A Symbiotic Human–Machine Learning Approach for Production Ramp-up

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

***Abstract*—Constantly shorter product lifecycles and the high number of product variants necessitate frequent production system reconfigurations and changeovers. Shortening ramp-up and changeover times is essential to achieve the agility required to respond to these challenges. This work investigates a symbiotic human–machine environment, which combines a formal framework for capturing structured ramp-up experiences from expert production engineers with a reinforcement learning method to formulate effective ramp-up policies. Such learned policies have been shown to reduce unnecessary iterations in human decision-making processes by suggesting the most appropriate actions for different ramp-up states. One of the key challenges for machine learning-based methods, particularly for episodic problems with complex state-spaces, such as ramp-up, is the exploration strategy that can maximize the information gain while minimizing the number of exploration steps required to find good policies. This paper proposes an exploration strategy for reinforcement learning, guided by a human expert. The proposed approach combines human intelligence with machine's capability for processing data quickly, accurately, and reliably. The efficiency of the proposed human exploration guided machine learning strategy is assessed by comparing it with three machine-based exploration strategies. To test and compare the four strategies, a ramp-up emulator was built, based on system experimentation and user experience. The results of the experiments show that human-guided exploration can achieve close to optimal behavior, with far less data than what is needed for traditional machine-based strategies.**

***Index Terms*—Decision support, machine learning, ramp-up, symbiotic human–machine systems.**

I. Introduction

RAMP-UP is the manufacturing phase during which a production system is brought up to its required operational performance, after its initial build or subsequent changes [1]. The long time required by this phase and its unpredictable nature are well-known issues in industry and academia [2]. Several studies have investigated these issues and highlighted the adverse effect of ramp-up on the overall economic viability of a production system [3]. Prolonged ramp-up has significant effect on companies' abilities to invest in more complex manufacturing systems, to respond to new market opportunities, and to readily react to changes. Although the criticality of ramp-up is widely recognized, no clear solution could be found so far to resolve it, consequent to which it continues to be a major bottleneck for new system introductions and changeovers.

During ramp-up, the technical operators apply actions to improve the system's performance. They often follow a trial and error approach that gives them the experience and understanding of the system's behavior [4]. However, the operators' experience is not well utilized and often gets lost between different systems and operators. This is caused mostly by inefficient communication and staff turnover [5]. Additionally, ramp-up sessions are often repeated across production lines with similar characteristics and the lack of transferable experience results in multiple repetitions of the ramp-up. Those issues are well highlighted in the literature, with stress on the need for a more effective approach to capture and share experience amongst the humans involved in the process [6]. Several studies have indicated that extracting ramp-up knowledge is a potential solution to capture human experience [4], [7]. This requires a formal ramp-up model and a learning mechanism, which can work across different ramp-up cases.

The need for human–machine collaboration during assembly and for efficient communicating interfaces has been demonstrated in the literature [8]. Hence, in their previous work, the authors have developed a formal framework for the capture and analysis of ramp-up processes [9]. The framework combines interfaces that capture human observations along with automatic recording of information on machine status, besides formally representing the ramp-up state of a machine. This combination allows for extraction and reasoning of the cause-and-effect relationships between actions and their impacts. Machine learning algorithms have been shown to be capable of extracting more effective decision-making policies from several recorded ramp-up episodes [10].

Humans are very good at reasoning under conditions of uncertainty and incomplete data, while machines are good in storing, retrieving, and processing data, quickly and in a repeatable manner. Hence, it is suggested that creating a symbiotic environment which builds on their respective capabilities will be

Manuscript received August 31, 2016; revised February 15, 2017; accepted May 4, 2017. This work was supported in part by the European Commission, as part of the FP7 NMP FRAME project (CP-FP 229208-2), and in part by the EPSRC Centre for Innovated Manufacturing in Intelligent Automation (EP/IO33467/1). This paper was recommended by Associate Editor Dr. David B Kaber. *(Corresponding author: Niels Lohse.)*

S. Doltsinis is with the Department of Automation, Piraeus University of Applied Sciences, Aegaleo 12244, Greece (e-mail: doltsinis@teipir.gr).

P. Ferreira is with the Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough LE11 3QZ, U.K. (e-mail: P.Ferreira@lboro.ac.uk).

N. Lohse is with the EPSRC Centre for Innovative Manufacturing in Intelligent Automation, School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough LE11 3QZ, U.K. (e-mail: n.lohse@lboro.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/THMS.2017.2717885

mutually beneficial as suggested in [11]. Therefore, to generate better ramp-up strategies, human capabilities for understanding and exploring new solutions need to be combined with machine capabilities for faster processing and systematic pattern recognition.

This work combines human and machine capabilities and focuses on investigating the learning efficiency, with limited data, by proposing a human expert guided exploration strategy for reinforcement learning during ramp-up. The proposed approach hypothesizes that a human operator guided exploration strategy will be more efficient than a purely algorithmic one. To test this hypothesis, a comparative study of the existing exploration strategies was carried out to find the most effective ramp-up policy. The efficiency of different strategies, along with the resulting characteristics of the policy and completeness of the state space, was assessed using a ramp-up simulator.

The remainder of this paper is organized as follows: Section II presents a review of the literature on the learning approaches developed for ramp-up; Section III proposes the symbiotic human–machine learning approach to support the ramp-up process; Section IV presents the experimental process, using a ramp-up simulation model of a real production system, and discusses the results; finally, Section VI presents the conclusions drawn from this study.

II. Knowledge and Learning During Ramp-Up

The problems of long ramp-up times were extensively researched, over the years [2]. A number of researchers analyzed the characteristics that influence the ramp-up and suggested several solutions to address different problems involved in this process [12], [13]. Ramp-up includes all the processes, right from the top strategic management decisions down to the lower level technical procedures [1]. The need for decision support is a common requirement of all the processes across different levels. In the shop floor level, ramp-up happens in new production lines, where newly built production systems operate suboptimally. Numerous equipment variations, building procedures, and production processes render ramp-up behavior unpredictable and stochastic.

Many researchers stressed the need for a formal model that captures data in a structured manner to utilize the experience gained during ramp-up and to draw conclusions in the form of reusable knowledge [4], [14]. Terwiesch and Bohn, for instance, studied the effect of learning from different sources during ramp-up and demonstrated the link to process-improvement [4]. They presented a dynamic model to balance the needs for deliberate learning through experiments and for increased production yield. Fjallstrom *et al*. highlighted the effect of learning during ramp-up and studied different types of information that represent relevant experience to support this phase [1]. They show that experienced personnel tend to seek equipment and process specific information that provides better ramp-up results. Hansen and Grunow highlighted the effect of experience on effective production capacity [7]. They modeled the ramp-up as a process of changing effective capacity (capabilities, based on gained experience) over time to provide a decision support tool for capacity expansion in the pharmaceutical industry.

Current research efforts build on the idea of enhancing a system's embedded intelligence and enhancing the data already collected during preproduction phases. The intelligent network devices for fast ramp-up EU project, for instance, aims at achieving fast ramp-up through networking between intelligent devices [15]. The concept is based on agentification of production devices by enhancing them with standardized interfaces and communication protocols to reduce their integration effort. The structure of the agent allows for negotiation between the devices on dynamic agreement of their roles in the production process. This helps in reducing custom programming effort to facilitate rapid changeovers, but does not directly address the need for parameter tuning. The adaptive production management EU project focuses on integrating information and communication technology (ICT) based solutions to support real-time decisions into planning and operation during ramp-up [16]. Leitao *et al*. presented an overview of the project's architecture, which combines the functionalities, such as ontology services, data transformation, key performance indicators (KPIs), etc., implemented through multi-agents and intelligent enterprise service bus [17]. Ramp-up times can be reduced by capturing previous ramp-up experience and utilizing the knowledge, so accumulated, to more effectively select actions during future processes [18], [19]. But, this cannot be done based on the available sensor information alone, and requires close collaboration between operators and machines to capture human actions during ramp-up and their decision making strategies.

Ramp-up depends mainly on the knowledge and expertise of the operators, who tune the production system until its performance is stable and the disturbances are reduced to the minimum. This process is very complex, requiring an intimate understanding of the equipment, the production process, the product, the required performance, and the targeted quality. There are many sources of uncertainties and variations that render ramp-up a highly unpredictable and stochastic process and thus difficult to model in advance. However, there are a few published works on modeling the required information, besides enabling learning.

Konrad *et al*. presented a generic semantic model for fast ramp-up modeling, which combines human decisions data with machine data [20]. The use of semantic technology allows for context-sensitive exploration of past ramp-up cases and thus provides the foundation to employ learning techniques to discover more effective ramp-up strategies. Oates *et al*. used machine learning to learn from previous ramp-up situations in a case-based reasoning approach to provide decision support during the ramp-up. They reported an algorithm, based on the k-nearest neighbors algorithm, to generate decision support for case-based queries [21]. The algorithm performs well on similar cases, but not so well on unknown states. The authors presented a different approach where they focus on the sequential nature of ramp-up and the advantage it creates for reinforcement learning algorithms [10]. They define ramp-up as a sequence

of state-transitions, driven through the adjustments applied by operators. They used a reinforcement learning algorithm to correlate state changes to performance, and to extract an efficient decision-making policy.

Despite the increasing interest in ramp-up and numerous on-going investigations of different decision-support methods, based on experience-capture and learning, no clear strategy could be evolved so far to gaining experience. Against this background, this paper proposes a symbiotic human–machine learning approach to guide exploration, within the space of the ramp-up process, which mainly maximizes the benefit of the gained experience.

III. Symbiotic Human–Machine Learning Approach

Providing support during ramp-up has so far been focused predominantly on building frameworks to capture data and information in a structured manner. This provides the basis for analyzing, and ultimately for guiding the process in a systematic manner [14]. Cognitive science and machine learning approaches are being increasingly investigated to provide more intelligent decision support [22]. These efforts have led to the concept of a symbiotic framework [23], wherein the human intelligence, which can capture the right data, can be combined with the machine's ability for fast and exhaustive data processing to achieve effective learning.

The literature supports the notion of applying machine learning for improving the effectiveness of ramp-up, while it simultaneously stresses the challenge of limited data. Reinforcement learning, in particular, has been identified as a promising approach to address this challenge [10], [24]. This involves learning of a policy from experience and interaction with the system, without the need for significant preparation and predefinition of a model.

The ramp-up process includes a number of characteristics that render learning from experience a difficult, but a promising, task. Previous studies of the process show that operators follow a sequential procedure in which they apply changes to tune the system, based on monitoring its performance changes [20], [25]. The experience so gained contributes to enhancing the operator's learning curve, which helps in handling unexpected events, but it also delays the process. The uncertain and often stochastic consequences of a change (action) render learning from experience difficult. This results in slowing down the ramp-up process due to unnecessary exploration.

The authors propose here a ramp-up learning and decision support framework, in which the operator carries out the targeted exploration, supported by a computationally powerful device (see Fig. 1). This is de facto realization of the symbiotic assembly concept proposed in [11], which combines free exploration by people with complex pattern recognition by machine learning. The actions of an operator and their consequences, captured as a Markov decision process (MDP) model, constitute a rich history of previous experience. The MDPs are most commonly used to model such sequential and stochastic processes [26]. The MDP model has a number of characteristics that can benefit formal analysis and optimization of the ramp-up. It allows for the mapping of correlations between states and adjustments (actions), and supports the learning of the associated transition probabilities. The MDP model can be solved to find a policy that maximizes the system's output (reward). For ramp-up, this amounts to achieving the targeted goal state as fast as possible or, more simplistically, using the least number of steps (state-transitions). Reinforcement learning is used to solve MDPs without having prior knowledge [27].

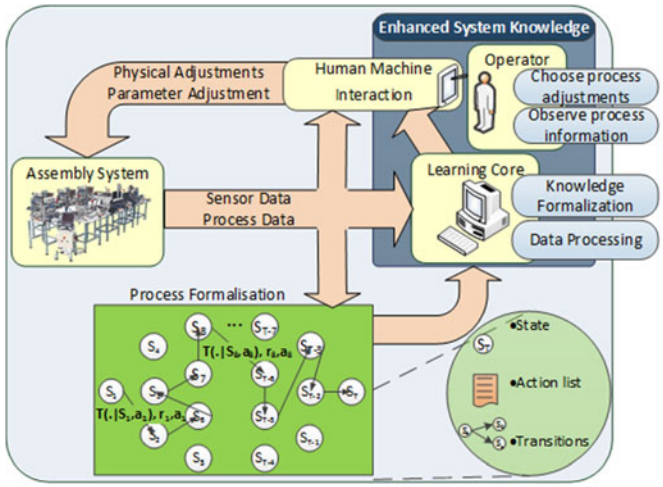

Fig. 1. Symbiotic ramp-up cycle.

In the proposed approach, the ramp-up states (S) of an assembly system are defined by a finite set of observable characteristics (p). To change the behavior of the system, the operator can apply a finite set of discrete actions (a). The learning core captures the transition probabilities $T(.\,|S_n,\,a_n)$ from a state (S_n), as the result of an applied action (a_n), and uses a reinforcement learning algorithm to extract the most effective ramp-up strategy (policy) from successful ramp-up episodes [28].

The transition probabilities are learned through the learner's interaction with the environment. Alternatively, a policy can be learned directly, and it is even possible to learn a partial model [27]. This can accelerate learning by generating a policy, based on a combination of successful ramp-up sessions (episodes). Another advantage of model-free approaches is that the algorithms can compensate for nonstationary environment. The change in the environment will be captured by learner's exploration, which will update its policy.

The authors have previously analyzed ramp-up data, using a Q-learning algorithm [10]. These experiments focused on the algorithm's ability to converge even with limited data, achieving promising results. However, those results fail to address the problems arising from limited data during ramp-up. This is best overcome using better quality data and by infusing human intelligence into the learning process. This can be achieved by creating a human–machine symbiotic environment and improving the algorithm's performance.

To determine the viability of the proposed approach for ramp-up, the overall performance of the learning algorithm and the optimality of the exported policy were assessed. Two main elements can affect the performance of a reinforcement learning

algorithm, in terms of the learning process and the quality of the exported policy [27]: the update rule and the exploration strategy of the algorithm. They both control how well an algorithm can cope with limited data during ramp-up, besides controlling the algorithm's speed and convergence.

This work expands on previously published results and explores for the most effective exploration strategy for learning during ramp-up, based on the Q-learning algorithm. The ultimate aim is to compare different exploration strategies and benchmark them against the proposed symbiotic learning approach, using human-guided exploration.

A. Update Rule

The update rule in reinforcement learning (RL) algorithms refers to the approach used to update a state value when new data become available. The Q-learning algorithm uses an update rule which estimates the state action value $Q'(s,a)$ by comparing it to the existing estimate $Q(s,a)$. The new value is a combination of the previous $Q(s,a)$ one, plus the immediate return (r) from the new experience, and the difference between the best known subsequent state action value $Q(s',a')$ and the existing estimate $Q(s,a)$. The update rule of the Q-learning algorithm is defined as

$$Q'(s,a) = Q(s,a) + \alpha\left[r + \gamma \cdot \max_{a'} Q(s',\ a') - Q(s,\ a)\right]. \tag{1}$$

The two important characteristics of the algorithm's update rule that affect the convergence and the output policy are the discount factor γ (gamma) and the learning rate α (alpha). The first weighs the effect of the existing best state action value and the second controls the significance of a new value for the previously estimated state action value. Small learning rates should result in slow convergence, minimizing the risk of instabilities and oscillations between two values. However, very small values require a high number of samples before the algorithm converges to a final value, and this renders this approach difficult to use for ramp-up.

In previous works that applied Q-learning during ramp-ups, large gamma values and small alphas resulted in better ramp-up policies [28]. The gamma chosen should be large enough, but not extremely large (equal to one), since the algorithm's aim can change. Alpha value has been shown to affect the number of iterations required to achieve convergence. Although large values allow for faster convergence to a policy, the exported policies were shown to perform poorly. Therefore, the value chosen should be small enough, but not too small (approaching zero), or else the algorithm will converge very slowly. Also, in the reported experiments, it was assumed that infinite number of episodes would be available, and constant exploration would be possible. These assumptions are not realistic for a ramp-up, where data are very limited. One way to overcome this problem is the experience replay approach, developed by Kalyanakrishnan and Stone [29]. In their approach, the data from an episode are processed several times to propagate the effect of change in a state action value and eliminate data noise. By this approach, faster learning can be achieved even with a limited number of ramp-up episodes. Experience replay allows the partial state space to be fully fitted, but does not discover any unknown states. Discovering the full behavior model of a system requires a much more exhaustive exploration strategy. During ramp-up, only a fraction of the state space will be used, and exhaustive exploration does not provide a proportional benefit compared to the large extra effort it would require. It needs to be noted that a learning algorithm can discover the best policy only when the quality of the provided dataset is equally good. In this work, the experience replay method has been tested to find an optimal policy from limited data. However, if the transitions within the data do not include a good or optimal policy, then it will not be possible to generate one either. The quality of the data is a result of the exploration strategy, which can be enhanced by human exploration.

B. Human Exploration Strategy

The exploration strategy is a key aspect of any model-free algorithm, because the entire space might never be explored, unless the exploration strategy specifically aims for it. Most of the exploration strategies aim to balance exploration against exploitation by focusing on only the most relevant part of the wider state-space. For the ramp-up case, it is proposed to limit exploration to the choice of the operators, as they are intrinsically aware of the most relevant actions, though not necessarily of the consequences or most effective overall strategy. Hence, it is expected that by allowing the operator to choose the exploration strategy, better results can be obtained compared to any standard algorithmic exploration strategy. Also, by doing so, no artificial disturbances will be created in the process, because of prior knowledge of the operators and their ability to learn, in parallel to any algorithm, during the ramp-up. No doubt, such an approach might leave certain states and actions unexplored, but the outcome is not necessarily detrimental, because the extra effort that goes into exploration of rare states might be much higher than the additional benefit that might accrue from such exploration.

Finally, it needs to be stressed that the exported policy does not aim at finding the optimal performance of the system that is being ramped-up, because that would require several ramp-up episodes, covering every state-action combination. Owing to its stochastic nature, ramp-up can be achieved in different ways, by applying different policies and depending on the nature of the system behavior. Therefore, it is reported that policies, defined during the development of a production line, can fail to support ramp-up on site [1]. Thus, the exported policy has to be optimal for a small number of datasets, because only limited data will be available during the actual ramp-up, when the stochastic behavior of the system will become apparent. Clearly, better policies, in terms of time or even overall performance, should exist if the system can be observed over a longer period of time. The algorithm's results can, therefore, be expected to improve as new episodes arrive and the operator acquires a better understanding of the system, say from subsequent system reramp-ups, e.g., following breakdown recoveries or changeovers. In this paper, the emphasis will be on the quality of the data that

can be gained from smaller sets of ramp-up episodes, which is assumed to be better when generated by a human operator. The

Algorithm 1: Q-learning algorithm for ramp-up..

Initialize $\pi(s), Q(s,a)$,
Choose large $\gamma \geq 0.9$ and small $a \leq 0.05$
generate a ramp-up episode restricted to human exploration
for each step:
update
$Q'(s,a) = Q(s,a) + \alpha[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)]$
end of the step
Replay the batch until
$d\,Q' = \alpha[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)]$
is stable.
Until there are no more episodes find the optimal policy
$\pi(s) = \operatorname{argmax}_a Q(s,a)$

aforementioned methods and the key aspects of the proposed learning approach are summarized in a reinforcement learning algorithm, presented as Algorithm 1. The proposed algorithm will be tested during ramp-up support and compared with other exploration strategies.

IV. Experimental Set-Up

Many experiments were carried out to assess the functionality and the efficiency of the ramp-up learning algorithm. The aim is to compare the efficiency of the human exploration based algorithm with those of the other established exploration strategies and study its effect on the generated data. All the experiments were carried out with the ramp-up data generated from a ramp-up simulator that was developed, based on an industry-like assembly system test bed. The simulator was used to automatically test different exploration strategies, allowing for various degrees of exploration in good time, which would not be possible in a real physical system. Further details of the experimental procedure and the ramp-up simulator are presented in the following sections.

A. Simulation Model

A production station ramp-up simulation model was developed to generate ramp-up sessions for testing and evaluating different learning algorithms. The model was formulated as an MDP, and according to the proposed ramp-up model [10]. The production station runs three processes sequentially, as shown in Fig. 2.

The station delivers a box, filled with a predefined amount of plastic components to emulate functionality, common to pharmaceutical and food industry. First, a pick and place process is triggered where a box is picked, checked, and delivered to the processing area. There, the filling process assures that the desired amount of product is placed into the box and then another pick and place process removes the box out of the station area.

The MDP model includes the state parameters, the action list, and the reward function. The following are the actions available:

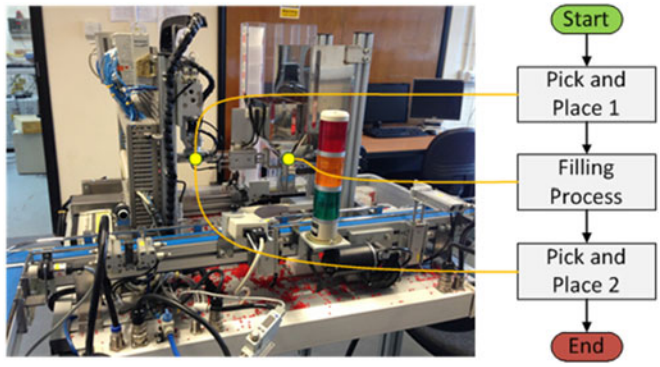

Fig. 2. Assembly station process sequence.

TABLE I
State Variables

Not.	Description	Values
p_1	The duration of the first pick and place process	Acceptable/Unfunctional/Slow
p_2	The duration of the filling process	Acceptable/Unfunctional/Slow
p_3	The duration of the second pick and place process	Acceptable/Unfunctional/Slow
p_4	The weight of the produced product discretized within acceptance and rejection limits.	Less/Ok/More
p_5	Signals observed by an operator but cannot be captured by a sensor.	Spilling/Box Lost/None

a1: increase pressure (Yes/No); a2: reduce pressure (Yes/No); a3: increase limits (Yes/No); a4: reduce limits (Yes/No); and a5: align gripper (Yes/No). The state variables are presented in Table I. A detailed description of the states and actions, as also their full definitions, can be found in [28].

An appropriate reward function for ramp-up is defined in [9] and the same can be seen in (5):

$$f_f(j) = -\sum_{j=1}^{n} k_j D_j \tag{2}$$

$$f_q(j) = -\sum_{j=1}^{n} \lambda_j Q_j \tag{3}$$

$$f_o(j) = -\sum_{j=1}^{n} \beta_j T_{o_j} \tag{4}$$

$$R_{\mathrm{PM}} = \frac{1}{s_t} w \cdot f_{\mathrm{RU}} \tag{5}$$

where $w = [w_f, w_q, w_o]$ is the weight vector and $f_{\mathrm{RU}} = [f_f, f_q, f_o]$ is the performance metric's vector, composed of three individual metrics relating to the functionality of the system, the output quality, and the process optimization. These are presented in (2)–(4), respectively. Parameters k_j, λ_j, β_j denote the weights and D_j, Q_j, T_{o_j} the state parameters of the MDP model. The reward function varies from –1 to 0, where –1 indicates the worst state and 0 the best, which also denotes the end of an episode.

The size of the state space is based on the number of state variables and their number of values. In this system, with 6 parameters and 3 discrete values for each parameter p, the state space has 243 states.

The dynamics (transition probabilities) of the simulation model are defined, based on a combination of expert knowledge and experimentation. For building the simulation model, transition probability matrices were generated for every state-action combination. This led to the generation of 5 transition probability matrices, one for every action during the 243 states, resulting in 1235 state-action probabilities. The size of the model highlights why it is not plausible to explore the entire state-space during a normal ramp-up, particularly considering the stochastic nature of the process and the resulting model.

B. Experimental Procedure

The proposed algorithm incorporates two main aspects; it combines experience replay in order to gain the maximum information from the data and introduces human exploration. The experiment was conducted in two parts. First, the efficiency of the batch learning method was tested for different learning values (alpha) along with the efficiency of the exported policy. The efficiency of the batch learning method was determined, based on the algorithm's ability to converge on stable Q values. The algorithm was tested, with and without the experience replay, for ten datasets (ramp-up episodes), under conditions similar to those of real ramp-up applications, where the number of available datasets is limited. The ramp-up datasets were randomly generated from the simulator, independent of the exploration strategy followed, aiming to test only the effect of the experience replay mechanism.

In the second part of the experiment, the effectiveness of the human exploration strategy was tested and compared with that of the other well established exploration strategies: 1) random exploration, 2) greedy exploration, and 3) greedy exploration with increased probability. Hence, the experiment focused on assessing the efficiency of the exported policies for different exploration strategies, with or without experience replay. An overview of all the four exploration strategies (automatic and human based) is given as follows:

Random exploration: This strategy applies experience replay on a dataset, which was generated by randomly choosing different ramp-up actions.

Greedy exploration strategy: Greedy exploration aims to choose those actions that can receive the highest possible reward. This is done with a fixed probability, big enough to assure the best choices. In this work, the greedy probability E_g was chosen to be 0.9 to reflect the greediness of the strategy.

Increased greedy exploration strategy: This exploration strategy is greedy with an increasing probability, rather than a constant. Like the previous strategy, it aims to choose actions with the best potential reward, but changes the probability of choice throughout a ramp-up episode. The probability decreases as a function of the number of iterations, which, in effect, means that the probability of choosing a random action is high at the beginning of each episode, but decreases as the iterations of the policy increase. Hence, the strategy becomes greedier. The probability of exploration is given by

$$E_{ig} = 1 - \left(\frac{1}{\log\left(\text{iteration} + 2\right)} \right). \tag{6}$$

During each set of experiments, the number of iterations increases from 1 to 70 and the probability of exploration E_{ig} from $\sim$0.23 to $\sim$0.91.

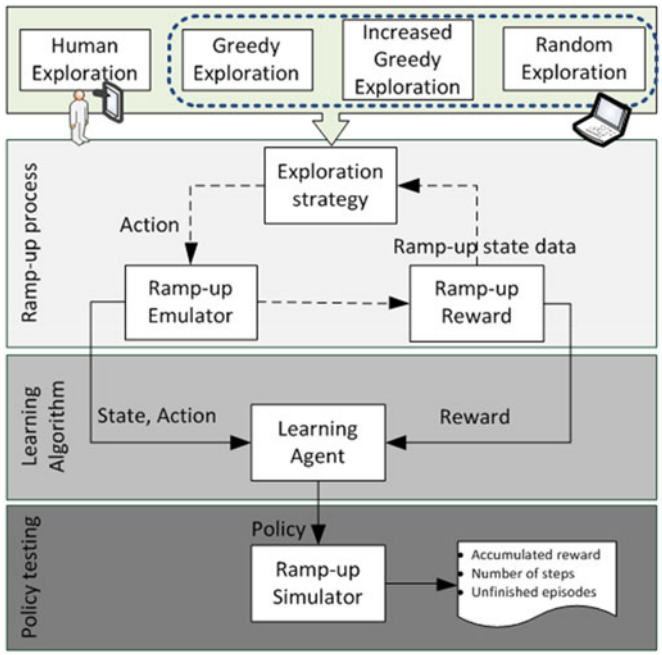

Fig. 3. Experimental set-up overview.

Human exploration: The human exploration strategy uses inputs from human experts, who select the actions during ramp-up. During the experimentation, ten different people (operators) were asked to ramp-up the same emulated assembly station that was used for the other three exploration strategies. The operators were first provided with information about the system functionality and normal operation. The ramp-up process was described, along with all possible actions that can affect the system's operations and the required targets for ramp-up. Thereafter, the operators carried out the ramp-up, step by step. During each step, they were interacting with the emulator through receiving information, such as cycle times, product quality, and performance measures. The operators would then choose to apply an action, based on the situational context presented. The result of the action would be computed and the new status information fed back to the operator to choose the next action to apply. This process was continued until the targeted ramp-up state was achieved. The ramp-up was carried out using the simulator described above.

Fig. 3 presents an overview of the underlying logic used during all the experiments and shows how individual functions take place. The same setup was used for all the exploration strategies. The exploration strategies were applied, one after the other, to the ramp-up simulator to extract specific ramp-up datasets.

These were then processed, using Algorithm 1, to export the corresponding policies. After applying the experience replay to five and then ten ramp-up episodes, the policies attained from different exploration strategies were compared with those exported from the same number of episodes, without applying experience replay. This was done to compare the exploration strategies, as also the effects of experience replay. Number 10 was chosen for the number of episodes to reflect a realistic ramp-up scenario and number 5 to capture the effect of limited data. All the exported policies were then applied to the ramp-up simulator to test their performance. To obtain statistically sufficient results, each policy was applied to 1000 randomly generated ramp-up episodes. To avoid the problem of endless episodes from bad policies, the number of steps per episode was limited to 1000. The above-mentioned process was tested ten times (for each set of 1000 episodes) for every policy and the results were averaged.

V. Results and Analysis

In the first set of experiments, the focus was on understanding the effect of batch learning on limited ramp-up data scenarios, as described in "Section IV." The evaluation criteria for the batch learning mechanism are the Q matrix hash function and the Q-value discrepancy. The hash function calculates the unique number for a matrix, which can therefore be used to infer every change in the exported policy. The nonzero Q value discrepancy indicates that the algorithm is still updating and that all the data information is reflected in the outcome.

In the following sections, the results of the experiments, using the four different exploration strategies, are presented and analyzed to understand which one of them is more efficient in converging onto a good quality policy. The following are the four strategies: 1) random choice of actions, 2) greedy strategy, 3) greedy strategy with increasing probability, and 4) a human-based exploration strategy, as presented in Section IV.

Four criteria were used for this comparative study:

1) the number of unfinished episodes,
2) the average number of steps required per episode,
3) the accumulated reward per episode, and
4) the number of unsupported states.

The number of unfinished episodes shows how many times a policy has failed to finish the ramp-up process within the predefined limit (1000 steps). The number of steps required should be as small as possible to indicate that the policy is good. The accumulated reward indicates the quality of the states (in terms of the system's performance) that the ramp-up process has gone through during an episode. It should be close to zero. The unsupported steps indicate the number of cases for which the applied policy has no information regarding a state, because of which a random action has to be chosen to move to the next state. This value should be as close to zero as possible.

A. Experience Replay

The results of applying experience replay on the simulator, as described for the first set of experiments, are shown in Fig. 4 which shows the changes in Q-value discrepancy and the policy hash function, while replaying the experience of ten datasets

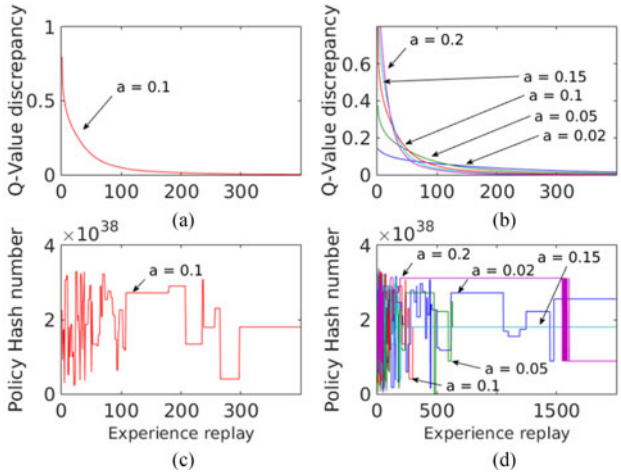

Fig. 4. Q value discrepancy for every experience replay iteration for (a) alpha = 0.1, (b) different alpha values, and policy hash change for (c) alpha = 0.1 and (d) different alpha values.

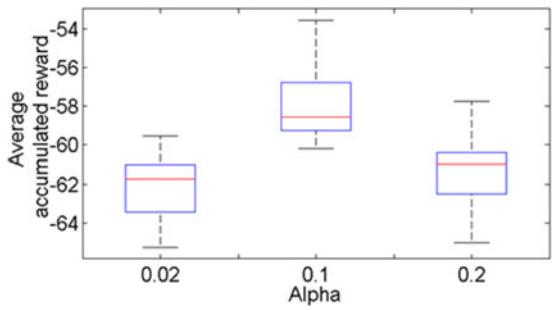

Fig. 5. Average accumulated reward for different alpha values after applying experience replay.

(ramp-up episodes). The graphs show the effect of experience replay on the learning process, through Q-value and policy change.

Fig. 4(a) and (c) shows that the policy changes significantly during experience replay, but stabilizes after about 300 iterations of the whole dataset. As a result, all the information of the dataset was extracted and reflected in the policy. By varying the learning rate (alpha) of the algorithm, it can be seen that small alpha values increase the number of iterations required to achieve convergence and larger ones reduce it. Fig. 4(b) and (d) shows the changes in Q-value and policy discrepancy for different alpha values. The alpha values were chosen to be small, but not extremely small, based on the results of previous research [28]. For three of the values (0.05, 0.1, and 0.5), the algorithm converges to the same policy, requiring about 600, 300, and 200 iterations, respectively. For values, smaller or larger than these, the algorithm converges to different policies. In the case of larger values, this can be explained as due to the big change in the Q-value, which is controlled by alpha and does not provide enough resolution to the algorithm. In the case of a very small alpha, the algorithm does eventually converge to the same policy, but after numerous iterations. The average accumulated reward [based on (5)] for the three policies, namely for alpha equal to 0.02, 0.1, and 0.2, is, respectively, 62.11, 57.97, and 61.15, as shown in Fig. 5. This confirms that the second policy (alpha = 0.1) has better quality.

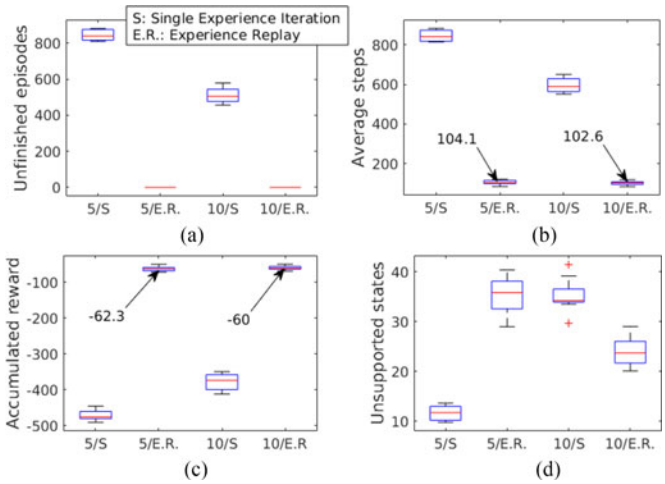

Fig. 6. Policy evaluation for random exploration in terms of (a) unfinished episodes, (b) average number of steps, (c) accumulated reward, and (d) number of unsupported states.

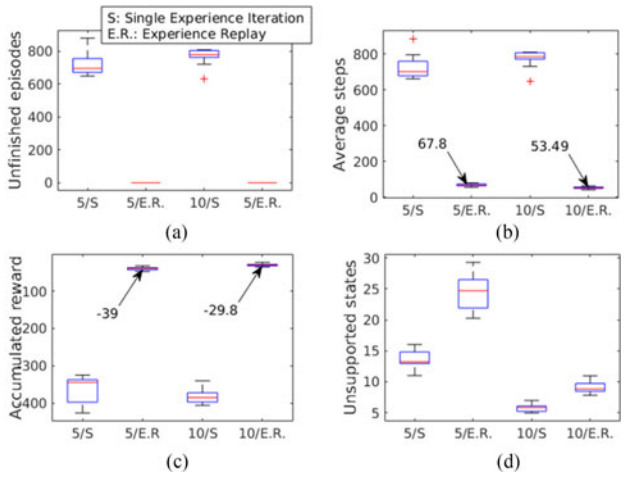

Fig. 7. Policy evaluation for a greedy exploration in terms of (a) unfinished episodes, (b) average number of steps, (c) accumulated reward, and (d) number of unsupported states.

The above-mentioned results highlight the significant effect of experience replay on the exported policy and which alpha value is more suitable for the current experimental setting. The effect of experience replay on policy quality is shown in the next part wherein the results of comparison are presented for all the aforementioned exploration strategies.

B. Random Exploration Strategy

The results of the exported policy for five and ten ramp-up episodes, with random action exploration, are shown in Fig. 6. Both single experience iteration and experience replay algorithm were applied to compare their effects on the quality of the resulting policy.

Fig. 6(a) shows that the number of medians of the unfinished episodes was reduced by ∼42% (840 to 505) after the number of episodes was doubled from 5 to 10. When experience replay was applied, the number of unfinished episodes became 0, for both 5 and 10 episodes, thus indicating that both the policies could successfully manage to guide all the ramp-up episodes to the end state. Fig. 6(b) shows that the average number of steps required per episode is roughly proportional to the number of unfinished episodes, except for a small difference between the two cases for which experience replay was applied. It can be further seen that the policy derived from ten episodes requires, on average, 1.5 fewer steps. The same relationship is also reflected in Fig. 6(c), which presents the accumulated reward per episode. On average, the policy from ten episodes accumulated –2.35 more reward. Considering that the reward is negative and that it varies from –1 to 0, this reduction indicates that the policy is better. Fig. 6(d) shows a different perspective by showing the number of cases where the policy could not support the ramp-up process, because of which a random action had to be chosen. Contrary to expectation, this graph shows that the number of unknown cases is very small for the first case, even though experience replay was not used and the number of episodes to learn the policy was small.

When considering the large number of unfinished episodes, it can be seen that this is due to a poor policy which creates oscillations between states. This artificially reduces the number of unknown cases. This number is expected to increase when experience replay is applied, because the new policy better reflects the optimal combination of the datasets and can generate much different paths. When the number of the episodes becomes big enough to generate a more complete policy, the unsupported states should decrease. Finally, the policy generated from ten episodes, without experience replay, resulted in a large number of unsupported states, which do not follow the expected trend. This is because, the dataset becomes big enough to have enough data over a wider part of the state space, but because of using single experience iteration, optimal information is not well reflected in the final policy and therefore the policy again oscillates between states.

C. Greedy Exploration Strategy

Results for five and ten ramp-up episodes, after applying greedy exploration with 0.9 probability, are presented in Fig. 7.

Fig. 7(a) shows that the number of unfinished episodes was 0 for both the policies exported with experience replay algorithm, whereas it is significantly high for those to which experience replay was not applied. The same trend can be observed for the required number of steps per episode and the accumulated reward. The policy exported from the bigger dataset requires the least number of steps and accumulates the largest reward. From Fig. 7(b), it can be seen that, although the second policy (5/E.R.) presents very good results in terms of the accumulated reward and the required number of steps, the number of unsupported steps is higher than that of the other policies. This reflects the effect of the training data size on the policy. However, this is not the case for policies without experience replay, because they require a large number of steps due to oscillation between nonfinal states.

From a comparison of the best policies exported, using greedy and random exploration strategies, it can be seen that the former

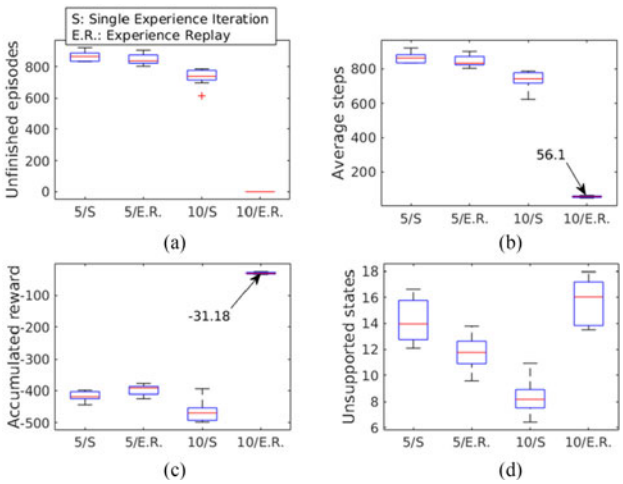

Fig. 8. Policy evaluation for increased greedy exploration in terms of (a) unfinished episodes, (b) average number of steps, (c) accumulated reward, and (d) number of unsupported states.

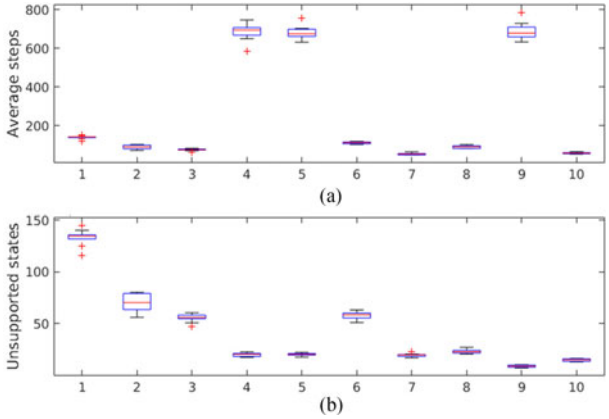

Fig. 9. Policy comparison in terms of (a) average steps and (b) unsupported states, for increasing number of ramp-up episodes.

strategy required an average of 53.49 steps, and the latter 102.6. This shows that greedy exploration strategy gives a significantly better policy.

D. Increased Greedy Exploration Strategy

Fig. 8 presents the results after applying the policies exported from five and ten ramp-up episodes, using the increased greedy exploration strategy.

The behavior of the applied policies is similar to that of the policies exported from episodes, using the greedy exploration strategy. The experience replay, when applied to all the episodes, generates a very good policy with an average accumulated reward of –31.18 and average required steps of 56.1. The performance of the policy is similar, but slightly inferior, to the policy exported from greedy exploration. A significant difference compared to the previous cases is that the policy presents poor performance, when experience replay was applied after five episodes. This is due to the small size of the dataset, which contains a small number of nonoptimal actions, and the number of episodes is not enough to guarantee good exploration. This changed with the second set of five ramp-ups, when the policy performed better.

E. Human Exploration Strategy

To test the efficiency of human exploration, ten ramp-up episodes were carried out under the guidance of different people. These episodes were processed, and a policy was exported for every new episode. This is to realize the value of every new human-based episode and its effect on the efficiency of the policy. The results of applying the exported policy are presented in Fig. 9.

The average numbers of the required steps and the number of unsupported states are presented for every policy. Fig. 9(a) shows a continuous improvement of the policy during the first three episodes, but after the fourth episode, a rapid change occurred resulting in a very poor policy. Thereafter, the policy kept improving until the seventh episode, but after the ninth episode, it again became poor. However, the final episode improved the policy, giving the second best policy, in terms of the required number of steps. Fig. 9(b) shows a constant improvement of the policy from the first to the fifth episode, in terms of the number of unsupported states. The performance became poor after the sixth episode, but thereafter the results started improving until the last episode with small variations.

Considering the two graphs together, the following conclusions can be drawn. With successive episodes, the number of unsupported states decreases. With the addition of a new episode that contains unexplored states, the performance drops, but the number of unsupported states decreases. This trend continues for a few more episodes with the same states, until the policy matures for these set of states. This is repeated until another episode with previously unexplored state(s) is added and the performance drops, but the number of unsupported states increases. This is expected to continue until new episodes do not any longer include unexplored states. At this point the policy will only improve.

The first seven episodes generate the best p, because they contain enough cases with the same states. On the other hand, the final policy need not be considered worse than the seventh policy, if the number of supported states is also reckoned to be an important performance indicator, because it indicates a more complete policy, which can produce more robust results. In that sense, the final policy is the most complete one.

A comparison of the different exploration strategies, in terms of average required steps and unsupported states, is presented in Fig. 10. From this figure, it can be seen that the random exploration policy gave the worst results, in terms of both the average number of required steps and the number of unsupported states. All the other policies gave similar results in terms of the required number of steps. The greedy exploration strategy gave the smallest number of unsupported states, because the amount of exploration was very high and the policy guided the decisions through known states. The increased greedy exploration strategy and the human-based exploration strategy yielded policies with very similar performances, because the human exploration is very similar to the increased greedy exploration. At the beginning of the episode, the human operators tend to explore until they acquire knowledge and then follow the same strategy.

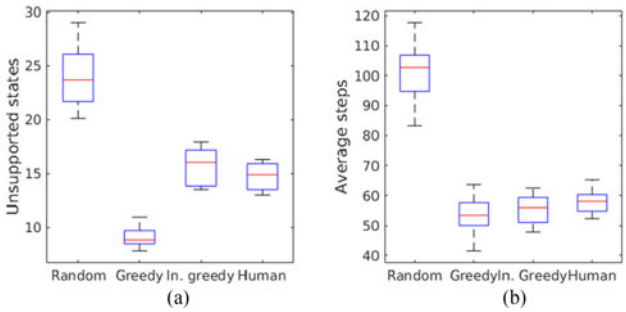

Fig. 10. Comparison of different exploration strategies in terms of (a) average required steps and (b) unsupported states.

TABLE II
P-VALUE AND *T*-VALUE RESULTS FOR THE *T*-TEST BETWEEN HUMAN EXPLORATION AND THREE OTHER STRATEGIES

	Random	In. Greedy	Greedy
Unsupported states (*p*-value)	4.5∗e–08	1.2∗e–09	0.1674
Average reward (*p*-value)	2.3∗e–10	0.0363	0.1802
Unsupported states (*t*-value)	8.98	–11.35	1.43
Average reward (*t*-value)	12.57	–2.26	–1.39

Overall, the performance of human exploration strategy can be considered very efficient, because human–machine collaboration during the ramp-up creates a good balance between the optimality of the solution and the efficiency of the exploration required. This symbiotic framework is more robust than the systems, based entirely on computers or humans. The similarities of the last two policies can also be observed in the generated datasets, which will be discussed in the next section.

To evaluate the statistical significance of the produced results, a *t*-test [30] was run on the data presented in Fig. 10. The test was run in pairs with human exploration strategy on one side and one of the other three strategies, on the other side. The *p* and the *t* values of the test are presented in Table II, from which it can be seen that the results are statistically significant for both random and increased greedy exploration strategies, but not for the greedy exploration strategy. Additionally, the mean value of greedy exploration is very close to that of the human exploration, as indicated by the *t* values. This finding need not be viewed negatively, because the proposed approach is not aimed at proving its superiority to the other two exploration strategies, but at highlighting the additional advantages it offers, as will be discussed in the following section.

F. Dataset Analysis

The exploration strategies directly influence the nature of the generated datasets that were used in exporting the policies. The quality of the dataset, in terms of its volume and information, gives an additional indication of the performance of the exported policy. In this part, the training sets exported by different exploration strategies were analyzed directly and the results are presented in Table III. The first two rows of the table give the number of visited states for ten episodes (all states/ten episodes) and for five episodes (all states/ten episodes); rows 3 and 4 give the number of unique states in every dataset, generated by each strategy; finally, row 5 gives the usage of exploration by each strategy, in comparison to the total visited states.

TABLE III
DATASET COMPARISON

	Random	Greedy	In. Greedy	Human
All States/10 ep.	640	641	638	290
All states/5 ep.	324	323	298	108
Unique states/10 ep.	161	115	107	109
Unique states/5 ep.	111	67	68	51
Exploration usage	640/640	530/641	194/638	N/A

The total number of states explored by each strategy gives an indication of the size of the dataset used in exporting a policy. It can be observed that, for the same number of episodes, the sizes of the datasets generated by the first three exploration strategies are very similar, while that of the dataset generated by human exploration is less than half their size. This trend becomes even more pronounced when only half the number of episodes is considered. Human exploration resulted in only 108 states. This confirms the expectation that a person would ramp-up a system faster and the significantly smaller number of required steps since they can use their understanding and wider knowledge of how the system operates. It also highlights the significance of human–machine collaboration and stresses the need for a symbiotic environment that enables such collaboration.

The number of unique states in each episode reflects the amount of exploration carried out by each strategy. Random exploration generated the most unique states, as expected. The other three explorations strategies generated almost the same number of unique states, i.e., 115, 107, and 109. This finding is very interesting in that it confirms that human exploration strategy is a lot more efficient, because, with half the ramp-up data, it manages to explore the same unique number of states and exports an equally efficient policy.

The exported data of every exploration strategy confirm the results presented in Section V-B and highlight the influence of exploration strategy on the exported policy. Overall, the greedy strategy can provide a thorough state-space exploration, both in terms of the number of states and the applied actions. However, in practice, it is not possible to automatically implement an action, and, therefore, an operator is always needed. This makes the greedy strategy more effort-intensive, as operator can otherwise ignore the less promising actions in each state. Besides, it helps in exploring a smaller proportion of the state space. The humans can apply their knowledge of the system's operation and support the ramp-up with less data. This capability is significant for ramp-up, because every change applied to a real system has to be decided carefully to avoid performance deteriorations and production delays.

Finally, a conclusion has to be drawn regarding the feasibility of implementation and practical limitations of the proposed symbiotic approach. The results presented are based on ramp-up simulation and therefore, their reliability depends on the quality of the ramp-up simulator. In practice, the stochastic nature of ramp-up cannot be simulated exactly, because of which the response of the real system will always be a little different from

that of the simulated system. Also, the actions applied by human operator, during a simulated ramp-up, might be much different from those applied during the actual ramp-up. Additionally, the implementation of the RL model can become very challenging in practice, because it is not easy to predefine the list of actions, whose recognition requires much effort during the development phase. The implementation of the proposed approach in large systems can also be challenging, unless it is broken down into simple workstations. Despite this challenge, this work shows the advantages of symbiotic human–machine collaboration and the impact it can have on reducing ramp-up time, with little data. Keeping these challenges in view, future research will have to be carried out to further enhance the benefits of the proposed symbiotic approach.

VI. Conclusion and Future Work

This paper proposes a reinforcement learning algorithm where human exploration can be implemented. The proposed algorithm is based on reinforcement learning due to its ability to operate without a predefined model. For generating ramp-up data, a simulation model of an assembly station was used. The proposed algorithm was designed, based on Q-learning and the positive results of previous studies. The algorithm uses experience replay to reduce noise from the data and apply human exploration to improve its quality. The results obtained from the experiments facilitate, for the first time, a comparison between different exploration strategies, with focus on human exploration and the advantages of human–machine symbiosis. The human exploration strategy exploits the intrinsic system-operating knowledge of the humans and their ability to reason under uncertainty.

The results show that the experience replay approach significantly improves the quality of the policy with fewer datasets. The application of different exploration strategies reveals the significant effect they have on the quality of the exported data and the final policy. It is found that random exploration strategy will not lead to good results, unless unlimited data are available. The greedy exploration strategy, on the other hand, can generate good results, but does not take advantage of human knowledge. Even though the results of greedy exploration are marginally better than those of the human-based exploration, the greedy exploration strategy visits more states, implying a longer ramp-up time. The human exploration strategy takes advantage of human's perception and intelligence in generating better data and transforming them into knowledge. This shows how human–machine symbiosis can be established with mutual benefit to both sides. Humans can offer intelligence in taking the best decisions, while the machines can uncover underlying patterns and extract more optimal policies.

Despite the promising results of this work, some more research is required for an in-depth understanding of the symbiotic human–machine learning during the ramping up of automated systems. Future work should include additional experiments in this direction, including inferential analysis, to further evaluate the differences amongst the approaches presented. It is clear from the present and previous works that this approach has the potential to significantly reduce ramp-up and changeover times. This reduction in ramp-up time is very important to allow faster new product introductions and smaller economic lot sizes. Besides, the development of formal models and pattern-recognition algorithms for ramp-up can help in communication of knowledge and wider utilization of the outcome from learning algorithms. This should lead to proposing new frameworks that can prove more effective for closer human–machine symbiosis.

While this work has demonstrated the potential of reinforcement learning, using an industry-like system, it is necessary to further test the proposed algorithm in full scale industrial applications. Also, further work is required to validate the feasibility of the proposed approach in actual production systems, in the context of more complex industrial environment, and to ensure its more generic applicability.

References

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

[2] L. Surbier, G. Alpan, and E. Blanco, "A comparative study on production ramp-up: State-of-the-art and new challenges," *Prod. Plan. Control*, vol. 25, no. 15, pp. 1–23, 2013.

[3] J. E. Carrillo and R. M. Franza, "Investing in product development and production capabilities: The crucial linkage between time-to- market and ramp-up time," *Eur. J. Oper. Res.*, vol. 171, no. 2, pp. 536–556, 2006.

[4] 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.

[5] 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.

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

[7] P. Taylor, K. R. N. Hansen, and M. Grunow, "Modelling ramp-up curves to reflect learning: Improving capacity planning in secondary pharmaceutical production," *Int. J. Prod. Res.*, vol. 53, no. 18, pp. 37–41, 2015.

[8] Y. Yin, L. Da Xu, and Z. Bi, "A novel human–machine collaborative interface for aero-engine pipe routing," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2187–2199, Nov. 2013.

[9] 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, no. 1, pp. 85–94, 2013.

[10] S. Doltsinis, P. Ferreira, and N. Lohse, "An MDP model-based reinforcement learning approach for production station ramp-up optimization: Q-learning analysis," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 44, no. 9, pp. 1125–1138, Sep. 2014.

[11] P. Ferreira, S. Doltsinis, and N. Lohse, "Symbiotic assembly systems - A new paradigm," *Procedia CIRP*, vol. 17, pp. 26–31, 2014.

[12] I. Niroomand, O. Kuzgunkaya, and A. A. Bulgak, "The effect of system configuration and ramp-up time on manufacturing system acquisition under uncertain demand," *Comput. Ind. Eng.*, vol. 73, no. 1, pp. 61–74, 2014.

[13] M. Haller, "Cycle time management during production ramp-up," *Robot. Comput. Integr. Manuf.*, vol. 19, pp. 183–188, 2003.

[14] C. H. Glock and E. H. Grosse, "Decision support models for production ramp-up: A systematic literature review," *Int. J. Prod. Res.*, vol. 53, no. 21, pp. 6637–6651, 2015.

[15] IRAMP-EU, "Intelligent network devices for fast ramp-up," 2013. [Online]. Available: www.i.ramp3.eu.

[16] ARUM-EU, "Adaptive production management," 2012. [Online]. Available: www.arum-project.eu.

[17] P. Leitao, J. Barbosa, P. Vrba, P. Skobelev, A. Tsarev, and D. Kazanskaia, "Multi-agent system approach for the strategic planning in ramp-up production of small lots," in *Proc. 2013 IEEE Int. Conf. Syst. Man, Cybern.*, 2013, pp. 4743–4748.

[18] 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, pp. 103–111, 2007.

[19] P. Bußwolder, F. Burgahn, M. Hübner, and M. Werker, "Classification of company-specific influence factors as part of a knowledge management system for ramp-up projects," *Procedia CIRP*, vol. 51, pp. 44–50, 2016.

[20] 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. Precis. Assem. Technol. Syst.*, 2012, pp. 163–174.
[21] R. Oates, D. Scrimieri, and S. Ratchev, "Accelerated ramp-up of assembly systems through self-learning," in *Proc. Precis. Assem. Technol. Syst.*, 2012, pp. 175–182.
[22] D. Bruckner, "Cognitive automation—Survey of novel artificial general intelligence methods for the automation of human technical environments," *IEEE Trans. Ind. Informat.*, vol. 8, no. 2, pp. 206–215, May 2012.
[23] N. Lesh, J. Marks, C. Rich, and C. Sidner, "Man-computer symbiosis' revisited: Achieving natural communication and collaboration with computers," *IEICE Trans. Inf. Syst. E Ser. D.*, vol. 87, no. 6, pp. 1290–1298, 2004.
[24] S. Doltsinis, P. Ferreira, and N. Lohse, "Reinforcement learning for production ramp-up: A Q-batch learning approach," in *Proc. 2012 11th Int. Conf. Mach. Learn. Appl.*, vol. 1, 2012, pp. 610–615.
[25] G. Schuh, J.-C. Desoi, and G. Tücks, "Holistic approach for production ramp-up in automotive industry," in *Advances in Integrated Design and Manufacturing in Mechanical Engineering*, A. Bramley, D. Brissaud, D. Coutellier, and C. McMahon, Editors. Berlin/Heidelberg, Germany: Springer-Verlag, 2005.
[26] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.
[27] M. Wiering and M. van Otterlo, *Reinforcement Learning: State-of-the-Art*. New York, NY, USA: Springer, 2012.
[28] S. Doltsinis, "A decision support system for ramp-up - A reinforcement learning approach," Univ. Nottingham, Nottingham, U.K., 2014.
[29] S. Kalyanakrishnan and P. Stone, "Batch reinforcement learning in a complex domain," in *Proc. 6th Int. Jt. Conf. Auton. Agents Multiagent Syst.*, vol. 5, 2007, p. 1.
[30] G. Casella and R. L. Berger, *Statistical Inference*, 2nd ed. Boston, MA, USA: Cengage Learning, 2008.

Stefanos Doltsinis (M'16) received his B.Sc. degree in automation engineering from the Automation Department, Technological Educational Institute of Piraeus, Aigaleo, Greece, in 2007, the M.Sc. degree in control systems from the University of Sheffield, Sheffield, U.K., in 2008, and the Ph.D. degree in manufacturing engineering and operations management from the University of Nottingham, Nottingham, U.K., in 2013.

He has worked as a Researcher with the University of Nottingham and Piraeus University of Applied Science. He is currently a Research Fellow in the Information Technologies Institute, Aigaleo, Greece. His research interests include production ramp-up, decision support systems, machine learning in manufacturing, robotics, and intelligent energy management.

Dr. Doltsinis is a member of the IEEE System Man and Cybernetics Society.

Pedro Ferreira received the Master degree in electrical, electronic and computer science engineering 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 has a brief experience as an IT consultant. He took up a Research Associate position with the University of Nottingham, and progressed to Research Fellow after completing the Ph.D. degree, finally becoming a Senior Research Fellow in 2012. In 2014, he became a System Analyst at Maersk Line IT, Denmark, and was promoted to a Technical Manager after a few months. In June 2015, he joined Loughborough University, Loughborough, U.K., as a Lecturer in manufacturing systems. His research interests include the vision of making intelligent, interconnected, and self-adaptable systems that integrate seamlessly into production environment.

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.

He is a Senior Lecturer with Loughborough University, Loughborough, U.K., and leading the Centre for Intelligent Automation. His research interests include the field of cyber-physical production systems focusing on systems modeling, robotics and autonomous systems, human–machine interaction, distributed control, and decision-support.

Dr. Lohse is a member of the IEEE System Man and Cybernetics Society, IEEE IES Technical Committee on Industrial Agents, and the IEEE SMC Technical Committee on Intelligent Industrial Systems. He has received a number of awards for his work in assembly automation.