# Modular reconfiguration of flexible production systems using machine learning and performance estimates

**D. Scrimieri** * **O. Adalat** * **S. Afazov** ** **S. Ratchev** ***

* *Department of Computer Science, University of Bradford, Bradford, BD7 1DP, UK (e-mail: {d.scrimieri,o.j.adalat}@bradford.ac.uk)*
** *Department of Engineering, Nottingham Trent University, Nottingham, NG11 8NS, UK (e-mail: shukri.afazov@ntu.ac.uk)*
*** *Institute for Advanced Manufacturing, University of Nottingham, Nottingham, NG8 1BB, UK (e-mail: svetan.ratchev@nottingham.ac.uk)*

**Abstract:** This paper presents an agent-based framework for reconfiguring modular assembly systems using machine learning and system performance estimates based on previous reconfigurations. During a reconfiguration, system integrators and engineers make changes to the machine to meet new production requirements by increasing capacity or manufacturing new product variants. The framework provides a method for automatically evaluating these changes in terms of impact on the performance of the production system, and building a knowledge base. Such knowledge is used to support future reconfigurations by recommending changes that are likely to improve the performance based on previous reconfigurations. The agent architecture of the framework has two levels, one for individual assembly stations and one for the entire production line. Knowledge bases of changes are built and utilised at both levels using machine learning and performance estimates. A prototype implementation of the proposed framework has been evaluated on an assembly production system in an industrial scenario. Preliminary results show that framework helps to reduce the time and resources required to complete a system reconfiguration and reach the desired production objectives.



*Keywords:* Reconfiguration, production systems, assembly, agents, machine learning.

## 1. INTRODUCTION

Modern markets are characterised by ever-changing product requirements, shorter product life cycles and a large number of product variants determined because of mass customisation. Manufacturers must be able to respond to dynamic markets quickly by adapting their existing equipment for making new products and meeting new levels of demand, as well as minimising disruptions. However, the reconfiguration of equipment require a feasible plan, which may involve a significant amount of resources and appropriate knowledge and skills. Therefore, manufacturing companies often prefer to buy new equipment and design new production systems, as opposed to reusing existing resources.

This paper presents an approach to capturing information on system changes part of reconfigurations, together with the corresponding performance impact, and producing reusable knowledge that can guide engineers in future reconfiguration scenarios. The knowledge is produced by a machine learning technique based on a k-nearest neighbour classification algorithm, which represents machine states in adaptation contexts as points in a multidimensional space. The proposed framework defines a multi-agent system in which agents are deployed at two levels, at the level of an individual station or for the entire production system, collecting sensor data, building and utilising knowledge on system adaptations. The framework has been applied to an assembly system, but it is general and can be used in other types of production systems. This research is closely related to smart factories and the fourth industrial revolution (Industry 4.0), in which machine data and intelligence play a major role.

The advantage of our solution is that knowledge on reconfigurations is captured automatically from machine data, without the need for eliciting it from system integrators, engineers or shop floor operators. Such knowledge is not limited to a number of known cases, but it can be generalised and applied to similar cases in new production scenarios with, for example, new product variants. The multi-agent system is useful to learn and reconfigure both locally, at the module or station level, and globally, at the production system level. New agents can be deployed when a module is added to the production system and the knowledge generated on similar modules can be shared.

The rest of the paper is organised as follows. The rest of this section reviews related research, limitations and

approaches. Section 2 presents the agent types of the multi-agent architecture. Section 3 describes the process of recording system changes. Section 4 introduces the automated learning technique. Section 5 discusses an experimental evaluation on an assembly system. Section 6 contains some conclusions and future work.

### *1.1 Background and related work*

Reconfiguration processes are often largely human-driven, primarily based on the experience of system integrators. While there are methods and tools for solving specific problems, these methods and tools are not integrated into a general framework that can be used in a wide range of scenarios. Despite the development of self-organising intelligent systems that are able to perform logical adaptations with some level of autonomy, there is not yet a full solution to support the full reconfiguration process. A framework comprising information models and automated learning mechanisms to capture knowledge on reconfigurations is required to support engineers. This paper aims to develop such a framework.

Related work in this direction includes a capability-based methodology for adaptation planning, with the objective of developing tools for the rapid reconfiguration of production systems (Järvenpää et al., 2016). However, this methodology has only been applied in an academic research environment and not yet in an industrial context. Also, the computational processes associated with this ontological approach are not investigated. Research on the computational aspects, which are the focus of this paper, includes experience-based learning techniques using classification algorithms for the adaptation of plug and produce systems (Scrimieri et al., 2017). In plug and produce systems, identification and configuration of new devices is performed with minimal human intervention. One particular objective of automated learning in intelligent manufacturing systems is accelerating the production ramp-up phase, as investigated by Scrimieri et al. (2015). This paper further develops these automated learning techniques and integrates them in a multi-agent system. A reinforcement learning approach, guided by human experts, for the production ramp-up problem is presented by Doltsinis et al. (2018). The results of this last work indicate, as anticipated, that an exploration strategy guided by a human operator is more efficient than one that is purely algorithmic. Reinforcement learning has also been used for intelligent scheduling of reconfigurable flow lines (Yang and Xu, 2021). The design of the components, knowledge representation models and intelligent agents of an automated learning system is presented by Scrimieri et al. (2021), which this paper is based on.

Different manufacturing paradigms, together with physical and logical enablers, have been introduced to facilitate system changes (ElMaraghy, 2009). The paradigms include flexible and reconfigurable manufacturing (ElMaraghy, 2006; Koren and Shpitalni, 2010), holonic and agent-based manufacturing (see, for example, the ADACOR architecture by Barbosa et al. (2015)) and evolvable assembly systems (Chaplin et al., 2015). A review of reconfigurable manufacturing systems is conducted by Bortolini et al. (2018). An interesting research direction is the combination of machine and workforce reconfigurations (Hashemi-Petroodi et al., 2021). Reviews of agent-based manufacturing are presented by Shen et al. (2006) and Leitão et al. (2013). The contributions of holonic manufacturing to Industry 4.0 are examined by Derigent et al. (2020). Although agent-based and holonic manufacturing provide clear benefits, in particular in terms of flexibility and robustness, there are still barriers to their widespread industrial adoption. The fact that there is no guarantee on the operational performance of agent-based solutions prevents them from being applied to real-time control problems and is an obstacle to their acceptance by the management of companies.

## 2. MULTI-AGENT ARCHITECTURE

The proposed framework is based on a multi-agent architecture, which includes the following types of agents:

- ER Agent: The Experience Recognition agent captures the changes being made by engineers during reconfigurations, evaluates their impact on the system performance and stores them in an *experience base* (Section 3).
- LEARN Agent: The learning agent recommends adjustments based on the machine state received from the ADAPT agent and the experience base. This agent implements the machine learning algorithm described in Section 4.
- ADAPT Agent: When invoked by the HMI Agent, the adaptation agent captures the current machine state, sends it to the LEARN agent and queries this agent for recommended adjustments (Section 4).
- HMI Agent: The Human-Machine Interface agent allows the user (the engineer or system integrator making changes) to interact with the agent framework during a reconfiguration. On user request, this agent queries the ADAPT agent for a ranked list of adjustments and presents these to the user.

The agents are deployed on two levels: a *station level* and a *system level*. The agents of the station level are used for reconfiguring individual modules or subsystems, whereas the agents of the system level are used for reconfiguring the entire production system. A subsystem is a group of modules for assembly and checking a part. We study, in particular, production systems performing quality checks at various steps during the assembly process. Multiple agents of the same type can be deployed on different stations or at the system level. A station-level experience base is created for each station, containing only the local changes applied to that station, whereas a system level experience base is created for the whole system, containing both the local changes applied to each station and the global changes applied to the system. The agents of the station level work independently in each station.

## 3. EXPERIENCE RECOGNITION

Experience recognition (ER) is the process of recording the adjustments made during a system reconfiguration and evaluating their effect on the system performance. An adjustment is an atomic change (i.e. consisting of an individual operation) such as updating pick-and-place

points, pressure of grippers, speed of conveyor belt, cam angles or pallet geometry. While some adjustments can be made and recorded through software, others require physical operations and their details must be entered manually in the system. A KPI (Key Performance Indicator) must be defined in the framework in order to measure the performance of the production system and evaluate the impact of adjustments. A KPI is typically defined in terms of part quality of throughput. Sensor data from, e.g., quality inspection is used to calculate the KPI. When new sensor data is available, a *state event* is generated. The state events are stored in an *event base*, which can be queried by the ER agent.

When an adjustment is performed, the framework generates an *adjustment event*. These events contain the details of the adjustment, including the parameter and component affected and the associated value. Events are serialised into XML format and transmitted using a publish-subscribe mechanism. The ER agent subscribes to adjustments events in order to be informed when changes are made and to capture them. When the ER agent receives an adjustment event, it queries the event base for the state events generated by the machine before and after the adjustment. The ER agent then constructs a representation of the machine state before the adjustment by using the latest events generated before the adjustment. Similarly, the ER agent constructs a representation of the state of the machine after the adjustment. A machine state is represented by a list of machine parameters and variables being monitored. The ER agent then calculates the KPI on both machine states and uses this information to create an *experience instance* in the experience base.

Experience can also be captured in *adjustment sessions*. A "start adjustment session" event signals that a series of adjustments is beginning. When the ER agent receives this event, it captures the current machine state and calculates the KPI. Then, it records all the adjustment events being generated until it receives an "end adjustment session" event signalling the end of the series of adjustments. At that point, the ER agent captures the state of the machine again, calculates the KPI and uses this information to create an experience instance. Note that the effect of an adjustment is not immediately visible after the adjustment and it may be necessary to wait for some parts to be assembled before capturing the new machine state and calculating the resulting KPI.

## 4. AUTOMATED LEARNING

When invoked by the HMI agent during a reconfiguration, the ADAPT agent captures the current machine state, sends it to the LEARN agent and queries this agent for recommended adjustments. The LEARN agent generates a ranked list of adjustments that are applicable to the given machine state (also called adaptation context in the following, when referring to the machine state before a change). The ranked list is then presented to the user through the HMI Agent. The adjustments are found by searching the experience base and their rankings are calculated based on the similarity of the adaptation contexts and the effectiveness of the adjustments in the experience base. Similarity of adaptation contexts is measured using

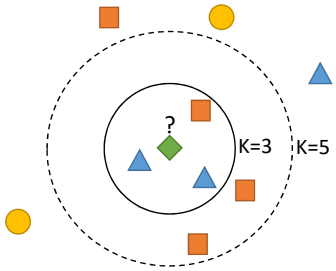


Fig. 1. kNN algorithm: The point to classify is indicated by '?'. The possible classes are: 'blue triangles', 'orange squares' and 'yellow circles'

a distance function calculated on multidimensional points representing machine states, constructed from sensor data and configuration parameters. The effectiveness of a past adjustment is measured by the KPI calculated on the machine state after the change.

The adjustments that were applied in the most similar contexts and that produced the best results are placed at the top of the list. The rationale is that an effective adjustment in a similar context is likely to produce a positive result. The first adjustment of the list is therefore the one recommended by the LEARN agent. The list may contain a large number of adjustments, depending on the accumulated experience and the number of possible adjustments. The adjustments with the lowest rankings could be filtered out. However, it may be useful to provide even the adjustments with the lowest rankings, so that the user can avoid them.

The process carried out to determine which adjustment to perform in one adaptation context can be framed as a *classification* problem. The objective of this problem is to learn how to identify the class of an instance based on a set of examples. The learning system receives in input some examples (training set) and produces a program (classifier) that is able to infer the class of instances that are not in the training set. In our reconfiguration problem, the instances to classify are the adaptation contexts in which the user queries the learning system, the classes are the possible adjustments and the training set is given by the experience instances in the experience base.

The reconfiguration problem is, however, more complex than a typical classification. The set of possible adjustments depends on the specific context. In general, there are preconditions associated with an adjustment, indicating if the adjustment is applicable or not. For example, adjustments related to devices not currently connected are clearly not applicable. In addition, we do not want only to identify what type of adjustment is the best, but also how to make it, that is, the values of associated parameters. For example, if the recommended adjustment is "increase the pressure of cylinders", we need to know what the new pressure should be. In the rest of this section, we will show how to determine both adjustment types and values.

### *4.1 Experience base search*

The search in the experience base is based on a variant of the k-nearest neighbour classification algorithm (kNN). Adaptation contexts are represented by points in a multi-

dimensional space and their similarity is captured by a distance function. In kNN, given a point $\mathbf{x}$ to classify, the $k$ nearest points to $\mathbf{x}$ are located and the class to assign to $\mathbf{x}$ is chosen among the neighbours' classes by applying a voting scheme. A simple voting scheme consists of assigning the most common class among the neighbours, as illustrated in Figure 1. If $k = 3$ (solid line circle) the most common class is 'blue triangles'. If $k = 5$ (dashed line circle) the most common class is 'orange squares'. The class of an adaptation context is the adjustment being applied in it.

Adaptation contexts can be defined by numerical or categorical attributes. The value of an attribute can be undefined if, for example, the module or sensor that produces it is not present or is faulty. Hence, we use a *heterogeneous Euclidean-overlap metric* (HEOM) (Wilson and Martinez, 1997):

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{n} w_i (d_i(\mathbf{x}, \mathbf{x}'))^2},$$

where:

- $\mathbf{x}$ and $\mathbf{x}'$ are two $n$-dimensional points,
- $w_i \in [0, 1]$ is the weight assigned to attribute $i$, and
- $d_i(\mathbf{x}, \mathbf{x}') \in [0, 1]$ is the distance between $\mathbf{x}$ and $\mathbf{x}'$ on attribute $i$, defined as:

$$d_i(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \mathbf{x}_i \text{ or } \mathbf{x}'_i \text{ is unknown,} \\ \text{overlap}(\mathbf{x}_i, \mathbf{x}'_i) & \text{attribute } i \text{ is nominal,} \\ \text{rndiff}_i(\mathbf{x}_i, \mathbf{x}'_i) & \text{otherwise.} \end{cases}$$

The function overlap gives 0 if its arguments are the same, otherwise 1. The function $\text{rndiff}_i$ (range normalised difference) is defined as:

$$\text{rndiff}_i(x, y) = \frac{|x - y|}{\max_i - \min_i},$$

where $\max_i$ and $\min_i$ are, respectively, the maximum and minimum values observed in the training set for attribute $i$.

In the generation of the rankings, we consider not only the similarity between adaptation contexts, but also the system performance obtained by the adjustments in the experience base. The performance is measured by the KPI of the machine state after the adjustment. An experience instance has the form $(\mathbf{x}, \mathsf{adj}, \mathbf{x}')$, where:

- $\mathsf{adj} = (t, v)$ is an adjustment of type $t$ and value $v$
- $\mathbf{x}$ is the machine state before $\mathsf{adj}$
- $\mathbf{x}'$ is the machine state after $\mathsf{adj}$

Let

- $E = (\mathbf{x}, \mathsf{adj}, \mathbf{x}')$ be an experience instance,
- $\mathbf{y}$ be an adaptation context,
- $d$ be a HEOM, and
- $f$ be a KPI.

The *similarity-performance function* $e_f^d$ (Scrimieri et al., 2015) is defined as:

$$e_f^d(E, \mathbf{y}) = \begin{cases} \dfrac{d(\mathbf{x}, \mathbf{y})}{f(\mathbf{x}')} & \text{if } f(\mathbf{x}') \neq 0 \\ \infty & \text{otherwise.} \end{cases}$$

The smaller the value of $e_f^d(E, \mathbf{y})$, the better $\mathsf{adj}$ is anticipated to be in $\mathbf{y}$.

**Procedure 1** rank-adjustments

**Input:** experience instances $E_1, \ldots, E_n$ $(k \leq n)$, adaptation context $\mathbf{y}$, HEOM $d$, KPI $f$
**Output:** ranked list of adjustments
  **for all** $1 \leq i \leq n$ **do**
    calculate $e_f^d(E, \mathbf{y})$
  **end for**
  sort $E_1, \ldots, E_n$ by $e_f^d(E, \mathbf{y})$
  $\{E_{\sigma_1}, \ldots, E_{\sigma_k}$ are now the $k$ nearest neighbours in sorted order$\}$
  **for all** $1 \leq i \leq k$ **do**
    calculate $w(E_{\sigma_i})$
  **end for**
  **let** $t_1, \ldots, t_m$ be the adjustment types in $E_{\sigma_1}, \ldots, E_{\sigma_k}$ $(m \leq k)$
  **for all** $1 \leq j \leq m$ **do**
    **let** $E'_i = (\mathbf{x}_i, \mathsf{adj}_i, \mathbf{x}'_i) \in \{E_{\sigma_1}, \ldots, E_{\sigma_k}\}$ be the experience instances such that $\mathsf{adj}_i = (t_j, v_i)$
    $\mathsf{w\_sum}_j \leftarrow \sum_i w(E'_i)$
    $\bar{v}_j \leftarrow \dfrac{\sum_i v_i w(E'_i)}{\mathsf{w\_sum}_j}$
  **end for**
  sort $(t_1, \bar{v}_1), \ldots, (t_m, \bar{v}_m)$ in descending order of $\mathsf{w\_sum}_j$

### *4.2 Voting and ranking*

Let $\mathbf{y}$ be an adaptation context to be classified, $d$ a distance function and $f$ a KPI. There are two cases to consider:

$k = 1$ The experience instances are sorted by $e_f^d(E, \mathbf{y})$, in ascending order, to produce a ranked list of adjustments for $\mathbf{y}$. The first experience instance in the list (the one with the smallest value of $e_f^d(E, \mathbf{y})$) contains the recommended adjustment type and value.

$k > 1$ This is described in Procedure 1. Let $E_1, \ldots, E_k$ be the $k$ experience instances having the smallest values of $e_f^d(E, \mathbf{y})$, in ascending order. A class must be selected among those of these $k$ nearest neighbours. To this end, a voting scheme is applied. Note that selecting the most common class among the neighbours may not be a good scheme, because this class is likely to be the most frequent in the training set. One method is to assign different weights to the neighbours, based on their respective distances from the point to be classified. In our solution, we use $e_f^d$ to calculate the weights. This way, both the distance of a neighbour and the impact of its associated adjustment on the performance of the system are considered. We define the weight $w(E_i)$ of $E_i$ as follows:

$$w(E_i) = \frac{e_f^d(E_k, \mathbf{y}) - e_f^d(E_i, \mathbf{y})}{e_f^d(E_k, \mathbf{y}) - e_f^d(E_1, \mathbf{y})}$$

The instances $E_1, \ldots, E_k$ are grouped by class. For each class, the weights of its instances are summed up. The recommended adjustment type is represented by the class having the largest sum of weights. For an adjustment type $t$, the instances among $E_1, \ldots, E_k$ having class $t$ are selected. If numerical, the adjustment value is calculated as the weighted mean of all the

adjustments values of these selected instances. The values $w(E_i)$ are used as weights. Further adjustments can be proposed similarly by considering the other classes in descending order of sum of weights.

## 5. EXPERIMENTAL EVALUATION

The proposed framework was evaluated on a production system for the assembly of injection pens. This system is made up of various pick-and-place and inspection modules. The pick-and-place modules are used for rotating and moving parts, and putting them together. The inspection modules are used for checking the presence and correct position of parts on pallets, and the quality of the finished product in terms of correct dimensions and assembly. Figure 2 shows a picture of the machine. If a finished product has passed all the checks, then it is accepted, otherwise it is rejected. Bad parts are not reworked because the process would not be cost-effective. Therefore, maximising the number of good parts is indispensable for this production system.

A conveyor system is used to transport pallets holding the parts from station to station. An injection pen consists of three parts: a body, a tank and a cap. When fed into the line, each pallet holds the parts for assembling 3 injection pens and, at the end of the process, holds the finished products. Each pick-and-place module operates 3 grippers, one for each pen to assemble. The assembly machine is cam-driven and consists of 7 modules performing the following operations:

(1) Check the presence of the parts on the pallet;
(2) Pick-and-place tanks into bodies;
(3) Check that tanks have been inserted correctly into bodies;
(4) Pick-and-place caps onto bodies;
(5) Check that caps have been inserted correctly onto bodies;
(6) Reverse pens;
(7) Check reversal.

The sequence of operations is linear.

The aim of the experiment was to evaluate how the self-learning framework can aid engineers in system reconfigurations. This was done by adapting the production system for the manufacture of a product variant with some minor differences compared to the original product, in terms of physical and geometrical characteristics. The adaptation involved making some mechanical changes, in particular changing some of the grippers. All the affected parameters in grippers and inspection devices had to be adjusted (e.g. position and dimensions of the parts, opening and closing angles of the grippers, pressure of the grippers). Although the product variant was not much different to the original one, such reconfiguration process could be laborious as assembly operations must be very precise and require repeatable positioning.

The experiment was organised as follows. First, the system was reconfigured by 4 engineers individually. During this phase the framework was used to capture the changes being made by the engineers and to build experience, but not to recommend changes. An experience base of adjustments was built by the ER agent at the system

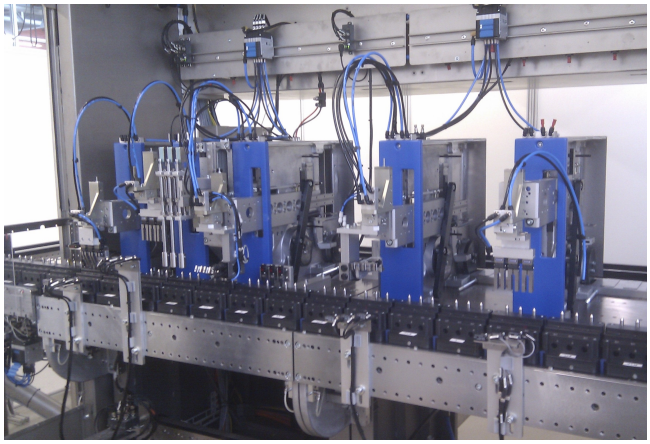

Fig. 2. Production system for the assembly of injection pens

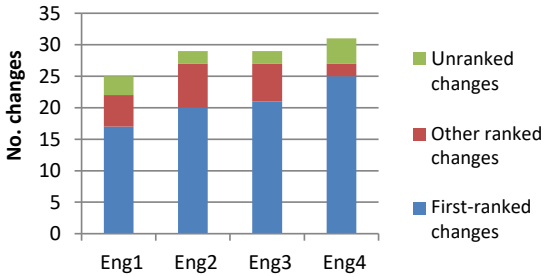


Fig. 3. Types of changes applied by the engineers of the group that used the framework

level. The original state of the machine was restored after each engineer completed the process. Then, the system was reconfigured by 4 other engineers individually using the framework with the experience base built in the previous phase. The engineers of the two groups had comparable experience and skills, and did not communicate during the experiment. The number of adjustments performed by the two groups for completing the process and the average levels of system performance reached at the end were compared.

The performance of the system was characterised in terms of quality of the finished product. The number of good parts out of the total number of parts assembled in a batch was used as KPI. The value of this KPI was updated every time the quality of an assembled part was checked. The number of parts produced by each engineer between two consecutive adjustments was fixed and it was equal to 6. The current KPI value was visible to the engineers to allow them to evaluate the result of their actions. The engineers were expected to obtain a target KPI value of 0.98 in order for the reconfiguration process to be considered complete. After 25 steps, the first group (that did not use the framework) reached an average KPI of 0.8, while the second group (that used the framework) reached an average KPI of 0.95. Further adjustments were required for all engineers except one to complete the process. To reach the target KPI, the 4 engineers of the group that did not use the framework made respectively 30, 33, 36 and 37 adjustments, whereas the 4 engineers of the group that used the framework made respectively 25, 29, 29 and 31 adjustments.

Figure 3 shows the number of times that the engineers of the second group applied the following types of changes: 1) the adjustment recommended by the ADAPT agent (i.e. the first-ranked adjustment), 2) a different adjustment ranked by the ADAPT agent, 3) any other adjustment not ranked by the ADAPT agent. The engineers followed the recommendations of the ADAPT agent most of the times. Only in a few cases did they choose another adjustment among those suggested, and in very few cases a different one. They decided not to apply any of the suggested adjustments when they thought that those changes were not useful or safe in that context, that they had already applied them (successfully or not), or that they could get better results with different changes.

The evaluation suggests that the use of the proposed framework to rank and recommend changes allows engineers to perform system reconfigurations in fewer steps and thus in a shorter period of time.

## 6. CONCLUSION

An agent-based framework with an automated method for learning from experience captured on an assembly production system has been presented. The experimental evaluation indicates that the framework can effectively support engineers in the selection of suitable adjustments during a reconfiguration. The framework can be applied to other production systems by specifying the attributes of the machine states, adjustment types and KPI.

The accuracy of the technique depends on the similarity-performance function, and thus on the distance and KPI functions. The use of a distance function is based on the assumption that the more similar two states are, the more likely an adjustment has the same effect on them. To refine the similarity-performance function, attributes should be weighted dynamically based on their relevance in the selection of a certain adjustment.

## FUNDING

This work was supported by the SURE Research Projects Fund of the University of Bradford and the European Commission [grant agreement n. 314762].

## REFERENCES

Barbosa, J., Leitão, P., Adam, E., and Trentesaux, D. (2015). Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution. *Computers in Industry*, 66, 99–111. doi: https://doi.org/10.1016/j.compind.2014.10.011.

Bortolini, M., Galizia, F.G., and Mora, C. (2018). Reconfigurable manufacturing systems: Literature review and research trend. *Journal of Manufacturing Systems*, 49, 93–106. doi:https://doi.org/10.1016/j.jmsy.2018.09.005.

Chaplin, J., Bakker, O., de Silva, L., Sanderson, D., Kelly, E., Logan, B., and Ratchev, S. (2015). Evolvable assembly systems: A distributed architecture for intelligent manufacturing. *IFAC-PapersOnLine*, 48(3), 2065–2070. 15th IFAC Symposium onInformation Control Problems in Manufacturing.

Derigent, W., Cardin, O., and Trentesaux, D. (2020). Industry 4.0: contributions of holonic manufacturing control architectures and future challenges. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-020-01532-x.

Doltsinis, S., Ferreira, P., and Lohse, N. (2018). A symbiotic human-machine learning approach for production ramp-up. *IEEE Transactions on Human-Machine Systems*, 48(3), 229–240.

ElMaraghy, H.A. (2006). Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, 17, 261–276.

ElMaraghy, H.A. (2009). Changing and evolving products and systems – models and enablers. In H.A. ElMaraghy (ed.), *Changeable and Reconfigurable Manufacturing Systems*, 25–45. Springer London, London.

Hashemi-Petroodi, S.E., Dolgui, A., Kovalev, S., Kovalyov, M.Y., and Thevenin, S. (2021). Workforce reconfiguration strategies in manufacturing systems: a state of the art. *International Journal of Production Research*, 59(22), 6721–6744. doi:10.1080/00207543.2020.1823028.

Järvenpää, E., Siltala, N., and Lanz, M. (2016). Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems. In *2016 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, 120–125.

Koren, Y. and Shpitalni, M. (2010). Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 29(4), 130–141. doi: https://doi.org/10.1016/j.jmsy.2011.01.001.

Leitão, P., Mařík, V., and Vrba, P. (2013). Past, present, and future of industrial agent applications. *IEEE Transactions on Industrial Informatics*, 9(4), 2360–2372.

Scrimieri, D., Oates, R., and Ratchev, S. (2015). Learning and reuse of engineering ramp-up strategies for modular assembly systems. *Journal of Intelligent Manufacturing*, 26, 1063–1076. doi:10.1007/s10845-013-0839-6.

Scrimieri, D., Afazov, S.M., and Ratchev, S.M. (2021). Design of a self-learning multi-agent framework for the adaptation of modular production systems. *The International Journal of Advanced Manufacturing Technology*, 115(5), 1745–1761. doi:10.1007/s00170-021-07028-z.

Scrimieri, D., Antzoulatos, N., Castro, E., and Ratchev, S.M. (2017). Automated experience-based learning for plug and produce assembly systems. *International Journal of Production Research*, 55(13), 3674–3685. doi: 10.1080/00207543.2016.1207817.

Shen, W., Hao, Q., Yoon, H.J., and Norrie, D.H. (2006). Applications of agent-based systems in intelligent manufacturing: An updated review. *Advanced Engineering Informatics*, 20(4), 415–431.

Wilson, D.R. and Martinez, T.R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6, 1–34.

Yang, S. and Xu, Z. (2021). Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing. *International Journal of Production Research*. doi:10.1080/00207543.2021.1943037.