An MDP Model-Based Reinforcement Learning Approach for Production Station Ramp-Up Optimization: Q-Learning Analysis

Stefanos Doltsinis, Pedro Ferreira, and Niels Lohse, *Member, IEEE*

***Abstract*—Ramp-up is a significant bottleneck for the introduction of new or adapted manufacturing systems. The effort and time required to ramp-up a system is largely dependent on the effectiveness of the human decision making process to select the most promising sequence of actions to improve the system to the required level of performance. Although existing work has identified significant factors influencing the effectiveness of ramp-up, little has been done to support the decision making during the process. This paper approaches ramp-up as a sequential adjustment and tuning process that aims to get a manufacturing system to a desirable performance in the fastest possible time. Production stations and machines are the key resources in a manufacturing system. They are often functionally decoupled and can be treated in the first instance as independent ramp-up problems. Hence, this paper focuses on developing a Markov decision process (MDP) model to formalize ramp-up of production stations and enable their formal analysis. The aim is to capture the cause-and-effect relationships between an operator's adaptation or adjustment of a station and the station's response to improve the effectiveness of the process. Reinforcement learning has been identified as a promising approach to learn from ramp-up experience and discover more successful decision-making policies. Batch learning in particular can perform well with little data. This paper investigates the application of a Q-batch learning algorithm combined with an MDP model of the ramp-up process. The approach has been applied to a highly automated production station where several ramp-up processes are carried out. The convergence of the Q-learning algorithm has been analyzed along with the variation of its parameters. Finally, the learned policy has been applied and compared against previous ramp-up cases.**

***Index Terms*—Decision-making, learning systems, manufacturing automation, Markov processes.**

I. Introduction

TECHNOLOGY has been progressing at a very fast pace in recent years, which has brought great pressure for enterprises to incorporate it as soon as possible into new products. This has led to shorter product lifecycles and has provided significant challenges for their manufacturing in a timely manner. Within those lines ramp-up is constantly being highlighted by industry as one of the production phases with large potential for further improvement [1], [2]. This has led to the significant attention from the research community which provided interesting advances for the reduction of the ramp-up time [3], [4]

Terwiesch and Bohn [2] have defined ramp-up as the period between the end of product development and full capacity production. In order to get a system in its full capacity, human operators spend a lot of time tuning and debugging the processes. During ramp-up, a manufacturing system is adjusted and changed until it becomes sufficiently stable (disturbances reduced to minimum) and its production output reaches the desired level [5]. Ramp-up of modern manufacturing systems has turned into a very complex process, which commonly leads to long delays and an increased time-to-market. The human involvement highly affects the required time and their knowhow, and ability to make the correct decisions under uncertainty can make the required time vary significantly. Additionally, the human involvement in the decision making process and the highly uncertain environment, make the process both stochastic and unpredictable.

The knowledge is currently owned by the personnel involved in the ramp-up phase, which often makes it difficult to share and retain within an organization. Additionally, market pressures to quickly deliver products and reduce cost restricts the available time for experimentation. This hinders the complete understanding of the individual ramp-up behavior of a system. This paper focuses on capturing and analyzing the technical decision making process of operators during the ramp-up of automatic assembly workstations. The aim is to develop an approach that can monitor ramp-up in terms of the adjustment action taken and resulting process changes to build up a sufficient understanding of the process to later support operators with their decisions. Increased use of sensors in automation systems combined with more powerful embedded control systems make it plausible to investigate the application of machine learning approaches to support the ramp-up process and enable a more transferable knowledge retention. The challenge for machine learning during ramp-up is the limited similarity between different manufacturing systems. This is the critical challenge for finding effective ramp-up policies. The specific characteristics of each system

Manuscript received May 30, 2013; revised November 20, 2013; accepted November 26, 2013. Date of publication January 9, 2014; date of current version August 14, 2014. This work was supported by the European Commission as a part of the CP-FP 246083-2 IDEAS and CP-FP 229208-2 FRAME Project. This paper was recommended by Associate Editor R. J. Marks II.

The authors are with the Manufacturing Division, University of Nottingham, University Park, Nottingham NG7 2RD, U.K. (e-mail: emxsd@nottingham.ac.uk; p.ferreira@nottingham.ac.uk; niels.lohse@nottingham.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2013.2294155

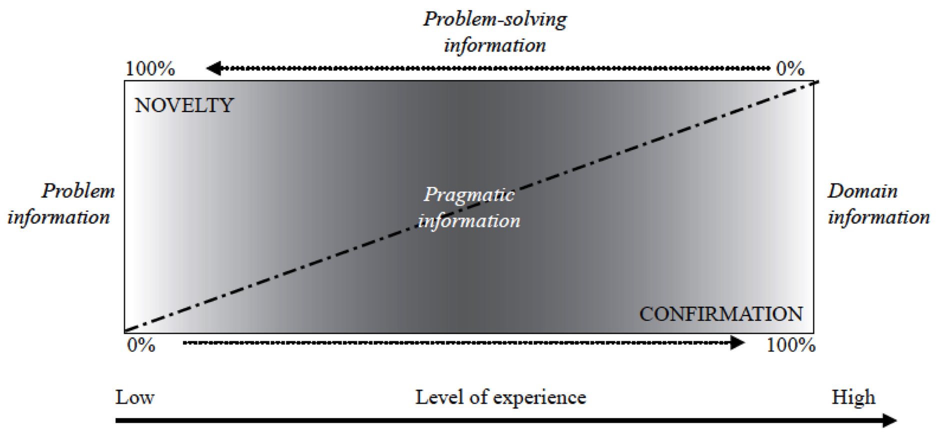

Fig. 1. Level of experience against pragmatic information during ramp-up [1].

will generate unique disturbances, which make the acquired knowledge case specific and not always possible to reuse.

Manufacturing systems are generally composed of stations (subsystems) or machines, which are ramped-up independently from each other. In addition, their ramp-up is sequential and iterative. These characteristics allow the ramp-up process to be formalized as a Markov decision process (MDP) where operator actions cause observable state transitions, which affect the ramp-up status of a station. Capturing the dependencies between actions and reactions allow machine learning methods to be used to find good policies for the ramp-up of observed stations.

Reinforcement learning (RL) is identified as a promising approach that performs well for the specific characteristics of the decision making process during ramp-up. This paper reports an MDP model for a RL approach to reduce ramp-up time. A literature review is carried out in the next section, which supports the problem definition in Section III. The RL approach for ramp-up is presented in Section IV, followed by the experimental description and the results in Section V. The paper is concluded along with proposed future work in Section VI.

II. Ramp-up Problem

Ramp-up is a phase that can be potentially improved to reduce the time to market for new products. A significant amount of work relates the level of knowledge during ramp-up to the time the process requires, which affects a companies' revenues [1], [4], [6]. The significance of knowledge and the problem of knowledge loss is also put forward as a bottleneck in contemporary manufacturing systems [1], [2].

A. Role of Knowledge and Learning in Ramp-Up

The process of learning can generate knowledge, which can later on support ramp-up. The role of learning, knowledge acquisition, and knowledge sharing within an enterprise during ramp-up has been studied and identified as the main cause for prolonged ramp-up times in several works [1], [2], [4], [6]–[8]. Terwiesch and Xu [4] emphasized on the role of process understanding and suggested the postponement of process changes, since they lead to disruptions and systematic learning cannot be achieved. The study is based on a copy-exactly (CE) ramp-up approach [9], which is recommended when the process understanding level is not high and the lifecycle is short with a quick growth. The need to carry out systematic experiments during ramp-up has also been highlighted in [2] and [10]. Terwiesch and Bohn [2] emphasized the need for learning during ramp-up and support the idea of reducing the output in early production stages regardless of the increased demand, and focused on the learning process. They stated that it is nonetheless still the moment to further reduce output to run engineering trials and work on yield and speed improvement [2]. Haller *et al.* [10] indicated the need for knowledge improvement while the yield is low. The way to achieve that is through designed experiments and the accumulated experience through yield increase. In a different approach, Fjallstrom *et al* [1]. conducted a study on the role of information during ramp-up. They classified the types of information into domain, problem, and problem solving. The sources of information are defined as other people, documentation, visits, and experience. The aim of the source classification is to find their effect on handling the critical events during the process. One of the interesting outcomes is the use of information by experienced and less experienced personnel. The former were found to prefer domain knowledge information, opposite to the latter, which preferred problem information. The authors concluded by presenting a model showing the analogy between the pragmatic information regarding the ramp-up problem and the level of experience of the personnel, shown in Fig. 1.

The role of the personnel's knowledge is characterized as highly valuable across the literature [1]. Along with the critical role of experienced technicians as a source of information, show a need for knowledge and experience transfer between the personnel involved in the ramp-up process.

B. Learning During Ramp-Up

The need for knowledge and its correlation to the ramp-up progress is widely accepted. However, only few papers study or propose ways of learning, which would lead to knowledge regarding the process. Instead, they focus on the correlation of the two. It is frequently stated that ramp-up is a good opportunity for learning or that learning is a process complementary to ramp-up [2].

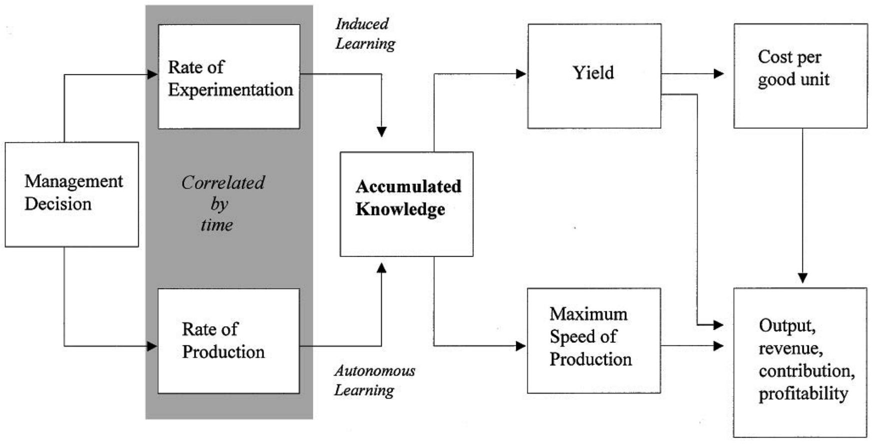

Fig. 2. Causes of learning and improvement [2].

Terwiesch and Bohn [2] in their work focused on the effect of learning for process improvement during production ramp-up and present different learning approaches. The authors proposed a model that establishes the cost for experiment design and learning highlighting its importance. Additionally, they presented two of the main drawbacks of learning by experimentation, namely, the capacity consumption and the effect of the deviation caused on the perception of optimal control. Fig. 2 shows the flow of causality for learning during ramp-up.

An article, which explores the effect of learning during ramp-up in a growing market demand environment with unsteady production rates, is presented in [11]. It claims that if the learning rate cannot be improved, a blind additional assignment of worker to support ramp-up will show a direct correlation between learning rate and result. It is important to highlight that one of the proposed additions to current models is a factor for forgetting the learned knowledge and the use of different workers. This stresses the lack of knowledge maintenance and transferability across ramp-up cases and workers, respectively. Mannar and Ceglarek [7] pointed out that generating a model in advance of the actual process is difficult due to the lack of data. Therefore, the authors proposed a methodology that learns a model based on training data. The methodology uses the rough sets technique to achieve self-learning and uses that to detect new faults during the ramp-up adjustments. The outcome is an if then set of rules, predicting hidden patterns of failure.

The literature reveals the significant role of factors like information, experience, learning, and knowledge during ramp-up. These factors have been included in ramp-up models, which aim to measure their effect and impact [2], [4], [12]. It is also a common view that knowledge is not static information but comes from process change and experimentation [6]. Vits *et al.* [13] established that the investment in process changes is unavoidable, otherwise performance improvement would never occur.

C. Learning Through Process Change

The effects of process and equipment change are a significant part of manufacturing process models. Ramp-up is by definition a change process, which is related to such models. Winkler *et al.* [14] provide a quantifiable relation for change by establishing cost and knowledge level reduction as a critical factor. Other approaches capture the level of learning through performance indicators and its effect is modeled in the overall product cost [10], [12]. Especially for the ramp-up phase where the knowledge level is very low, the learning rate has been highly evaluated and shows significant performance improvement. The process improvement tends to become synonymous with the effectiveness of the learning process.

Three ramp-up models based on process change have gained a lot of attention in the literature, proposed in [4], [6], and [13]. All three are developed to support decision making within a process change environment, based on optimization criteria. In the first model, the investment in learning is justified by introducing a cost for learning and process change during ramp-up. The model's output is the financial outcome in a ramp-up process by balancing four factors, the product's revenue, the costs of the process change, the knowledge creation rate, and learning [4]. Both [6] and [13] proposed a profit-based model, considering a balance between process change and learning. In overall, the literature shows the importance of increasing knowledge and how such efforts affect the decision making process. All of the models in the literature, although they highlight the significance of knowledge, appear not to address the question of how to formalize or transfer it. Machine learning (ML) techniques can provide a great tool in this direction [15].

D. Learning Approaches for Ramp-Up

In manufacturing, a lot of ML and data mining applications have been reported in the past decades in several review papers [16]–[20]. All of them advocate that ML and data mining techniques are required for knowledge gathering. Pham and Afify [18] for instance concluded that ML techniques will help automating knowledge gathering, which is essential for reducing the time of manufacturing operations.

ML methods and semantic models have been used in manufacturing to formalize knowledge [17], [21]. The literature reports on ML applications in manufacturing, which are mainly focusing on controlling and optimizing specific processes

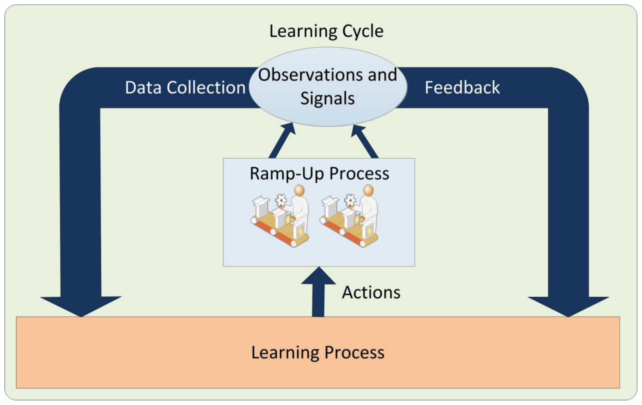

Fig. 3. Learning during ramp-up, an overview.

rather than a whole manufacturing phase [17]. Ramp-up needs to be approached as a phase independent of specific processes and enables machine learning that is not process specific. This can enable the generation of knowledge that can be used repeatedly across different ramp-up cases. Additionally, ramp-up is a process, which incorporates characteristics, such as uncertainty of information and lack of data. These characteristics provide a significant challenge for ML applications.

RL is a machine learning approach, which considers a sequential environment response and emulates the process of human learning [22]. Ramp-up requires practical characteristics that can take advantage of RL properties, such as episodic learning and efficient data utilization [22]. Additionally, it is enabled through MDP models that can support learning from humans [23]. Although the concept of RL is not widely applied in manufacturing, it is still not entirely new. RL techniques have been used in problems, such as scheduling [24], [25], production goal regulation [26], and the concept of biological manufacturing systems (BMS) [27]. Although, these approaches are not directly applicable to the ramp-up problem, they provide insight on how to model manufacturing problems for ML. RL algorithms, such as batch learning (BL), have been demonstrated to provide good results in limited data scenarios [28]. This fact indicates that RL approaches are very promising for ramp-up applications. Kalyanakrishnan and Stone [28] presented BL algorithms which performed better than others in complex domains with limited amount of data. Algorithms like experience replay achieve maximum data utilization and convergence. They can support learning with limited data and learn quickly be replaying the acquired data [29]

Only a few works have focused on promoting learning during production ramp-up. Strictly classified ML application have not been reported until recently [5], [30]. In [31], the overall assembly lines are semantically mapped to define attributes and characteristics during ramp-up. In that approach, the ontology of the ramp-up process is defined, though without including the correlations between the main concepts, namely, the product, the process, and the equipment. Significant work has been reported in the same direction by the collaborative project EU FRAME [32]. In [32], the requirement of learning during ramp-up has been highlighted. A combination of semantics and ML algorithms is proposed and a nearest neighbor's-based algorithm is developed with promising results. However, the required amount of data is a drawback of the algorithm. RL has been applied for ramp-up in [5] where authors reported results on a model free Monte Carlo algorithm applied for ramp-up time reduction under a Copy-Exactly casestudy. Similar approach has been followed in [30] where Q-learning algorithm shows time reduction with fast algorithm convergence. In this paper, the framework proposed in [30] is further explored. The MDP model for RL is formalized and extensive algorithmic analysis is carried out. The model is also tested against different types of algorithms.

III. Learning Process During Ramp-up

Ramp-up can be defined as a decision making process where the decision maker observes the system behavior and apply corrective actions to reach the desired target behavior. Human decision makers are very well suited for this task as they are very good at making decisions based on incomplete information and continuously learn from experience. There are, however, a number of challenges with human driven ramp-up of systems. Humans tend to be overconfident when applying changes, which can lead to bad results [5]. Additionally, knowledge transferring and sharing between individuals is a time consuming process with no guarantee of results. The generalization of knowledge in terms of similar equipment is also quite challenging, especially when the manufacturing system is composed of several different processes. The problem lies in the abstraction of not only the equipment but also the processes the equipment performs.

The use of learning algorithms is a way of overcoming such issues by supporting the human decision process. In order to provide this support, it is crucial to be able to observe the process, extract and capture experience in a structured way, and link this to a formal model of the ramp-up process and identify the most effective learning methods. To understand the process, it is important to be able to capture the adjustment and tuning actions of the operator and their effect on the system. Figs. 3 and 4 present an overview of the learning cycle and the different ramp-up cases, which are defined in more detail in [30]. The acquired data from monitoring the process, namely, sensor data or human observations, will serve as the basis for the analysis of the ramp-up state. In practice, an operator makes changes (actions) on the system and expects a reaction, which indicates the system's status. This needs to be stored and processed to analyze and learn the system's behavior. The way ramp-up is approached can support different ways of data processing and learning algorithms.

Current practices for system ramp-up can be summarized as two main strategies. These are the single time ramp-up of one-of-a-kind systems and the repetitive ramp-up processes of similar systems. In the first case, ramp-up happens once with generally little to no significant prior knowledge of the system behavior (Fig. 4, Case 1). This means that learning would have to occur online (during the actual ramp-up process) with limited data and heavy restrictions on the exploration

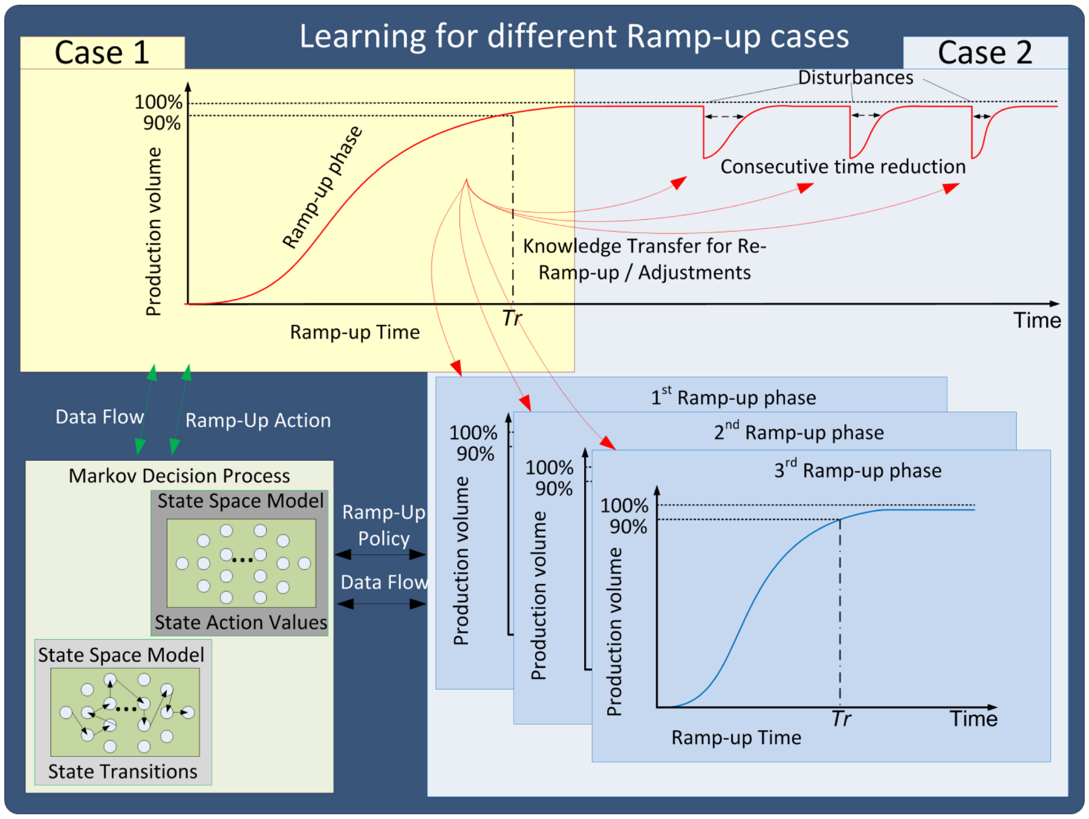

Fig. 4. MDP model for different ramp-up cases.

of the system behavior. In the second case, ramp-up of the system has been achieved before, thus the aim is to replicate the good practices from a previous ramp-up process (Fig. 4, Case 2). This means that existing data can be feed to the learning process in batches.

A formal model to capture ramp-up experience is required to enable the learning of the most effective sequence of actions to achieve the required system performance. An MDP model, which correlates the state of a system with an adjustment action and the resulting change is proposed to formalize the ramp-up process. This model can be used as the basis for different learning approaches, which apply model-based (indirect) and model free (direct) algorithms to generate system transitions and state values, respectively. There are multiple learning approaches that could be applied in the proposed learning framework. Nevertheless RL has been identified as one of the most promising due to its ability to solve MDPs and the feedback based learning.

IV. RL Approach

RL is a learning method based on a feedback (reward) coming from the environment. The idea is that the learner (controller) interacts with the system by applying changes (actions) while their effect is monitored through the reaction of the environment. Based on this, a policy is generated which tries to maximize the received reward. The described sequence of actions during ramp-up from a RL perspective is summarized in Fig. 5, where the decision making process is presented as a model identification adaptive control (MIAC).

In practice, learning would take place as a complementary process to ramp-up, while providing support to its decision making process. In the first loop, in Fig. 5, the ramp-up process is under the operator's control without any support. The output of the system (performance) is fed back to the controller, which considers the overall aim and acts accordingly. The process goes on until the system reaches in the desirable behavior. In the second loop in Fig. 5, the RL process monitors the decision making process and collects online data. Additionally, data batches can be fed in the system identification (see Fig. 5). These can be used to learn a model and the model's state values. Based on all the information (from the system and batch data), a policy is generated (adjustment mechanism) that targets reward maximization. The policy can then be used to make recommendations to the operator who chooses the next action. The recommendation is based on the current state of the system and a forecast mechanism, which predicts the expected result of an action. The quality of the policy depends on the amount and quality of the experience data. In cases where no information and no previous experience exist, a policy cannot be suggested.

The sequential structure followed in RL matches the ramp-up process and can support learning based on multicriteria, which is also the approach followed by operators. An operator usually considers several factors before a decision is taken such as input/output signals, unexpected observations and behavior of the system, product quality results, stability of

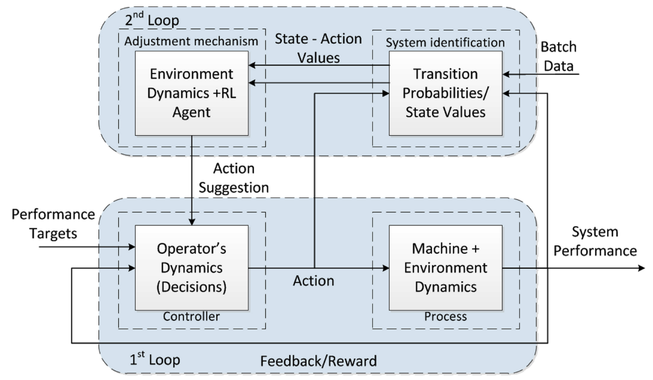

Fig. 5. Reinforcement learning during ramp-up.

the process, etc. One of the advantages of RL is that to export a policy, an explicit model is not necessarily required since knowledge can be based on instant response. This is a significant characteristic that helps to cope with the limited amount of data during ramp-up. The MDP model is formally defined in the next sections followed by the proposed learning approach.

A. MDP Model

In order to define a formal model of the ramp-up process that can be used to analyze and learn from previous experiences, it is important to consider all the factors, which are being considered by the human decision makers, their chosen action and the effect of the action on the system. The effect an action has on the system behavior is subject to the complex interdependencies between its constituent equipment components, their functional roles, and their individual inaccuracies. This often makes the result of an action stochastic and difficult to predict. Generally, ramp-up can be considered to be a sequential process driven by environment response with a limited amount of data. MDP modeling is a powerful tool to formalize stochastic processes and combined with reinforcement learning has the potential to deal with the aforementioned ramp-up characteristics [33].

Ramp-up can be said to be a Markov process, since in practice the information used to reach a decision about the next action, usually only regard the current state of the system. This is defined as the Markov property (1), and is a notable assumption for the modeling of the ramp-up process. Even if a process within ramp-up does not strictly incorporate the Markov property, a model can be nevertheless designed as such by choosing the right state variables. MDP further guarantees convergence to an optimal solution for a range of algorithms. An MDP for ramp-up consists of the ramp-up process states S, a list of actions A, a reward R and the process horizon T

$$\begin{aligned} \mathrm{P}\left\{\mathrm{X_n}=\mathrm{j_n}|\mathrm{X_{n-1}}=\mathrm{j_{n-1}},\mathrm{X_{n-2}}=\mathrm{j_{n-2}},\ldots,\mathrm{X_0}=\mathrm{j_0}\right\} \\ =\mathrm{P}\left\{\mathrm{X_n}=\mathrm{j_n}|\mathrm{X_{n-1}}=\mathrm{j_{n-1}}\right\} \end{aligned} \tag{1}$$

where $\{\mathrm{X_n}, \mathrm{n}=0,1,2\ldots\}$ is a sequence of random variables and $\mathrm{j_k}\in \mathrm{S}$ is a state.

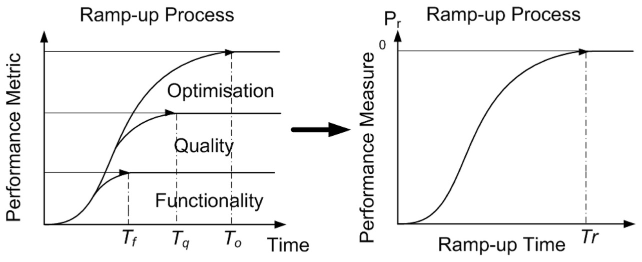

Fig. 6. Ramp-up break down.

B. Ramp-up Process State

The state variables are those that describe the system's condition at every time epoch. The state variables are chosen with the aim to support the learning process, while including all the essential information of the process. An exhaustive state description would generate a large state space, which would require large amount of data for exploration. In fact, during ramp-up a large state space would probably never be explored and its policies would be case specific. Therefore, it is proposed that the state parameters are chosen to be similar to those a human operator would use to define the system's condition. These are descriptive variables that can provide information in a more compact form. Following the analysis of ramp-up in [34], a state $\mathrm{s_t}=\left\{\mathrm{p_1^t},\mathrm{p_2^t}\cdots,\mathrm{p_m^t}|\mathrm{P}\right\}$ in time epoch $\mathrm{t}\in\mathrm{T}$ is defined as a set of state variables $\mathrm{p_m}\in\mathrm{P}$. A framework has been proposed in [34] to define performance during ramp-up, which can serve as the bases for the state definition. Ramp-up performance is broken down to three main phases, functionality, quality, and optimization (see Fig. 6), which can provide the state parameters.

A state during ramp-up is defined as the combination of functionality, quality, and performance optimization parameters. Based on this definition a state $\mathrm{s_i}$ is formalized in (2) as the union of the parameter sets of the three types of parameters under this state

$$s_i = F_i \cup Q_i \cup T_i. \tag{2}$$

The parameter sets of functionality, quality, and optimization are further broken down into system parameters. The functionality set is defined as the union $F = \{D \cup Ob \cup S_e\}$ of disturbance parameters $D=\left\{d_1,d_2\cdots d_j\right\}$, observations $Ob=\left\{ob_1,ob_2\cdots ob_j\right\}$ and emergency signals $S_e=\left\{s_{e_1},s_{e_2}\cdots s_{e_j}\right\}$. The quality parameter set is formalized as $Q=\left\{q_1,q_2\cdots q_j\right\}$. Finally, the performance optimization set $T_o=\left\{T_p\cup T_d\cup T_{oc}\right\}$ is the union of phase parameters $T_p=\left\{ph_1,ph_2\cdots ph_j\right\}$ operation duration, $T_d=\left\{du_1,du_2\cdots du_j\right\}$, and overall cycle time $T_c=\left\{t_1,t_2\cdots t_j\right\}$. The state parameters and their values are further defined in Table I.

The state parameters are monitored during a ramp-up process and are used to calculate a ramp-up state. The parameters are further used to generate the system's response through a reward formula presented later in this paper.

C. Actions

All the applied actions $a \in A$, $|A| < \infty$ across the state space of the MDP define the policy or decision rules. The actions during ramp-up cannot be generalized since the available actions are case and domain specific. However, they should comply with the following rules.

1) An action must incorporate one physical action at a time and have a transparent definition. More than one physical action can be considered as actions as long as they are defined as one.
2) The way an action is applied is not a part of the action definition but it is a matter of the operator who applies the action. This provides the means to capture the differences between the operators in policies. The generated policy will then be independent of how operators apply actions.
3) There has to be at least one action affecting every state parameter.

D. Policy

The policy is the outcome of the decision making process to support. Its definition can differ according to the type of the process behavior that needs to be captured (deterministic or probabilistic). Assuming that the actions are Markovian and not history dependent, the policy is defined as deterministic if a decision at time t is $d_s(s_t) \in A_s$ and, hence, the policy can be said to be Markovian deterministic Π^{MD}. Alternatively, if the decision at time t is probabilistic $q_{d_t(s_t)}(.) \in \wp(A_{s_t})$, then, the policy is said Markovian random Π^{MR}, where $q_{d_t(s_t)}(.)$ is the probability distribution on a set of decisions d_t. The definition of the policy affects the accumulation of returns as well as the transition probabilities of the decision process. In a deterministic case, the return is the result of a decision $d_s(s_t)$ under a state s_t denoted as $r_t(s, d_t(s))$ and a transition probability $p_t(j|s, d_t(s))$. In a probabilistic design, the return considers all the possible state transitions based on the transition probability $p_t(j|s, d_t(s)) = \sum_{a \in A_s} p_t(j|s, a)\, q_{d_t(s)}(a)$ and is denoted as $r_t(s, d_t(s)) = \sum_{a \in A_s} r_t(s, a)\, q_{d_t(s)}(a)$. Although the nature of the problem is rather probabilistic, a deterministic policy should cope better with limited data. The policy is further defined as stationary, since ramp-up is a process independent of time and only dependent on the system's state.

E. Learning Episodes, Horizon, and Reward

The horizon of the decision is the number of returns considered from future states to take a decision. Learning algorithms can be treated as either finite or infinite horizon problem. A Finite horizon process is the one that looks ahead only for a certain number of steps and aims for an optimal action according to this horizon. An infinite process considers infinite future returns normally until the end of the process to define a reward.

Another distinction is in regard to the length of the learning problem. It distinguishes between episodic and continuous problems. A problem is episodic when there is a clear end state and continuous when there is no end state but a constant seeking and updating of the policy. In the second case, the rewards received are infinite and a discounting factor $(0 \leq \gamma \leq 1)$ needs to be applied for each received reward to assure a bounded return. In the episodic case, learning often happens between episodes and the return is based on the accumulated rewards received throughout an episode. The discounting factor, although it may affect the decision maker's choice, it does not have any difference on the theoretical results of the episodic case [33]. The reward can even be considered for only one step ahead. A horizon longer than just the instant effect reveals the longterm effect of an action. In this case, delayed returns (r_t) are considered, as well as the effect of combined actions. The generic structure of a reward at time t is defined as

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}. \tag{3}$$

Ramp-up can be seen as a continuous or episodic problem. Referring to Fig. 3 and the two learning cases during ramp-up, it is clear that in the first case, ramp-up needs to be treated as a continuous learning task. In the second case, sequential ramp-up phases are imported as batch data, which will improve the learning. The model proposed in this paper is able to cope in principle with both cases.

The reward used for ramp-up is based on a performance measure proposed in [34], which is aligned with the state definition. Three measures have been defined and can be weight differently based on their significance. Equations (4)–(7) present the model of performance during ramp-up

$$f_f(j) = -\sum_{j=1}^{n} k_j D_j \tag{4}$$

$$f_q(j) = -\sum_{j=1}^{n} \lambda_j Q_j \tag{5}$$

$$f_q(j) = -\sum_{j=1}^{n} \lambda_j Q_j \tag{6}$$

$$R_{PM} = \frac{1}{s_t} w \cdot f_{RU} \tag{7}$$

where $\mathrm{w} = [\mathrm{w_f}, \mathrm{w_q}, \mathrm{w_o}]$ is the weight vector and $\mathrm{f_{RU}} = [\mathrm{f_f}, \mathrm{f_q}, \mathrm{f_o}]$ is the performance metrics vector. Parameters $\mathrm{k_j}, \lambda_\mathrm{j}, \beta_\mathrm{j}$ are weights and $\mathrm{D_j}, \mathrm{Q_j}, T_{o_j}$ are as defined in Table I. Although time reduction is the driving aspect of ramp-up, the system's performance can provide a more detailed insight in the ramp-up process. Using a performance measure provides a feedback of the effect a specific action has on the system and whether it resulted in a positive trend to reach the desired final system behavior. In the proposed performance model, the step number can also be taken, into consideration if s_i becomes the step number.

F. Batch Learning Approach

Batch reinforcement learning (BRL) is a RL approach, which does not allow the learner to interact directly with the environment [22]. In BRL, there is no option for the learner to explore and the learning process is consequently independent of the applied policy. BRL algorithms aim to discover the

TABLE I
RAMP-UP STATE VARIABLES

State variables Definition	Formulation
The functionality set $F \in [0,1]$ denotes disturbances of the process and consists of disturbances $D_j \subset F$ identified by **operational** deviation, identified by time excess **Observations** $Ob_j \subset F$ are used to characterise	$d_j = \begin{cases} 1, & T_c > t_f T_{ct}, T_{ct} \text{ the target value} \\ 0, & \text{otherwise} \end{cases}$ $ob_j = \begin{cases} 1, & \text{disturbance observed} \\ 0, & \text{otherwise} \end{cases}$
disturbances that cannot be captured by a sensor but by an operator and finally equipment **emergency signals** $S_{e_j} \subset F$. j is the disturbance parameter number, T_c is the cycle time and T_{ct} is its target value.	$s_{e_j} = \begin{cases} 1, & \text{triggered signal} \\ 0, & \text{otherwise} \end{cases}$
$Q \in [0,1]$, is the set of quality parameters for a ramp-up state space. Every quality parameter, $q_j \subset Q$ of the produced product, defines if every quality objective j is achieved.	$q_j = \begin{cases} 1, & \text{objective achieved} \\ 0, & \text{otherwise} \end{cases}$
The performance optimisation set $T_o \in [0,1]$, shows the optimisation status of the ramp-up process It consists of the **operation phase parameters** $ph_j \subset T_o$, the **operation duration parameters** $du_j \subset T_o$ and the **overall cycle times** $t_j \subset T_o$.	$ph = \begin{cases} 0, & \text{operation duration below target} \\ 1, & \text{operation duration on target} \\ 2, & \text{operation duration above target} \end{cases}$ $du = \begin{cases} 0, & \text{operation phase below target} \\ 1, & \text{operation phase on target} \\ 2, & \text{operation phase above target} \end{cases}$ $t = \begin{cases} 0, & \text{cycle time below target} \\ 1, & \text{cycle time on target} \\ 2, & \text{cycle time above target} \end{cases}$

best policy within the set of observed policies rather than seeking the optimal policy through interaction. Hence, two questions arise regarding the learned policy; its convergence and its quality. If the available data does not include a good or optimal policy, then it is not possible to generate one either.

The problem is how to find the best policy within the provided data. Batch reinforcement learning algorithms present practical advantages for the repetitive ramp-up case since they can guarantee a policy convergence. The fact that policies are applied by operators with some type of knowledge enhances the exploration value. In addition, the experience replay algorithm increases efficiency for small data batches while guaranteeing convergence under certain criteria [22]. Additionally, the use of a temporal difference (TD) based algorithm provides stable behavior

$$V'(s) = V(s) + a\left[r + \gamma V(s') - V(s)\right] \tag{8}$$

$V'(s)$ is the updated value of a state s based on the previous state value $V(s)$, and the temporal difference of the previous value of state s and the following state s′, added to the immediate return r. Parameters a and γ are the learning rate and the discount factor, respectively.

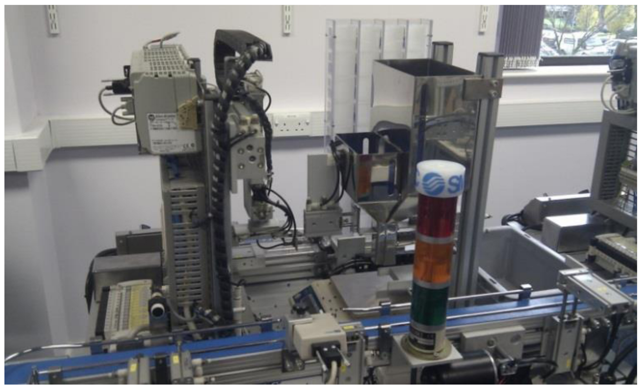
Fig. 7. SMC HAS production station.

G. Batch Q-Learning Algorithm

For discrete small state spaces, traditional RL algorithms, such as TD(0), Monte Carlo (MC), and Q-learning can be aligned to match the batch learning framework. The main difference between Q-learning and MC is that the update rule of the former is based on bootstrapping while the latter does not. Bootstrapping happens when the update of a state-value is based on the value difference compared to the previous state value. Hence, the result reflects the trend of a policy

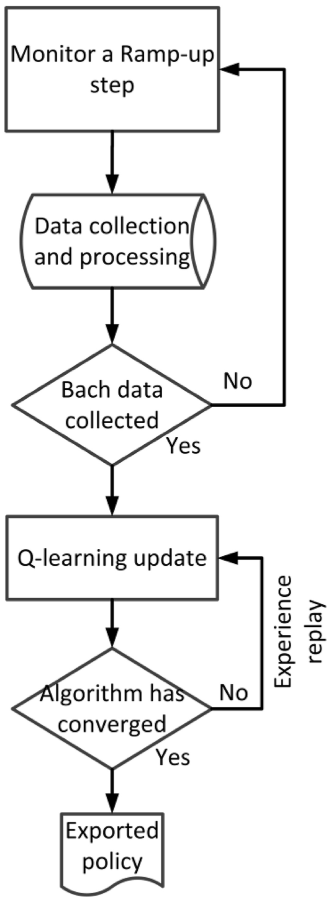

Fig. 8. Experimental process flow chart.

TABLE II
STATE PARAMETERS AND VALUES

State Variable (S.P.)	S.P. Value
Pick and Place 1Duration	UF/S
Filling Operation Duration	UF/S
Overall Cycle Time	UF/S
Pick and Place 2Duration	UF/S
Overall Cycle Time	UF/S
Weight	Low/Ok
Observations	
• No Movement	Yes/No
• Spilling	Yes/No
• Alarm	Yes/No
• None	Yes/No

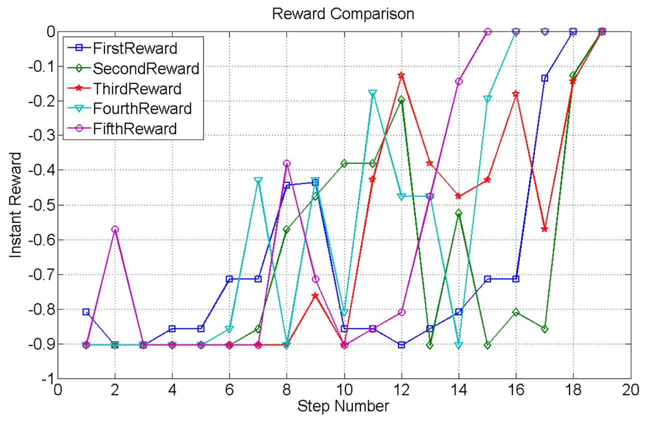

Fig. 9. Reward variation during ramp-up episodes.

in addition to the final result, whereas, the nonbootstrapping methods only consider state-action values independently from each other. MC does not update on step-by-step bases but only considers the averages reward at the end of an episode. Bootstrapping is expected to provide a better data use. The Q-learning algorithms is based on the TD update rule but instead of estimating the state value $V'(s)$, it estimates the state action value $Q'(s,a)$ based on the temporal difference of the current state action value and the maximum of the following state:

$$Q'(s,a) = Q(s,a) + a\left[r + \gamma \max_{a'} Q\left(s',a'\right) - Q(s,a)\right]. \quad (9)$$

For the case of ramp-up, R_{PM} becomes the return and $R_{PM}(s,a)$ is the Q value of a state s under an action a. More advanced algorithms have also been proposed; however, they mainly aim to operate on large continuous spaces using function approximation methods [22]. In this paper, a Q-learning algorithm is applied within the learning framework, which uses the proposed MDP model. The algorithm estimates all the Q values of the state space based on a number of data sets fed offline. The data sets are in the form of episodes and for every episode the Q values are updated. In the end of every episode, a policy is extracted and updated by finding those actions that generated the biggest Q values. The same process goes until the algorithm has converged and the policy does not update any more. This is called experience reuse.

V. Experimental Setup and Results

In this section, the proposed model is used to learn the best policy for a fully automatic production station. The performance of the Q-learning algorithm will be assessed and the resulting policy will be applied to test its impact on the actual ramp-up process. The equipment used is a production station, which is part of a ten station SMC highly automated production system (SMC HAS-200). The production station (see Fig. 7) is comprised of three independent processes, two pick and place processes and one filling process. The station operates through pneumatic actuators and is controlled by a single PLC. The purpose of the station is to dispense and fill a container with a predefined amount of material and place it on a conveyor to transport it to the next station.

A simplified model was defined for the aforementioned process to experimentally validate the proposed MDP and analyze the Q-leaning algorithm in a batch mode. Seven functionality (three operation durations, one cycle time and four

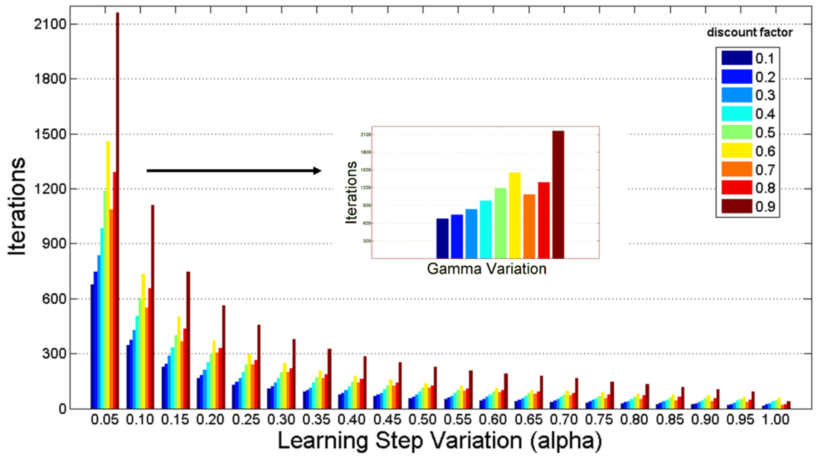

Fig. 10. Learning iterations against the variation of the learning step and the discount factor.

disturbances), one quality, and four optimization parameters were used for the definition of the MDP (see Table II). The MDP is defined by the resulting 128 states.

In order to emulate a repetitive process, five ramp-up processes were carried out independently from each other. All experiments were conducted under the same environmental conditions. The only allowed variation during the experimental set-ups was the choice of actions for carrying out the ramp-up process. The station was ramp-up from the same suboptimal starting conditions by five different operations. None of the operators had any prior knowledge of the station and had no contact with the other people who had already participated in the experiments. It is important to point out that the operators were not expert system integrators but had a background in manufacturing. Moreover, there was no time restriction in the decision making process to allow each operator to carefully consider the best action for the current condition of the station. The initial state of the station was set to similar conditions in all cases and a number of disturbances were induced to emulate the ramp-up process. Three different disturbances (observations listed in Table II) were identified during separate experimentation and used during the ramp-up emulation. For every ramp-up session, the system was initialized to start from a randomly selected disturbance and the rest naturally occur during the ramp-up process as a response of the station being ramped up. The initial state of the system was randomly generated to reflect the uncertainty of the system behavior after its initial build or after a change. The more disturbances occur during ramp-up, the wider is the exploration of the system and therefore, better policies can be learned. The operators were able to apply the following seven actions:

1) increase station pressure;
2) reduce station pressure;
3) barcode reader alignment;
4) increase weight acceptance limits;
5) reduce weight acceptance limits;
6) reset station;
7) none.

The collected data was processed offline in a batch mode, which reflects the second loop of the ramp-up process in Fig. 5. The first loop was also implemented by processing the data offline and applying the exported policy to a new ramp-up case. More specifically, Fig. 8 shows a flow chart of the learning cycle. The ramp-up process is first monitored and data is collected as experience sets, which relate the current state of the system (2) with an action and the state after the change. The experience sets are collected in a database to compute the reward (7) and for further analysis of the overall ramp-up process. Once a complete batch of ramp-up experiences has been collected, the batch Q-learning algorithm is applied (9) until convergence has been achieved (Q-value change ~ 0). Then, a policy is extracted than can be used to recommend actions for known states when a new ramp-up is being carried out.

A. *Results*

The conducted five ramp-up sessions presented very different behavior by accumulating different rewards and following different paths to achieve the system ramp-up. These paths are presented in Fig. 9.

The operators were allowed to explore freely instead of using a predefined exploration policy. Therefore, Fig. 9 also reflects their learning curve in terms of the accumulated reward. It shows the differences on their decisions and reveals human learning under ramp-up. Those datasets are perceived as very valuable since they provide a naturally followed exploration. This is believed to compensate for the requirement of RL algorithms for constant exploration in order to converge. The datasets from the five ramp-up episodes are used to run the Q-learning algorithm for several variations. In the next part, an algorithmic analysis is carried out to demonstrate the validity of the learning framework in the context of the batch learning approach. Furthermore, the performance of the algorithm for the characteristics of ramp-up will be analyzed.

B. *Q-Learning Algorithmic Analysis*

In this part, the results of a Q-batch learning algorithm are presented applied on the proposed model. Fig. 10 shows the variation of iterations for the learning step a and the discount factor γ after running Q-learning in a batch mode. The experience set is being reused until the algorithm is considered to have converged. The algorithm's convergence

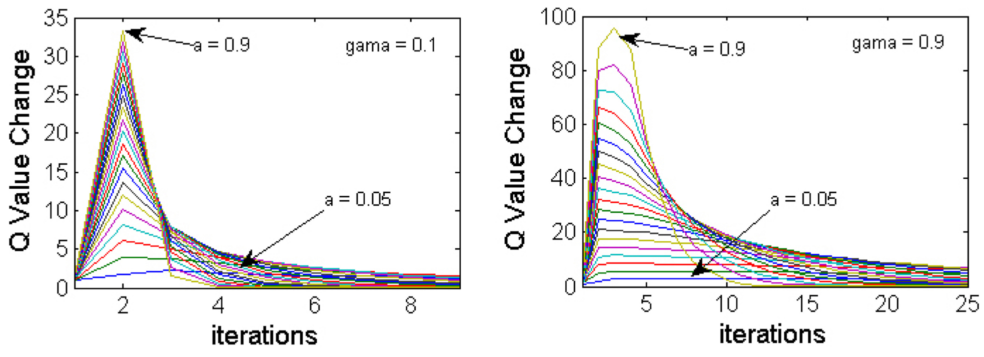

Fig. 11. Q-value change for different learning step and discount factor.

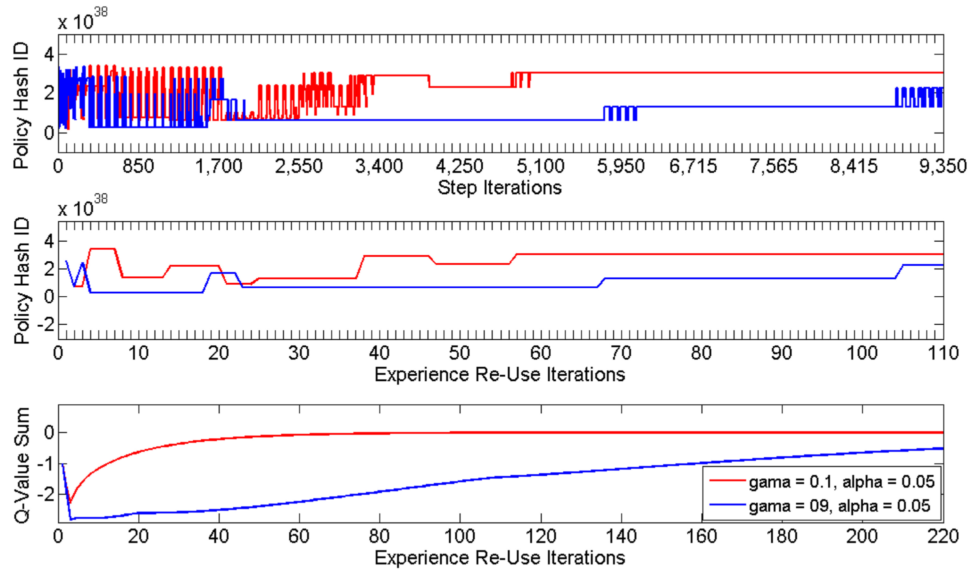

Fig. 12. Policy comparison and variation through iterations.

is checked through the absolute value of the relative Q-value change. The algorithm is run for fairly small a and γ values up to the extremes of $a = 1$ and $\gamma = 0.9$. As it is shown in Fig. 10, while the learning rate increases the number of iterations also increase for all the different discount factor values. This trend only changes for the values of $a = 0.7$ and 0.8. Although larger a values seems to be a much better option, a large learning rate value can cause instabilities or might not even converge to an optimum policy. For the boundary value of $a = 1$, the number of iterations explodes and the output becomes unstable since the algorithms never converges and keeps oscillating around the same values.

Furthermore, Fig. 11 shows the variation of the Q-value matrix throughout the iterations and the different settings for the extreme cases of $\gamma = 0.1$ and $\gamma = 0.9$. For all the intermediate values, the trend was the same. The algorithm generates a peak in the beginning when the changes to the Q-Value are big and then gradually slows down. Interesting is the fact that although the required number of iterations for convergence based on Fig. 10 is very high, Fig. 11 shows that only small changes happen after a certain point. The algorithm seems to generate a stable output relatively fast, in comparison to the overall amount of required iterations.

Regarding the effect of the different γ values on the output quality, it can be said that for a non-zero γ the update rule takes one or more of the future rewards into consideration while for $\gamma = 0$ it only takes the previous Q-value into consideration. A zero gamma value brings results a lot faster since the prediction only considers current state returns and becomes short sighted. Such an approach though could be argued to be very similar to a MC approach (nonbootstrapping method), which is expected to give different results without getting the advantaged of the experience reuse. A high γ value and a fairly small a should allow the algorithm converge to a good policy when less data are available and a large a when more data exist.

C. Policy Comparison

Previous results show that Q-Value stabilizes after several iterations. However, the convergence of the policy is achieved a lot faster where the policies of the two extreme values ($\gamma = 0.1$ and $\gamma = 0.9$) are presented (see Fig. 12). Although the Q-Value stops updating after more than 600 and 2100 times for the two presented cases, the policy seems to stabilize after 57 and 105 iterations, respectively. Fig. 12 includes three diagrams which from top to bottom show the policy change

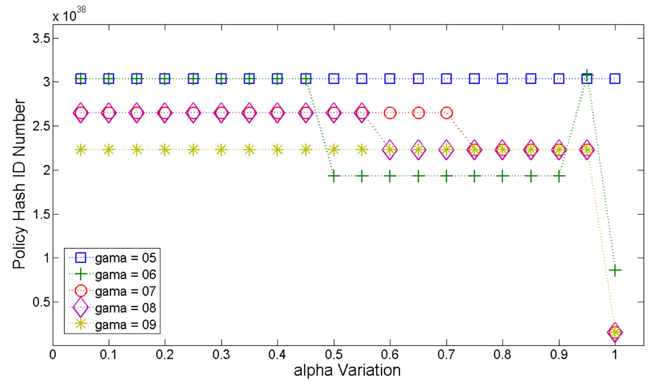

Fig. 13. Policy comparison.

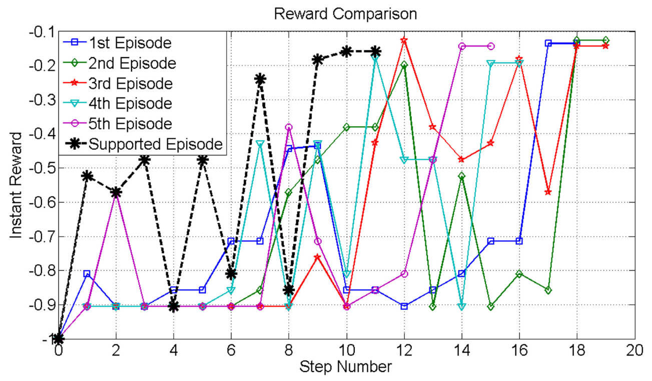

Fig. 14. Exported policy evaluation.

for every iteration step, the policy change for every experience replay iteration and finally, the change of the Q-Value matrix.

It has to be mentioned that every experience set is composed of 85 steps, which generates one experience iteration. For the policy comparison, a hash function (MD5) is used to generate a unique identity for every policy matrix. The diagrams concern the two extreme values $\gamma = 0.1$ and $\gamma = 0.9$, while $a = 0.05$. In the first two diagrams, the policy stops updating after more than 5100 and 6000 step iterations and after 57 and 105 experience iterations, for the two presented cases, respectively. Although the policy oscillates a lot on the step by step iteration it remains the same for every full experience iteration. That shows how the policy varies during an experience set until convergence. For different γ values, the algorithm exports a different policy. This is due to the increased emphasis on the next state-action value which becomes dominant for higher γ values to the exported policy and, hence, the different exported policies cannot be said to be wrong since the target has changed. It is a matter of the problem interpretation to define the horizon of prediction and the discount factor of future returns. In principle, since ramp-up is an episodic problem, the longer the prediction horizon, the better the policy target.

In order to further realize the effect of α and γ to the policy, Fig. 13 presents the hash ID of the exported policies for the variation of γ from 0.5 to 0.9 and the variation of α from 0.05 to 1. For all $\gamma < 0.5$, the policy follows the exact same trend as forγ =0.5. There are two conclusions that can be drawn from Fig. 13. For all $\gamma \leq 0.5$, the exported policy remains the same and stable. For higher values, the policy changes until γ becomes big enough that the following state-return becomes dominant. The variation of the output policy for the values between 0.5 and 0.9 shows the slow convergence to the domination of that return. Furthermore, different policies appear to be exported for larger α values, which is due to the instabilities these can cause. Large a values should be only being chosen when the luxury of time and computation is not an option. Furthermore, the fact the algorithm changes target for $\gamma \geq 0.6$ also justifies the drop on the number of iterations, taken for convergence, as shown in Fig. 10. As a derived rule, it can be said that large a creates faster convergence while it can cause instabilities overall and on small number of iterations, which can directly impact the results when RL is applied in an online bases. The γ value should be chosen to be either higher that 0.6 or a lot smaller depending if the following state value is considered or not, respectively.

D. Policy Evaluation

Finally, to evaluate the policy is applied in a new ramp-up case for the same station. The extracted policy for the values of γ =0.1 and a =0.01 was given to an operator and it was fully followed. Fig. 14 shows the accumulated reward of the policy against previous ramp-up cases. The aim of the application is not to find the best policy for the ramp-up of the station but find the best join policy between all recorded ramp-up experiences. A couple of interesting remarks are that in 8 out of 10 states, the proposed action was followed by the operator. In two of the cases, the states were unexplored. Finally, the policy shows overall time reduction by 4 to 8 steps, comparing to all of the initial ramp-up processes.

VI. Conclusion

In this paper, the first formal MDP model and RL approach have been reported for production ramp-up time reduction. A formal MDP model was presented for the capture and analysis of the ramp-up process. The MDP states, action characteristics, and feedback reward were defined based on the technical characteristics of a production station. An offline batch reinforcement learning algorithm has been developed, which utilizes the MDP to find the most optimal ramp-up policy from a small number of previous ramp-up experience cases.

The proposed approach has been experimentally evaluated using a production station to emulate ramp-up and applying a Q-learning algorithm. The focus was placed on Q-learning since bootstrapping algorithms were generally expected to perform better for the ramp-up case. Detailed results have been presented regarding the quality of the algorithm and its results. The Q-learning algorithm showed very good behavior when applied in batch mode. The approach demonstrated the ability of finding a stable policy under certain characteristics. The challenge to achieve good convergence with lack of data has been addressed by replaying the experience. Convergence was achieved after a short number of iterations highlighting the general suitability of the proposed reinforcement learning-based decision support approach. Further work is required to assess the efficiency of reinforcement learning and the effect of experience replay on ramp-up data. It would be interesting to explore how the policy changes throughout the experience replay and the operating characteristics curve of the algorithm.

To further improve convergence, future approaches should investigate initializing the algorithm's policy or Q-values. This could improve the effectiveness of the offline learning approach and also enable the development of online learning during ramp-up. Further work needs to be carried out to formally identify the advantages of a RL/Q-learning approach over to other methods.

Furthermore, an interesting observation was made when the horizon of the reward prediction was changed. Gradually increasing the horizon of consideration in the algorithm incorporates the idea of eligibility traces algorithms ($Q(\lambda)$-learning) with a horizon up to the end state. Such algorithms are expected to have good performance in online learning during a ramp-up episode. However, their specific performance will require further investigation. The development of an effective online learning approach for ramp-up is the crucial next step since it is often a luxury to have several consecutive ramp-up episodes. Hence, this approach should be explored further.

The experiments in this paper have been carried out on an automatic assembly station. The complexity of this ramp-up scenario is typical for the automation of individual assembly steps, which often have a high level of functional decoupling within larger assembly systems. It is expected that the proposed approach can be applied sequentially or in parallel to all the workstations in a system. In cases of functional dependencies among workstations, these will need additional analysis to be incorporated into the approach. Hence, further research should be carried out to investigate codependent parallel ramp-up processes for more complex multistation assembly systems.

The proposed method requires some initial data to export a good policy. It is better suited generally for either repetitive ramp-up of similar or the same stations or for the reramp-up of the same stations after a disturbance or change. Another direction to overcome such cases for future developments would be to investigate model-based approaches (indirect learning). It would be important to understand the amount of data required to learn a sufficiently complete model. In such a case, targeted exploration and predefined experimentation could reduce the required data and allow learning better policies. An indirect learning approach could provide a model that is easier to transfer and make it easier to incorporate existing expert knowledge during the initialization of the model, which could further reduce the number of required iterations and increase the quality of the policy.

REFERENCES

[1] S. Fjällström, K. Säfsten, U. Harlin, and J. Stahre, "Information enabling production ramp-up," *J. Manuf. Technol. Manag.*, vol. 20, no. 2, pp. 178–196, 2009.

[2] C. Terwiesch and R. E. Bohn, "Learning and process improvement during production ramp-up," *Int. J. Prod. Econ.*, vol. 70, no. 1, pp. 1–19, 2001.

[3] D. Ceglarek, W. Huang, S. Zhou, Y. Ding, R. Kumar, and Y. Zhou, "Time-based competition in multistage manufacturing: Stream-of-variation analysis (SOVA) methodology—Review," *Int. J. Flexible Manuf. Syst.*, vol. 16, no. 1, pp. 11–44, 2004.

[4] C. Terwiesch and Y. Xu, "The copy-exactly ramp-up strategy: Trading-off learning with process change," *IEEE Trans. Eng. Manag.*, vol. 51, no. 1, pp. 70–84, Feb. 2004.

[5] S. C. Doltsinis and N. Lohse, "A model-free reinforcement learning approach using Monte Carlo method for production ramp-up policy improvement: A copy exactly test case," in *Proc. 14th IFAC Symp. Inf. Control Problems Manuf.*, 2012, pp. 1628–1634.

[6] J. E. Carrillo and C. Gaimon, "Improving manufacturing performance through process change and knowledge creation," *Manag. Sci.*, vol. 46, no. 2, pp. 265–288, 2000.

[7] K. Mannar and D. Ceglarek, "Continuous failure diagnosis for assembly systems using rough set approach," *CIRP Ann. Manuf. Technol.*, vol. 53, no. 1, pp. 39–42, 2004.

[8] R. Oates, D. Scrimieri, and S. Ratchev, "Accelerated ramp-up of assembly systems through self-learning," in *Proc. IPAS*, vol. 371 AICT. Feb. 2012, pp. 175–182.

[9] C. J. McDonald, "Copy EXACTLY! A paradigm shift in technology transfer method," in *Proc. IEEE Adv. Semicond. Manuf. Conf. Workshop*, Sep. 1997, pp. 414–417.

[10] M. Haller, A. Peikert, and J. Thoma, "Cycle time management during production ramp-up," *Robot. Comput. Integr. Manuf.*, vol. 19, nos. 1–2, pp. 183–188, 2003.

[11] C. H. Glock, M. Y. Jaber, and S. Zolfaghari, "Production planning for a ramp-up process with learning in production and growth in demand," *Int. J. Prod. Res.*, vol. 50, no. 20, pp. 5707–5718, 2012.

[12] P. D. Ball, S. Roberts, A. Natalicchio, and C. Scorzafave, "Modelling production ramp-up of engineering products," *Proc. Inst. Mech. Eng. B, J. Eng. Manuf.*, vol. 225, no. 6, pp. 959–971, Jun. 2011.

[13] J. Vits, L. Gelders, and L. Pintelon, "Production process changes: A dynamic programming approach to manage effective capacity and experience," *Int. J. Prod. Econ.*, vol. 104, no. 2, pp. 473–481, 2006.

[14] H. Winkler, M. Heins, and P. Nyhuis, "A controlling system based on cause–effect relationships for the ramp-up of production systems," *Prod. Eng.*, vol. 1, no. 1, pp. 103–111, 2007.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4. New York, NY, USA: Springer, 2006.

[16] L. Monostori, "AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing," *Eng. Applicat. Artif. Intell.*, vol. 16, no. 4, pp. 277–291, 2003.

[17] L. Monostori, A. Markus, H. Van Brussel, and E. Westkämpfer, "Machine learning approaches to manufacturing," *CIRP Ann. Manuf. Technol.*, vol. 45, no. 2, pp. 675–681, 1996.

[18] D. Pham and A. Afify, "Machine-learning techniques and their applications in manufacturing," *Proc. Inst. Mech. Eng. B, J. E. Manuf.*, vol. 219, no. 5, pp. 395–412, 2005.

[19] J. Harding, M. Shahbaz, and A. Kusiak, "Data mining in manufacturing: A review," *J. Manuf. Sci. Eng.*, vol. 128, no. 5, p. 969, 2006.

[20] A. K. Choudhary, J. A. Harding, and M. K. Tiwari, "Data mining in manufacturing: A review based on the kind of knowledge," *J. Intell. Manuf.*, vol. 20, no. 5, pp. 501–521, 2009.

[21] N. Lohse, H. Hirani, and S. Ratchev, "Equipment ontology for modular reconfigurable assembly systems," *Int. J. Flexible Manuf. Syst.*, vol. 17, no. 4, pp. 301–314, 2005.

[22] M. Wiering and M. Van Otterlo, *Reinforcement Learning: State-Of-The-Art*. Berlin, Germany: Springer, 2012.

[23] Y. Lei, X. Yangsheng, and C. S. Chen, "Human action learning via hidden Markov model," *IEEE Trans. Syst., Man, Cybern. A, Syst, Humans*, vol. 27, no. 1, pp. 34–44, Jan. 1997.

[24] X. Li, J. Wang, and R. Sawhney, "Reinforcement learning for joint pricing, lead-time and scheduling decisions in make-to-order systems," *Eur. J. Oper. Res.*, vol. 221, no. 1, pp. 99–109, 2012.

[25] Y. C. Wang and J. M. Usher, "Application of reinforcement learning for agent-based production scheduling," *Eng. Applicat. Artif. Intell.*, vol. 18, no. 10, pp. 73–82, 2005.

[26] M. Shin, K. Ryu, and M. Jung, "Reinforcement learning approach to goal-regulation in a self-evolutionary manufacturing system," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 8736–8743, 2012.

[27] K. Ueda, I. Hatono, N. Fujii, and J. Vaario, "Reinforcement learning approaches to biological manufacturing system," *CIRP Ann. Manuf. Technol.*, vol. 49, no. 1, pp. 343–346, 2000.

[28] S. Kalyanakrishnan and P. Stone, "Batch reinforcement learning in a complex domain," in *Proc. AAMAS*, 2007, pp. 662–669.

[29] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 2, pp. 201–212, Mar. 2012.

[30] S. Doltsinis, P. Ferreira, and N. Lohse, "Reinforcement learning for production ramp-up: A Q-batch learning approach," in *Proc. ICMLA*, Dec. 2012, pp. 610–615.

[31] K. Konrad, M. Hoffmeister, M. Zapp, A. Verl, and J. Busse, "Enabling fast ramp-up of assembly lines through context-mapping of implicit operator knowledge and machine-derived data," in *Proc. IPAS*, vol. 371 AICT. 2012, pp. 163–174.

[32] E. Project. (2012). *Fast Ramp-up and Adaptive Manufacturing Environement (FRAME)* [Online]. Available: http://www.frame-eu.org

[33] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2nd ed. New York, NY, USA: Wiley, 2005.

[34] S. C. Doltsinis, S. Ratchev, and N. Lohse, "A framework for performance measurement during production ramp-up of assembly stations," *Eur. J. Oper. Res.*, vol. 229, pp. 85–94, Aug. 2013.

Stefanos Doltsinis received the B.Sc. degree from the Department of Automation, Technological Educational Institute of Piraeus, Piraeus, Greece, in 2007, the M.Sc. degree in control systems from the University of Sheffield, Sheffield, U.K., in 2008, and is currently pursuing the Ph.D. degree with the Department of Manufacturing and Operations Management, University Of Nottingham, Nottingham, U.K.

He is currently a Researcher with the University of Nottingham, where his research work includes production ramp-up, decision support systems, intelligent automation, agent-based control, and application of machine learning techniques in manufacturing.

Pedro Ferreira was born in 1981. He received the electrical, electronic, and computer science engineering degree from the New University of Lisbon, Lisbon, Portugal, in 2006, and the Ph.D. degree in manufacturing engineering and operations management from the University of Nottingham, Nottingham, U.K., in 2011.

He was a Research Associate with the Manufacturing Division, University of Nottingham, from 2010 to 2011. From 2011 to 2012, he was a Research Fellow with the Manufacturing Division. Since 2012, he has been a Senior Research Fellow with the Manufacturing Division. His current research interests include self-adaptable assembly systems, industrial agent systems, distributed self-organizing systems, sematic reasoning, transparent production, and decision making support for assembly systems.

Niels Lohse (M'13) received the Dipl.-Ing. degree in mechanical engineering from the University of Applied Science Hamburg, Hamburg, Germany, in 2000, the M.Sc. degree in technology management from the University of Portsmouth, Portsmouth, U.K., in 2001, and the Ph.D. degree in manufacturing engineering and operations management from the University of Nottingham, Nottingham, U.K., in 2006.

From 2006, he has been a Lecturer in advanced manufacturing technology with the University of Nottingham. His current research interests include intelligent automation including manufacturing system modeling, human–machine interaction, distributed control, diagnostics and design decision-support with a primary focus on modular and evolvable production systems, and applications of artificial intelligence techniques in manufacturing.

Dr. Lohse is a member of the IEEE Technical Committee on Industrial Agents and has received a number of awards for his work including the Outstanding Paper and Highly Commended Paper awards from the Emerald Literati Network for his publications in assembly automation.