made as follows. Each resource, grouped by resource type, has its own Gantt chart where it records its programmed tasks. Turtles (unit time tasks) are asked to move one unit to the left side and to disappear when they arrive to the left patch corresponding to the current time. Production operations run continuously whereas simulation time advances using equally distant event times every 10 min. Both the planning time horizon and the waiting time for arriving orders to be schedule before processing (i.e., the time window to allocate new orders in the detailed schedule) are determined by the user. Also the pre-booking duration intervals of orders in resource schedules and the maximum number of process options available for all planned orders can be defined by the user. Learning capabilities for both OAs and RAs can be set at the user screen, together with the chosen learning algorithm along with its parameters (see Fig. 10).

6. Simulation results

6.1. Case study

To illustrate the proposed approach a paint plant network based on the five-stage plant structure proposed in White (1989) with 80+ pieces of equipment was considered. In the plant network, there are different resource owners as shown in Fig. 11; each owner can voluntarily join or leave the network. There exists an enterprise (company E) handling only sales and logistics operations for all customer orders entering the network.

Product processing stages are: (1) formulation and mixing, (2) laboratory testing with product adjustment and release, (3) container filling, (4) carton packaging and (5) palletization. There are

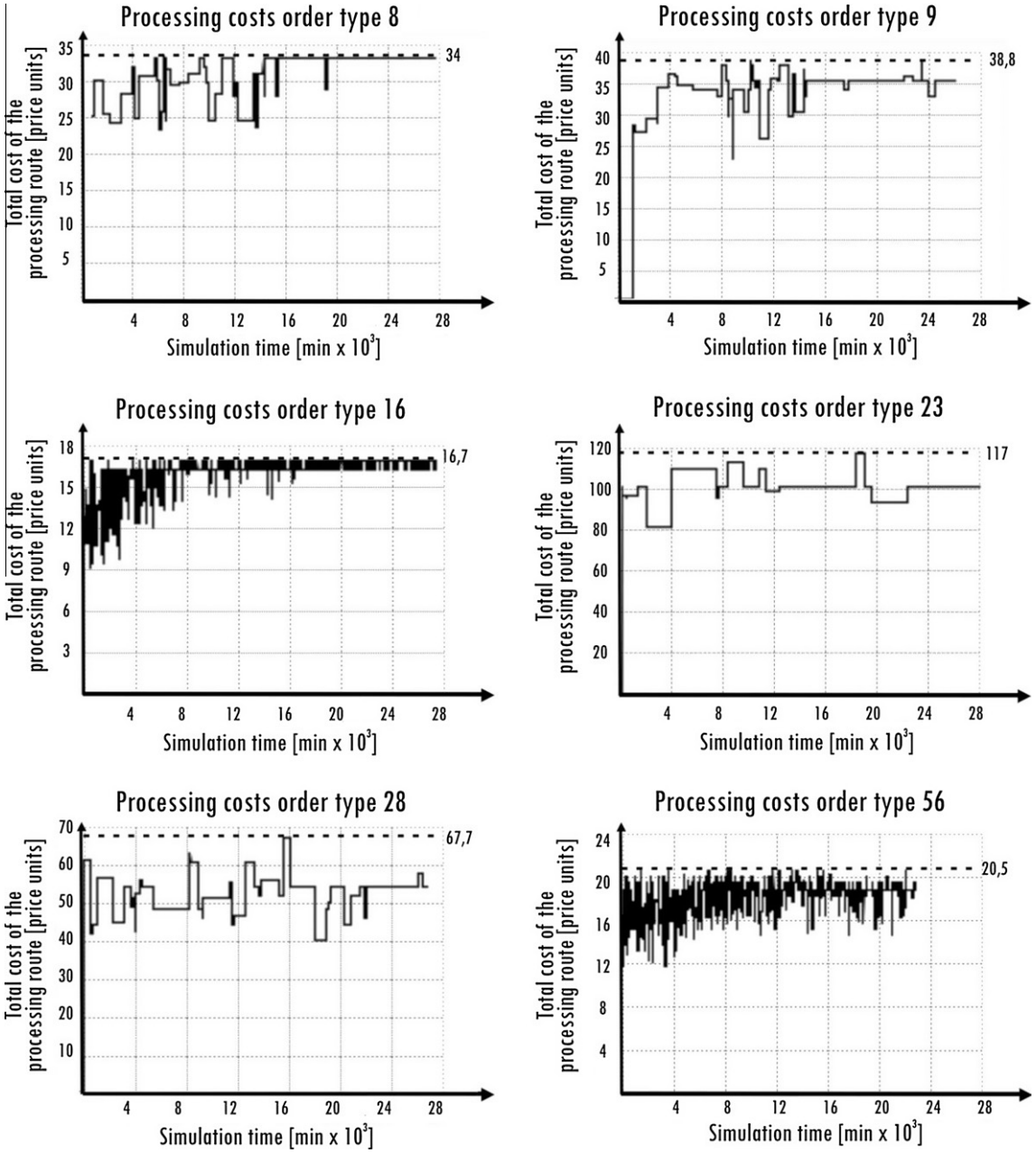

Fig. 13. Processing costs of some selected orders with Single-agent *Q*-learning implemented in the mechanism.

100+ products differing mostly by product types (alkyds and latex or water-based), product families (with identical characteristics and the same color, the container size being the only distinctive feature), packaging type, product size and lot size. See Table 3 for details.

The overall production recipe is given in Fig. 12. Formulation and mixing steps are tightly integrated processing activities, and it is assumed that filling machines are somewhat connected to their respective packaging machine, palletizer and all necessary connecting conveyors which made up alternative "fill-out" trains. Set-up times for cleaning and line reconfiguration are all included in fill-out processing times. The network structure and the number of equipment items available to carry out each processing activity (59 formulation units, 1 laboratory and 9 fill-out trains) were taken from White (1989), whereas for defining processing times, an in-depth shop-floor study was made at a paint plant located in Paraná, Argentina.

Equipment items used for both alkyds and latex products are virtual formulation units of different tank sizes, with 11 units used for latex products and 48 units for alkyds. Each formulation unit is made up of a disperser and a tank. After processing a batch, the corresponding disperser is disconnected from the tank such that a different formulation unit can be made up with a different tank. Dispersers are rotating toothed plates equipped with electric motors of different power rates which determine formulation (processing) times: for the latex product line there are 4 dispersers available whereas for alkyds there are 13 dispersers as it is indicated in Table 4. Tanks sharing the same disperser can hold a batch as work-in-process but cannot be simultaneously used for (re)formulating another product batch. Each product batch is kept in its corresponding tank until the packaging operation is completed via a feasible fill-out train.

There is a linear dependence between the time consumed in processing steps at the formulation/mixing units with the

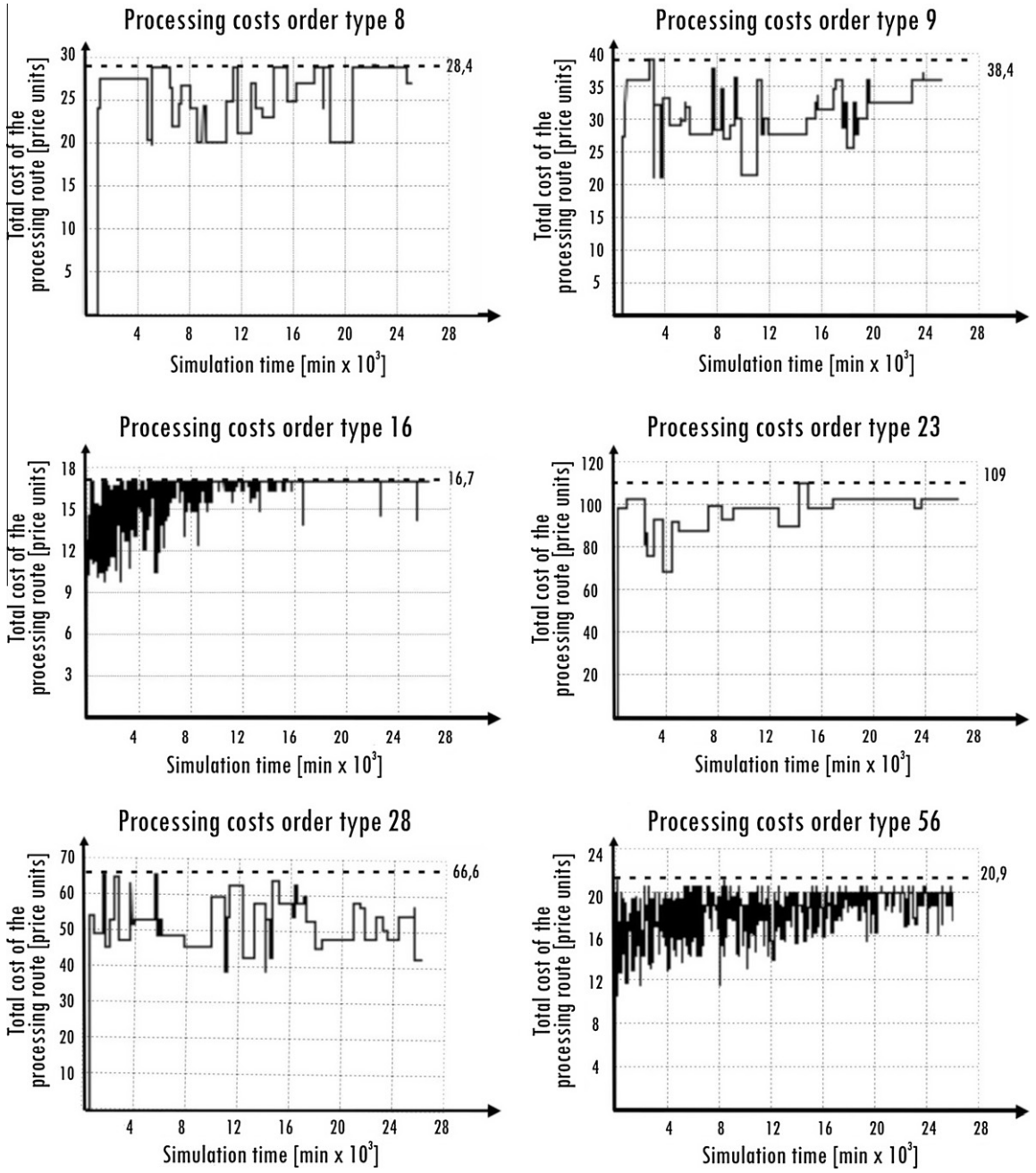

Fig. 14. Processing costs of some selected orders with individual *Q*-learning implemented in the mechanism.

corresponding batch size. For alkyds, it was assumed that the standard dispersing time is 0.3 + batch size $*$ 0.0015 h. For latex products, processing times are three times larger than for alkyds. Also, it is necessary to take into account variations of the disperser's power in calculating a formulation time. The laboratory has a capacity of 15 batches and testing results are only available after a laboratory processing time which follows an exponential distribution with a mean which is dependent on the product family (see Table 5). If a batch needs to be reformulated after its laboratory testing result is known, an adjustment operation is carried out and the corresponding reprocessing times are estimated as 25% of the original product formulation time. Thus, variations in the processing times of a product given by Eq. (6) above might be related to a change in the equipment item used for task processing, waiting times between subsequent resources, or rework operations is needed since a higher quality level is required. To move a batch out from a formulation unit

Table 10
Required quality levels for the products in the paint network.

Product type	Required quality level
Alkyds – varnishes	Q_{o1}–Q_{o15} = [0.2]
Alkyds – enamel	Q_{o16}–Q_{o34} = [0.3]
Alkyds – primers	Q_{o35}–Q_{o43} = [0.4]
Alkyds – industrial coatings	Q_{o44}–Q_{o79} = [0.5]
Alkyds – pools	Q_{o80}–Q_{o83} = [0.6]
Alkyds – demarcation	Q_{o84}–Q_{o87} = [0.7]
Alkyds – anilines	Q_{o88} and Q_{o89} = [0.8]
latex	Q_{o90} and Q_{o105} = [0.9]

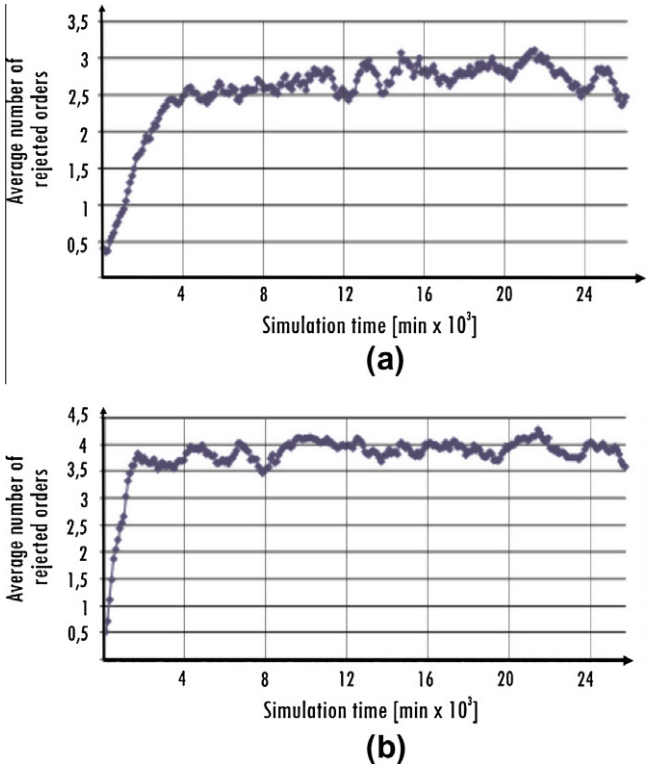

Fig. 15. Number of rejected orders. (a) Single-agent *Q*-learning; (b) Individual *Q*-learning.

(tank) for both alkyds and latex products in their corresponding packaging, fill-out trains with capabilities detailed in Table 6 are available.

Product batches are moved out from their tanks using alternative fill-out trains according to the equipment interconnections available at the network shop-floors and the constraints applying for the virtual formulation units (tank + disperser). See Tables 7–9 for details. Fill-out trains have speed variations based on the linear dependence 0.7 + batch size $*$ 0.0008 [h]. However, for filling machine #2 processing time is 10% lower, for filling machine #5 it is 15% higher, for filling machine #7 it is 29% lower and for filling machine #8 filling time it is 6% higher.

Orders arrive following an exponential distribution with a mean of 10 min for inter-arrival times. The pool of frequent orders includes orders # 1–3, 16, 44 and 55–57. Each frequent order type has a 6.20% arrival chance in the overall pool of arriving orders whereas non-frequent orders are also all equally probable with a probability of 0.52%. Due dates have a gamma distribution with shape parameter $\gamma = 13.\hat{8}$ days and scale parameter $b = 20$ min; each order's due date is added to its arrival time to define the absolute due date for the order. Quality levels (actions) that can be chosen by the RAs in shop-floors are determined in all cases by the vector $U_{r_*} = [0.0\ 0.1\ 0.2\ 0.3\ 0.4\ 0.5\ 0.6\ 0.7\ 0.8\ 0.9]$. That is, all RAs can choose quality levels ranging from 0.0 to 0.9 for all their processing tasks.

All tasks needed to process an order must comply with the required quality levels depending on the product type which are listed in Table 10. A unit processing price of $k_r = 1$ was used for all tasks in all equipments items. The planning time horizon is 10 days and orders must be scheduled at least one hour before its processing begins at the shop-floor. Pre-booking of orders in resource schedules lasts one hour and their reserved time slots are blocked so the corresponding RAs are not allowed to accept additional pre-bookings at their resource schedules during that time period. A maximum of 50 options for defining the processing route of any order was considered in order to avoid a combinatorial problem. Regarding the task insertion goal used by RAs, all of them use the FIFO dispatching rule (allocating tasks in the first available space of their local schedule).

It is difficult to predict the optimal actions that correspond to optimal matching between OAs and RAs without using a generative simulation model. Without a detailed simulation model, it is also difficult to determine when a set of orders will flow relatively smoothly through shop-floor in the enterprise network. So, the objective in this case study is to predict how the plant will operate under the proposed mechanism comparing Single-agent Q-learning and Individual Q-learning rules implemented in OAs and RAs. Both algorithms are studied in a normal operating scenario and in an abnormal condition where a breakdown of a fill-out train took place.

6.2. Simulation results

Performance indices considered in generative simulation of the proposed interaction mechanism for the @MES are: processing

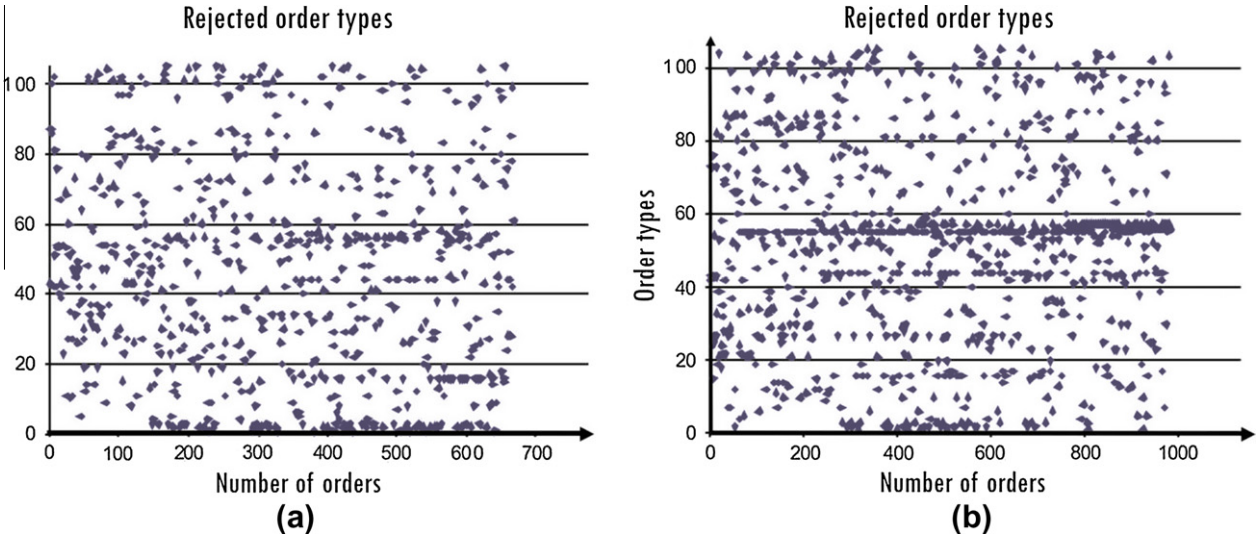

Fig. 16. Types of rejected orders. (a) Single-agent *Q*-learning; (b) Individual *Q*-learning.

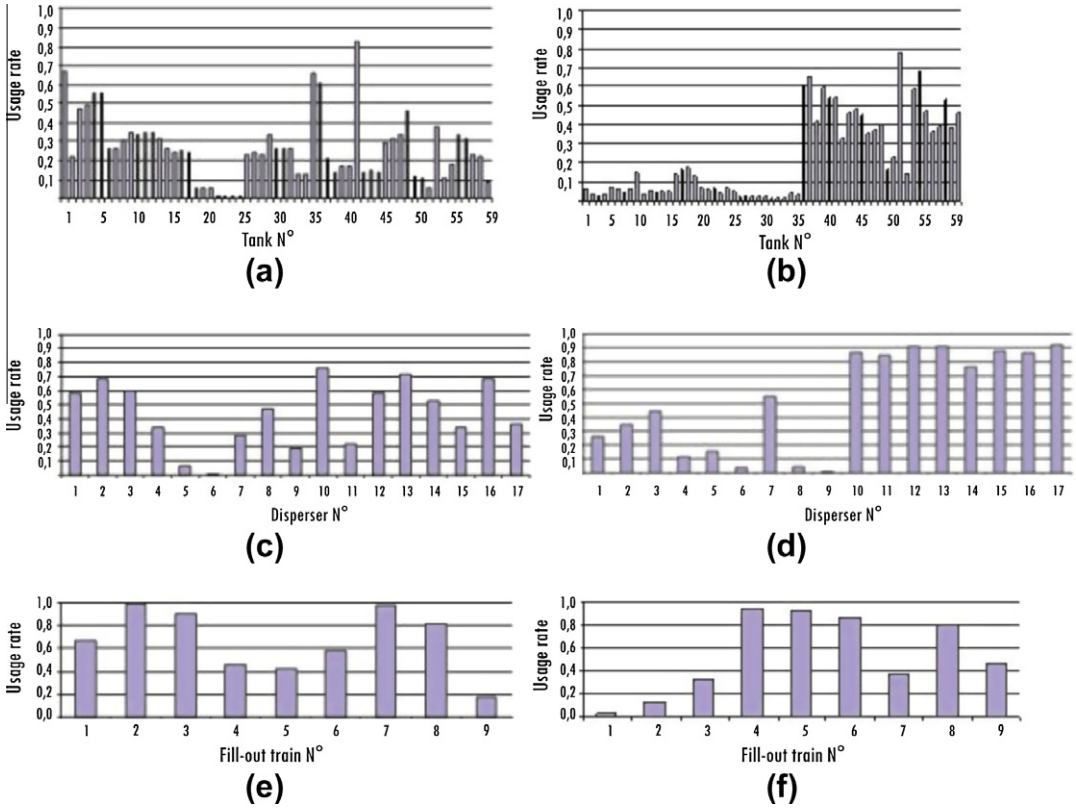

Fig. 17. Equipment utilization. (a) Tank utilization with Single-agent *Q*-learning; (b) Tank utilization with Individual *Q*-learning. (c) Disperser utilization with Single-agent *Q*-learning; (d) Disperser utilization with Individual *Q*-learning. (e) Fill-out train utilization with Single-agent *Q*-learning; (f) Fill- out train utilization with Individual *Q*-learning.

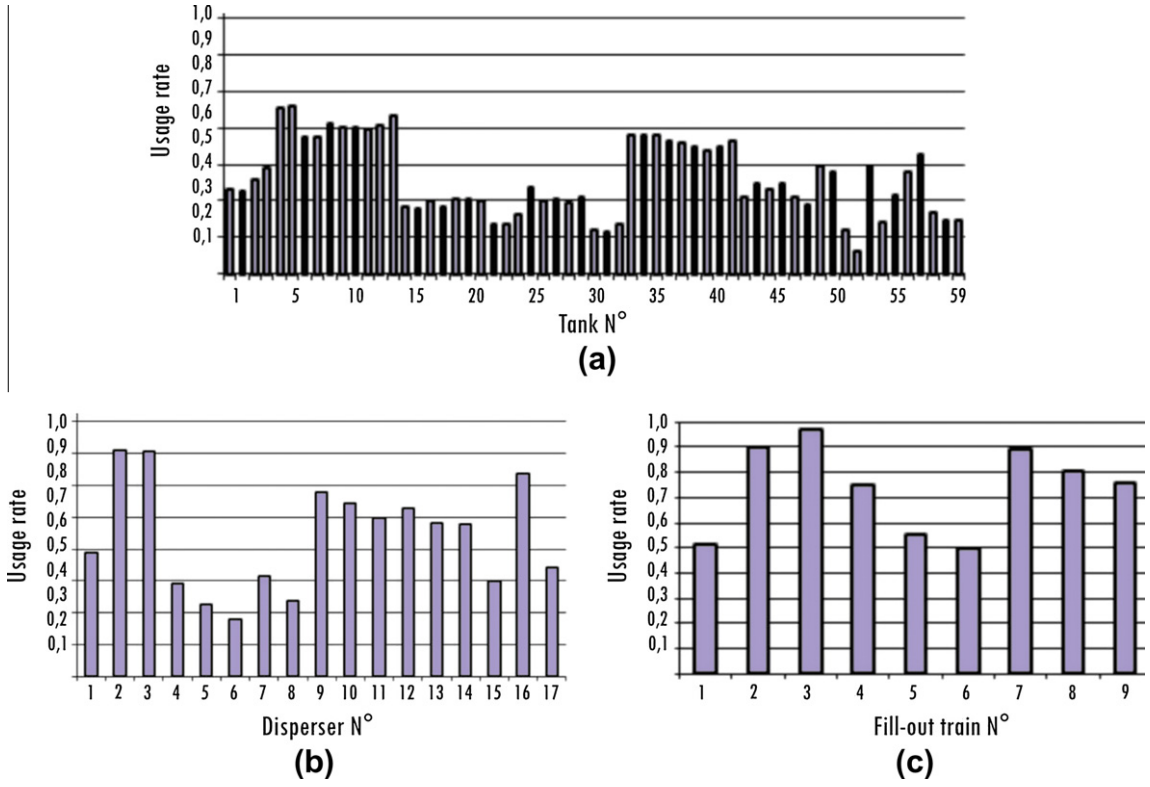

Fig. 18. Equipment utilization without agent learning. (a) Tank utilization; (b) Disperser utilization; (c) Fill-out train utilization.

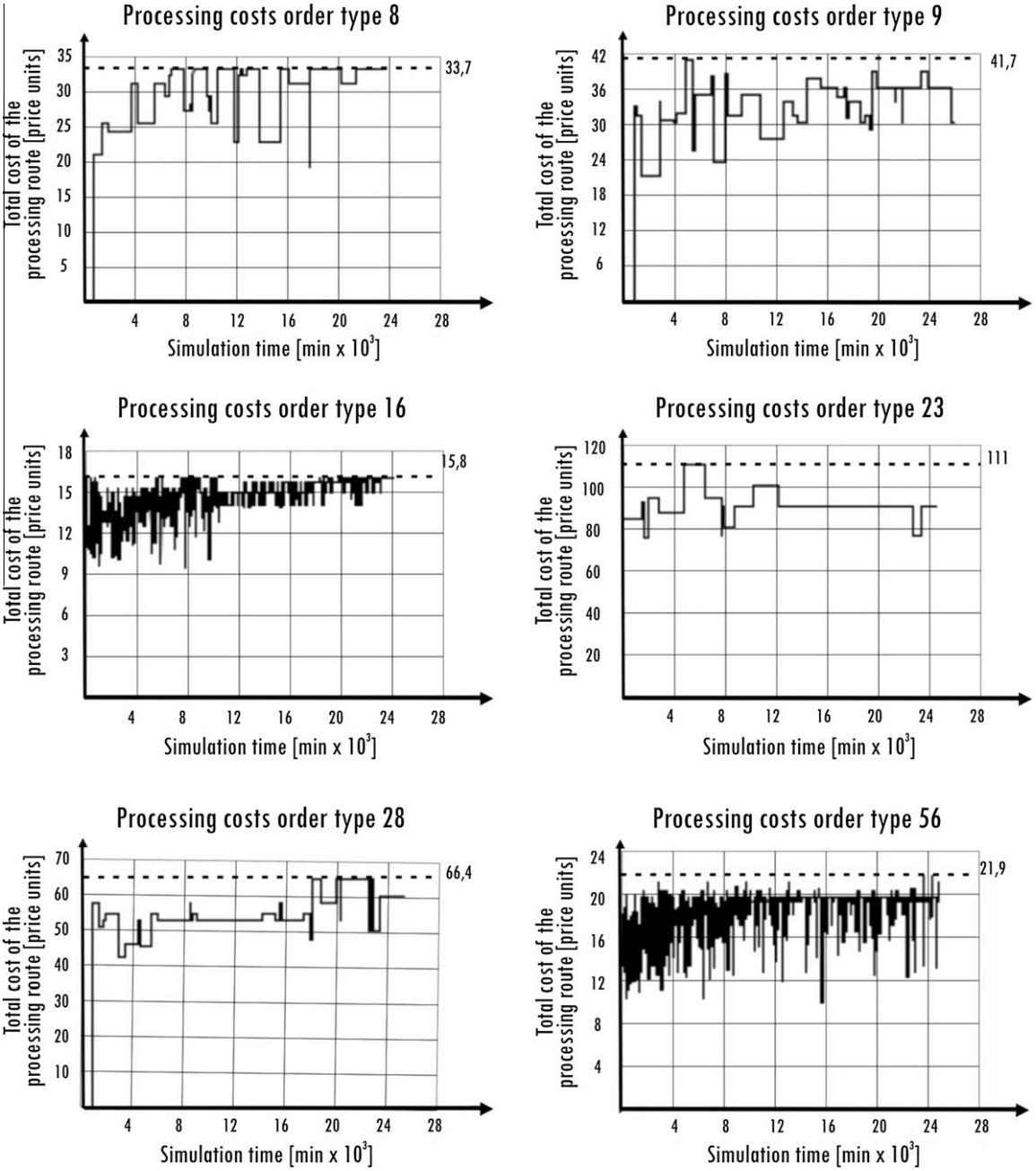

Fig. 19. Processing costs of some selected orders for the faulty scenario with Single-agent *Q*-learning.

routes at the end of the learning process, processing costs for each route, number and types of rejected orders and equipment utilization. In all runs, 259,200 min (6 months) of the plant operation were simulated. Both Single-agent Q-learning and Individual Q-learning were compared in a normal operating scenario and in a scenario with a breakdown of fill-out train #7.

Each OAs in the generative model manages only order type out of the 105 product types of the case study and their rewards are given by Eq. (8). On the other hand, RAs represent the 17 dispersers and the 9 fill-out trains with rewards given by Eq. (5). Tanks are considered merely work-in-process hold-ups without decision-making capabilities on their own. The laboratory is managed by an agent that has all the characteristics of a resource agent but without a learning capability (learning is not a necessity for this resource because there are no other laboratories to compete with).

A *Softmax* policy was used to balance exploration and exploitation. For the RAs in the Single-agent *Q*-learning, the temperature value τ was updated according to Eq. (8), where t is the time step, τ_0 = average processing times of all the tasks and $c = 1000$. For the OAs, τ_0 was initially set to a value of 25 whereas $c = 1000$. In the Individual *Q*-learning implementation, for the RAs $\tau = \tau_0$ = average processing times of all tasks, whereas for the OAs $\tau = \tau_0 = 25$.

For both algorithms the learning rate α_t was updated by Eq. (7), $\alpha_0 = 0.3$ and $m = 1000$. *Q*-values were initialized to zero for all OAs, and according to $Q_o = \tau_0(1 + 0.1 \times random)$ for all RAs, where *random* is a floating random number between 0 and 1.

Figs. 13 and 14 show the total processing costs of some randomly selected orders for the normal operating scenario. Orders #16 and #56 are representative examples of orders having high order arrival rates whereas the rest are non-frequent orders. The

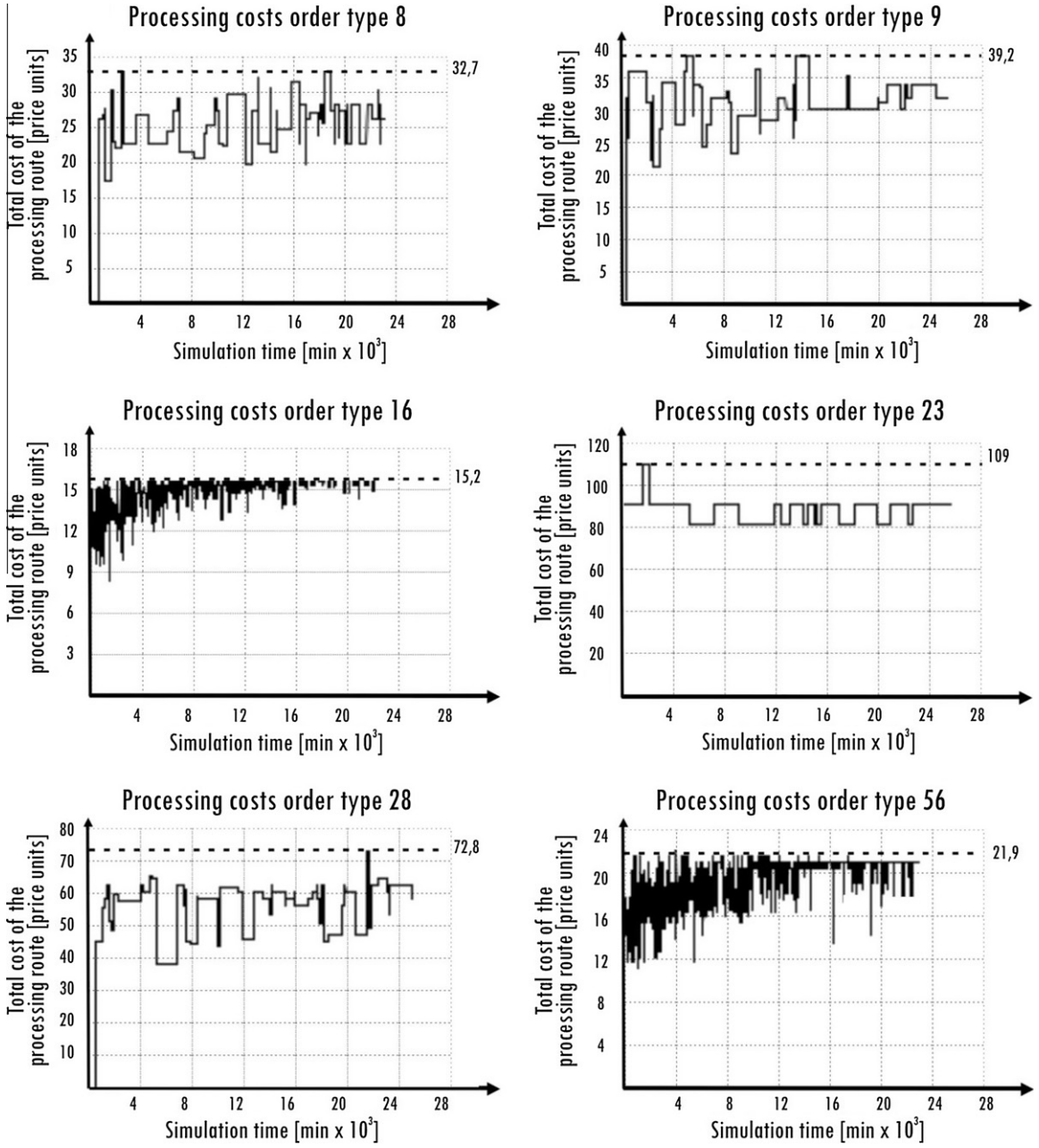

Fig. 20. Processing costs of some selected orders for the faulty scenario with individual *Q*-learning.

total processing cost is made up of the sum of the prices in the formulation unit and selected fill-out train for the respective product.

It is worth noting that the variations in the total processing costs of orders correspond to a change in the equipment used to execute a processing task or a rework operation due to a resulting product quality lower than the one required. As can be seen in Figs. 13 and 14, the total processing costs of the frequent orders in the Individual *Q*-learning implementation have a lower variability than in the Single-agent *Q*-learning. Processing costs stabilize at a high level in order #16 due to the fact that it can only be processed in fill-out train #3 (see Tables 8–10) and thus the greatest profits for the RAs are associated with higher processing costs. Variations in the Individual Q-learning implementation correspond to order route changes as the frequently selected resources are subjected to an overload. This is a very important outcome which highlights that the interaction mechanism allows stable client–server relations between OAs and RAs despite the total autonomy given to interacting agents and the lack of a master schedule. For non frequent order types, the learning curve is less steep and there are not significant differences between the two learning algorithms.

In Fig. 15, the number of total rejected orders is depicted for both agent learning rules. It is shown that rejected orders for the Individual *Q*-learning is 47% higher (982 rejected orders) than the number of rejected orders when the single *Q*-learning rule is used (668 rejected orders) over the simulation period. After 20 days following plant operation start-up, the number of rejected orders remained stable throughout the simulation runs regardless of which learning algorithm is used.

Rejected orders by type all over the simulation period are shown in Fig. 16. Single-agent *Q*-learning presents a more equitable distribution of the rejected order types than the Individual *Q*-learning. This multi-agent learning algorithm has a higher rejection rate of frequent orders by biasing the mechanism to look up for optimal matching between RAs and OAs at the cost of processing

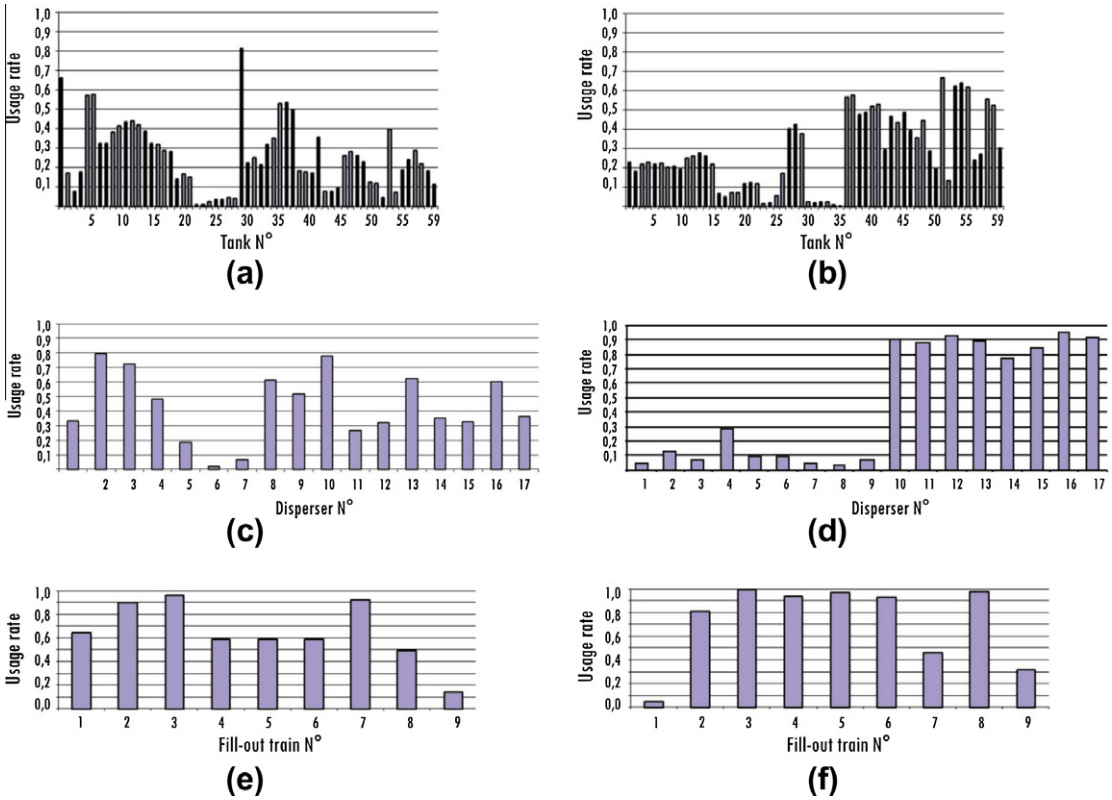

Fig. 21. Equipment utilization. (a) Tank utilization with Single-agent Q-learning; (b) Tank utilization with Individual Q-learning. (c) Disperser utilization with Single-agent Q-learning; (d) Disperser utilization with Individual Q-learning. (e) Fill-out train utilization with Single-agent Q-learning; (f) Fill- out train utilization with Individual Q-learning.

significantly fewer orders than the Single-agent *Q*-learning in the same simulation time interval.

As far as equipment utilization is concerned, resource usage pattern using Individual *Q*-learning algorithm does not favor balancing as it is shown in Fig. 17. Simulation results reflect that the selectivity of the mechanism that does not encourage order execution in any resource but prefer the ones with the appropriate quality level for each product and with a high reliability rate for processing orders of the corresponding order type. More specifically, in case of dispersers and consequently in the corresponding tanks, there are some equipment items that are chosen much more frequently by OAs in route selection. For fill-out trains, differences in their utilization are less important when comparing alternative learning rules but load distribution is critical due to the high usage rates of fill-out trains which are actually the throughput bottleneck in the enterprise network. In the Single-agent *Q*-learning implementation, a rather uniform equipment usage distribution is obtained, possible due to a lack of selectively in the learning rule. To reinforce this remark, the interaction mechanism without learning in the OAs and RAs give rises to a quite even distribution of the workload, as can be seen in Fig. 18.

Figs. 19 and 20 show the total processing costs of the selected orders when the fill-out train #7 experiences an unplanned failure between the simulated time 15.000 and 20.000. As it is depicted in these figures, variations in the processing costs of the frequent orders are more important for the Single-agent *Q*-learning. This fact is especially observed in frequent orders such as orders #2 and #16. It is worth noting that the number of rejected orders is 3% higher regardless which learning algorithm is used whenever arrival distributions for frequent orders are kept constant.

Utilization rates of working fill-out trains are little affected by the disruption in #7. A desirable emergence of the mechanism is that the workload is evenly distributed to compensate for the unavailability of fill-out train #7 (see Fig. 21). This is an important emergence of agent interactions that suggests that intelligent agents are able to adapt their policies and therefore enterprise networking is not affected by selfish behavior and asynchronous learning.

7. Concluding remarks

Requirements for MESs in enterprise networking include a distributed control system performed by autonomous decisional entities with learning capabilities, private (local) information, temporal client–server relationships according to order demands and available resources, and automatic local rescheduling triggered by disruptive events. Selfish behavior and decision policy adaptation was introduced in the @MES for tayloring it to an enterprise network environment.

Learning capabilities in the @MES results in a complex adaptive system where emerging behaviors cannot be predicted by analytical tools but through generative simulation. Therefore, an agent-based simulation model of an @MES based on the autonomic proposal augmented with selfish behavior and learning capabilities for enterprise networking was presented in this work. Emergent behaviors of the interaction mechanism highlighted the importance of generative simulation in designing complex adaptive systems such as distributed production control systems for enterprise networking.

Simulated experiments were used to compare the performance of Single-agent *Q*-learning and Individual *Q*-learning in the @MES interaction mechanism for a multiproduct plant network. Results obtained for different simulation runs of the Netlogo simulation

model in a normal operating scenario and in an abnormal scenario with a breakdown of a resource show that despite selfish behavior and autonomy of the individual agents, the @MES is able to work properly and to accommodate to learning incorporation while allowing the establishment of optimal client–server relationships between order and resource agents. It was also demonstrated by simulation results that the Individual *Q*-learning algorithm exhibits a higher performance than the Single-agent *Q*-learning in both scenarios, although resulting in a more selective mechanism for order acceptance reflected in unbalanced resource usage and high order rejection rates.

Our research work is now driven towards the incorporation of a higher level of complexity and realism in the decision logic of both the OAs and RAs. On this basis, RAs could consider reducing the price for resource utilization per time unit when a lower threshold in its usage rate is achieved. Besides, OAs could take into account the marginal contribution of each order when considering order acceptance or rejection decisions due to resource congestion. Current research efforts also includes the design and implementation of a reinforcement learning algorithm that will take into account some communication channels among interacting agents without revealing their strategies. This information sharing mechanisms is intended to accelerate the process of updating each agent's policy when new clients or servers join or leave the network. The learning algorithm having communication capabilities will be compared with the Individual *Q*-learning in different scenarios to guide the design of an information sharing mechanism.

Introducing cognitive capabilities (such as learning) in order and resource agents is mandatory for the @MES to implement *The Cognitive Factory* vision (ElMaraghy, 2009). As it has been highlighted recently (Brecher, 2012; Jovane, Westkämper, & Williams, 2009; Veres, 2011), collaboration between cognitive abilities of humans and artificial agents is key to the competitiveness of many enterprises in the developed world, and the only way forward to drastically increase product value at low costs in the era of mass customization. As a result, the human-agent-machine interface is of paramount importance to grasp the full potential of the @MES for automating tasks that require collaborative cognitive capabilities at the shop-floor. As a demonstrative example, the reader is referred to the work of Veres (2011) where the novel idea of "publishing for machines" is discussed. In this regard, the use of natural language programming and writing of documents that machines and artificial agents can read and utilize to improve their feedback control skills, their knowledge of the environment, and also their decision-making skills, is proposed. Summing up, agent learning in the @MES is a significant breakthrough to enter a new era of shop-floor automation where cognition is the name of the game.

References

Anussornnitisarn, P., Nof, S., & Etzion, O. (2005). Descentralized control of cooperative and autonomous agents for solving the distributed resource allocation problem. *International Journal of Production Economics, 98*, 114–128.

Azadegan, A., & Dooley, K. (2010). *Complexity and the rise of distributed control in operations management*. *SAGE handbook of complexity and management*. London: Sage Publications (pp. 420–437).

Bauer, M., & Stoeter, S. (2010). Enterprise connectivity. In *Collaborative process automation systems* (pp. 1036–1043). ISA.

Benaïm, M., & Hirsch, M. (1999). Mixed equilibria and dynamical systems arising from fictitious play in perturbed games. *Games and Economic Behavior, 29*, 36–72.

Blanc, P., Demongodin, I., & Castagna, P. (2008). A holonic approach for manufacturing execution system design: An industrial application. *Engineering Applications of Artificial Intelligence, 21*, 315–330.

Blumenthal, R. (2004). Manufacturing execution systems to optimize the pharmaceutical supply chain. *Pharmind, Pharmazeutische Industrie, 11a*, 1414–1424.

Brecher, C. (Ed.). (2012). *Integrative production technology for high-wage countries*. Berlin, Heidelberg: Springer-Verlag.

Camarinha-Matos, L. (2001). Execution system for distributed business processes in a virtual enterprise. *Future Generation Computer Systems, 17*, 1009–1021.

Canavesio, M., & Martinez, E. (2007). Enterprise modeling of a project-oriented fractal company for SMEs networking. *Computers in Industry, 54*, 794–813.

Chaharsoogui, S., Heydari, J., & Zegorki, S. (2008). A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems, 45*, 949–959.

Cheong, C., & Winikoff, M. (2006). Improving flexibility and robustness in agent interactions: Extending Prometheus with Hermes. *Software Engineering for Multi-Agent Systems, 6*, 189–206.

Chirn, J., & McFarlane, D. (2000). A component-based approach to the holonic control of a robot assembly cell. In Proceedings of the IEEE 17th international conference on robotics and automation, ICRA 2000.

Chung, W., Yam, A., & Chan, M. (2004). Networking enterprise: A new business model for global sourcing. *International Journal of Production Economics, 87*, 267–280.

Cicirello, V., & Smith, S. (2004). Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multi-Agent Systems, 8*, 237–266.

Covanich, W., & McFarlane, D. (2009). Assessing ease of reconfiguration of conventional and holonic manufacturing systems: Approach and case study. *Engineering Applications of Artificial Intelligence, 22*(7), 1015–1024.

ElMaraghy, H. (2009). *The cognitive factory. In changeable and reconfigurable manufacturing systems* (pp. 1015–1024). London: Springer.

Epstein, J. M. (2006). *Generative social science: Studies in agent-based computational modeling*. Princeton: Princeton University Press.

Epstein, J., & Axtell, R. (1996). *Growing artificial societies. Social science from the bottom-up*. Washington: Brooking Institution Press.

Ferber, J. (1999). *Multi-agent systems. An introduction to distributed artificial intelligence*. Boston: Addison-Wesley.

Gilbert, N. (2008). *Agent-based models*. London: Sage Publications.

Grant, E., & Leavenworth, R. (1996). *Control Estadístico de Calidad*. Mexico D.F.: Compañía Editorial Continental.

Grossmann, I. (2009). Research challenges in planning and scheduling for enterprise-wide optimization of process industries. In *10th International symposium on process systems engineering: Part A* (pp. 15–21).

Hadeli Valckenaers, P., Kollingbaumb, M., & Van Brussel, H. (2004). Multi-agent coordination and control based on stigmergy. *Computers in Industry, 53*, 75–96.

Harjunkoski, I., Nyström, R., & Horch, A. (2009). Integration of scheduling and control – Theory or practice? *Computers and Chemical Engineering, 33*(12), 1909–1918.

Hofbauer, J., & Hopkins, E. (2005). Learning in perturbed asymmetric games. *Games and Economic Behavior, 52*, 133–152.

Huang, B. et al. (2002). A framework for virtual enterprise control with the holonic manufacturing paradigm. *Computers in Industry, 49*(3), 299–310.

IBM Corporation (2006). An architectural blueprint for autonomic computing. < http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf> Accessed 30.05.12.

Jovane, F., Westkämper, E., & Williams, D. (2009). *The manufuture road: Towards competitive and sustainable high-adding-value manufacturing*. Berlin: Springer Verlag.

Kaihara, T., Fujii, N, Toide, S., Ishibashi, H., & Nakano, T. (2010). A proposal of socio-inspired manufacturing scheduling concept and its application into flexible flowshop. In *Proceedings of the 43rd CIRP conference on, manufacturing systems* (pp. 813–820).

Kephart, J., & Chess, D. M. (2003). The vision of autonomic computing. *Computer, 36*(1), 41–50.

Kinder, T. (2003). Go to the flow – A conceptual framework for supply relations in the era of the extended enterprise. *Research Policy*, 503–523.

Kletti, J. (2007). *Manufacturing execution systems – MES*. Berlin: Springer-Verlag.

Leitão, P., Colombo, A., & Restivo, F. (2005). ADACOR: A collaborative production automation and control architecture. *IEEE Intelligent Systems, 20*, 58–66.

Leslie, D., & Collins, E. (2005). Individual *Q*-learning in normal form games. *SIAM Journal on Control and Optimization, 44*, 495–514.

Mallidi, K., Praskevopoulos, A., & Paganelli, P. (1999). Process modelling in small–medium enterprise networks. *Computers in Industry, 38*, 149–158.

Mannor, S., & Shamma, J. (2007). Multi-agent learning for engineers. *Artificial Intelligence, 171*, 417–422.

McClellan, M. (2000). *Applying manufacturing execution systems*. New York: McGraw-Hill.

McClellan, M. (2003). *Collaborative manufacturing – Using real-time information to support the supply chain*. New York: St. Lucie Press.

Meyer, H., Fuchs, F., & Thiel, K. (2009). *Manufacturing execution systems (MES): Optimal design, planning, and deployment*. New York: McGraw-Hill.

Mezgár, I., Kovács, G., & Paganelli, P. (2000). Co-operative production planning for small- and medium-sized enterprises. *International Journal of Production Economics, 64*, 37–48.

Miller, J., & Page, S. (2007). *Complex adaptive systems: An introduction to computational models of social life*. Princeton: Princeton University Press.

Nilsson, F., & Darley, V. (2006). On complex adaptive systems and agent-based modeling for improved decision-making in manufacturing and logistics settings. *International Journal of Operations and Production Management, 26*, 1351–1373.

North, M., & Macal, M. (2007). *Managing business complexity, discovering strategic solution with agent-based modeling and simulation*. Oxford: Oxford University Press.

Rolón, M., & Martinez, E. (2010). Agent-based simulation modeling of an interaction mechanism for detailed design of autonomic manufacturing execution systems.

In *Proceedings of the 43rd CIRP international conference on, manufacturing systems* (pp. 1036–1043).
Rolón, M., Canavesio, M., & Martinez, E. (2009). Agent based modeling and simulation of intelligent distributed scheduling systems. *Computer Aided Chemical Engineering, 26*, 985–990.
Rolón, M., & Martinez, E. (2012). Agent-based modeling and simulation of an autonomic manufacturing execution system. *Computers in Industry, 63*, 53–78.
Schelling, T. C. (1978). *Micromotives and macrobehavior*. New York: Norton.
Shohan, Y., & Leyton-Brown, K. (2009). *Multiagent systems. Algorithm, game-theoretic and logical foundations*. New York: Cambridge University Press.
Smith, R. (1980). The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transactions on Computers, C-29*(12), 1104–1113.
Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction* (pp. 24–49). Cambridge: The MIT Press.
Tran, T. (2010). Protecting buying agents in e-marketplaces by direct experience trust modelling. *Knowledge and Information Systems, 22*(1), 65–100.
Trentesaux, D. (2009). Distributed control of production systems. *Engineering Applications of Artificial Intelligence, 22*, 971–978.
Ueda, K. (1992). A concept for bionic manufacturing systems based on DNA-type information. In *Proceedings of the IFIP TC5/WG5.3 eight international PROLAMAT conference on human aspects in computer integrated manufacturing, B-3* (pp. 853–863).
Ueda, K., Fujii, N., & Inoue, R. (2004). Emergent synthesis approaches to control and planning in make to order manufacturing environments. *CIRP Annals – Manufacturing Technology, 53*(1), 385–388.
Vaario, J., & Ueda, K. (1998). An emergent modelling method for dynamic scheduling. *Journal of Intelligent Manufacturing, 9*(2), 129–140.
Valckenaers, P., Van Brussel, H., Saint Germain, B., & Van Belle, J. (2010). Networked manufacturing control: An industrial case. In *Proceedings of the 43rd CIRP conference on manufacturing systems* (pp. 129–136).
Valckenaers, P., & Van Brussel, H. (2005). Holonic manufacturing execution systems. *CIRP Annals – Manufacturing Technology, 54*(1), 129–136.
Valckenaers, P., Van Brussel, H., Verstraete, P., Saint Germain, P., & Hadeli (2007). Schedule execution in autonomic manufacturing execution systems. *Journal of Manufacturing Systems, 26*, 75–84.
Valluri, A., & Croson, D. (2005). Agent learning in supplier selection models. *Decision Support Systems, 39*, 219–240.
Valluri, A., North, M., & Macal, C. (2009). Reinforcement learning in supply chains. *International Journal of Neural Systems, 19*(5), 331–344.
Van Brussel, H., Wyns, J., Valckenaers, P., & Bongaerts, L. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry, 37*, 255–274.
Veres, S. M. (2011). Knowledge of machines: Review and forward look. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering, 225*(1), 3–10.
Vernadata, F. (2010). Technical, semantic and organizational issues of enterprise interoperability and networking. *Annual Reviews in Control, 34*, 139–144.
Vidal, J. (2010). Fundamentals of multiagent systems with NetLogo examples. Unpublished textbook. <http://www.damas.ift.ulaval.ca/~coursMAS/ComplementsH10/mas-Vidal.pdf> Accessed 30.05.12.
Vlassis, N. (2007). *A concise introduction to multiagent systems and distributed artificial intelligence*. Amsterdam: Morgan and Claypool Publishers.
Wang, Y., & Usher, J. (2004). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence, 18*, 73–82.
Warnecke, H. J. (1993). *The fractal company: A revolution in corporate culture*. Berlin: Springer-Verlag.
Watkins, C. (1989). *Learning from delayed rewards*. PhD thesis. Cambridge University.
Westkamper, E., Huser, M., & Van Briel, R. (2000). Transicion processes: A survey of German industry. *Journal of Materials Processing Technology, 106*, 141–151.
White, C. (1989). Productivity analysis of a large multiproduct batch processing facility. *Computers and Chemical Engineering, 13*, 239–245.
Wiendahl, H., ElMaraghy, H., Nyhuis, P., Zäh, M., Wiendahl, H., Duffie, N., et al. (2007). Changeable manufacturing – Classification, design and operation. *CIRP Annals – Manufacturing Technology, 56*(2), 783–809.
Wilensky, U. (1999). Netlogo modeling environment. <http://ccl.northwestern.edu/netlogo/download.shtml> Accessed 30.05.12.
Wilson, J. (2003). Gantt charts: A centenary appreciation. *European Journal of Operational Research, 149*, 430–437.
Wu, N., & Su, P. (2005). Selection of partners in virtual enterprise paradigm. *Robotics and Computers-Integrated Manufacturing, 21*(2), 119–131.
Xu, W., Wei, Y., & Fan, Y. (2002). Virtual enterprise and its intelligence management. *Computers and Industrial Engineering, 42*, 199–205.