Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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

Full length article

A framework for manufacturing system reconfiguration and optimisation utilising digital twins and modular artificial intelligence

Fan Mo [a,*], Hamood Ur Rehman [a,b], Fabio Marco Monetti [c], Jack C. Chaplin [a], David Sanderson [a], Atanas Popov [a], Antonio Maffei [c], Svetan Ratchev [a]

[a] *Institute for Advanced Manufacturing, University of Nottingham, Nottingham, NG8 1BB, Nottinghamshire, United Kingdom*
[b] *TQC Automation Ltd., Nottingham, NG3 2NJ, Nottinghamshire, United Kingdom*
[c] *Department of Production Engineering, KTH Royal Institute of Technology, Stockholm, 114 28, Södermanland and Uppland, Sweden*

A R T I C L E I N F O

Keywords:
Reconfigurable manufacturing system
Modular artificial intelligence
Digital twin
Process simulation
Knowledge graphs

A B S T R A C T

Digital twins and artificial intelligence have shown promise for improving the robustness, responsiveness, and productivity of industrial systems. However, traditional digital twin approaches are often only employed to augment single, static systems to optimise a particular process. This article presents a paradigm for combining digital twins and modular artificial intelligence algorithms to dynamically reconfigure manufacturing systems, including the layout, process parameters, and operation times of numerous assets to allow system decision-making in response to changing customer or market needs. A knowledge graph has been used as the enabler for this system-level decision-making. A simulation environment has been constructed to replicate the manufacturing process, with the example here of an industrial robotic manufacturing cell. The simulation environment is connected to a data pipeline and an application programming interface to assist the integration of multiple artificial intelligence methods. These methods are used to improve system decision-making and optimise the configuration of a manufacturing system to maximise user-selectable key performance indicators. In contrast to previous research, this framework incorporates artificial intelligence for decision-making and production line optimisation to provide a framework that can be used for a wide variety of manufacturing applications. The framework has been applied and validated in a real use case, with the automatic reconfiguration resulting in a process time improvement of approximately 10%.

1. Introduction

The application of industrial digital technologies and the increasing use of digitalisation in manufacturing processes have reduced manufacturing costs and are revolutionising the design, manufacture, usage, and maintenance of products, as well as the operations, procedures, and energy footprint of companies and supply networks [1]. Digitalising the value chain has made it possible for the end-user to establish a new, closer relationship with the producers [2], potentially directly involving the customer in the design process to achieve high levels of product customisation [3]. Mass customisation can require system configuration changes and the ability for systems to change configuration on the fly and handle product variations through the implementation of reconfigurable manufacturing systems [4]. Manufacturing systems reconfiguration is now evolving to embrace the Industry 4.0 paradigm, driven by the enablers described by Mabkhot et al. [5].

There is a need for a reconfigurable manufacturing framework that reacts to sudden changes and is able to quickly and seamlessly reconfigure and adapt, as highlighted by the COVID-19 pandemic. Standard manufacturing practices have been challenged by these extraordinary circumstances, and supply chains have been subject to extreme disruption [6,7], with the need to react to the surge in demand for healthcare products [8], sudden changes in production requirements, and with limitations in distribution and supply [9].

In this context, there are some enablers that will help move towards dynamic manufacturing reconfigurability. The first is the digital twin (DT), which is a virtual representation of a physical asset, dynamically exchanging data with the "physical twin". The difference between DTs and simulation models is the real-time bi-directional connectivity, enabling automated updating of the model to reflect the real process [10,11]. DTs are used to plan and optimise processes [12], and given the large volumes of data generated, smart data analytics are often used to analyse them and let managers take responsible, rapid decisions to regulate the process and improve productivity [13].

The second enabler is artificial intelligence, a broad domain with a variety of methods, including Machine Learning (ML) algorithms

* Corresponding author.
E-mail address: fan.mo@nottingham.ac.uk (F. Mo).

https://doi.org/10.1016/j.rcim.2022.102524
Received 20 September 2022; Received in revised form 16 December 2022; Accepted 30 December 2022
Available online 23 January 2023

that provide computationally valuable solutions for processing large volumes of data, e.g., processing and making predictions with sensor signals [14], natural language processing to extract useful manufacturing data from documents [15], and the construction and embedding of knowledge graphs of manufacturing resource data in cloud manufacturing environments [16].

To help relate information coming from data streams to physical entities, the concept of a knowledge graph (KG) can be used [17]. Utilising the graph-based query language, it is possible to infer additional knowledge from data gathered from a production line to improve the management of manufacturing processes [18]. The main applications of KGs are recommendation systems and exploratory search, but in the industrial sector, they are considered a potential enabler to building "semantic" digital twins [19]. Reconfigurable manufacturing dynamic KGs have been developed to tackle the challenge of making the best possible suggestion in different manufacturing scenarios [20]. Other KG architectures have also been used to enhance intelligent DTs by offering capabilities such as internal information references, knowledge completion, error detection, and semantic querying [21].

The reconfiguration of production processes in response to external changes is a difficult challenge because system reconfiguration requires varied and disparate information on system layout, process parameters, operation times of multiple assets, the sequence of the operations, material handling systems, and other relevant aspects in combination [12]. However, implementing digital twin and artificial intelligence solutions makes it realistic to achieve reconfigurable systems and customised demand-based production by allowing this information to be connected and related. The application of artificial intelligence and knowledge graph concepts to the digital twin-based streams of data would allow companies to analyse and better react to sudden changes in demand or disruptions.

This paper proposes a new framework to enable manufacturing system reconfiguration through the application of artificial intelligence algorithms such as genetic algorithms (GA), particle swarm optimisation (PSO), and simulated annealing algorithms (SA). This approach is designed such that newly developed algorithms can be rapidly applied. With this framework, the manufacturing data gathered from the physical environment – such as information about machines, workers, sensors, raw materials, work-pieces, products, and other physical entities, as well as the workshop production activities and workshop – can be integrated into the digital twin through the information fusion [22], updating the information in the virtual environment. The results of the iterative optimisation can then be transferred to the actual production plant. A simulation environment that simulates a manufacturing process with multiple industrial robots performing multiple tasks is developed. The real equipment data can also be used to improve the performance of the simulation environment, which achieves the function of the digital twin. A data pipeline with an application programming interface (API) that integrates artificial intelligence is built in the simulation environment to facilitate the bidirectional information exchange required of DTs. The data is stored in a data model utilising the knowledge graph concept. With the help of artificial intelligence and DT, the performance of the decision-making process and the reconfiguration process are improved.

Section 2 presents a brief overview of works related to this topic. Section 3 introduces the framework for manufacturing system reconfiguration and a description of the decision-making engine and the reconfiguration engine. Section 4 describes a validation use case with multiple robots, which applies the previously mentioned reconfiguration framework. Section 5 presents the conclusion and suggestions for future research. A shorter conference version of this paper appeared [23] at the FAIM 2022 conference. Our initial conference paper did not elaborate on the related work, the reconfiguration framework (Reconfiguration stages 1, 2, and 3) or the candidate devices selection process and the reconfiguration optimisation process. This manuscript addresses these issues, provides a more comprehensive introduction and explanation of the reconfiguration framework, and thoroughly demonstrates the results.

2. Background

This section introduces the relevant literature for manufacturing system reconfiguration based on artificial intelligence, digital twins, and knowledge graphs. It collects and highlights the main features described in the related literature and summarises the content.

2.1. Rationale

Rapid manufacturing reconfiguration capabilities are required because of the increasing market need for mass customisation and increased variability in supply and demand. This calls for a potentially sudden change of system configuration whilst maintaining full system effectiveness in the event of unpredictable customer demands, line failures, supply disruption, or a need for maintenance [4].

Reconfigurable Manufacturing Systems (RMSs) are considered a solution to the presented problems in next-generation systems. RMSs are dynamic and have the capacity to follow market changes and allow a higher spread in product variants and customisation of the product [4].

RMSs still have barriers and challenges to overcome, e.g., the inability to identify long-term requirements in terms of product and demand changes [24] - they are a primarily physical approach for streamlining reconfigurability rather than a holistic system. Few examples are already applied in industry, but the limitation in adoption is mostly due to the high investment needed [25], and more commonly, individual features and elements of the concept are applied to existing systems [25]. In order to create an RMS, the knowledge to guide the reconfiguration in a structured design methodology is needed, but support for reconfigurability in the production system design process is still lacking [26].

The advent of Industry 4.0 and the definition of the enabling technologies and elements [5] is significantly changing how RMS are implemented, with multi-agent systems (MAS) and the concept of cyber–physical systems (CPS) becoming key contributors to research in the reconfiguration of manufacturing environments [27–29].

However, with the ever-increasing complexity of manufactured products, research on manufacturing reconfiguration has encountered some challenges in managing its scalability and flexibility. For example, some research on manufacturing reconfiguration is only focused on one specific topic, such as scheduling [30], planning [31], or layout [32]. The utilisation of digital twins for achieving reconfigurability is limited because of a lack of knowledge in the simulation environment and compatibility between different simulation environments [33].

2.2. Digital twins

The volume of data generated by manufacturing systems (and particularly reconfigurable ones) is a challenge to handle but also represents an opportunity exploitable by utilising the DT concept, a virtual representation of a physical asset where both counterparts are autonomously connected to each other and are dynamically evolving through the whole system life cycle [34].

A digital twin comprises a physical resource and its virtual counterpart, the two being connected and exchanging data and information in real-time [35,36]. The concept raises important questions on the future of data gathering, interaction, mining, merging, and optimisation [37,38]. This approach is typically applied towards improving simulations and the optimisation of equipment [35,39], but the concept is evolving towards being an instrument used as a novel approach for the reconfiguration of automated systems [40], and prognostics on complex equipment [41].

DTs can be used to model the real world and exchange data back and forth with a manufacturing environment's various assets to affect the system's decisions and behaviour [42]. They can represent the base on which to build smart, fast and responsive manufacturing systems

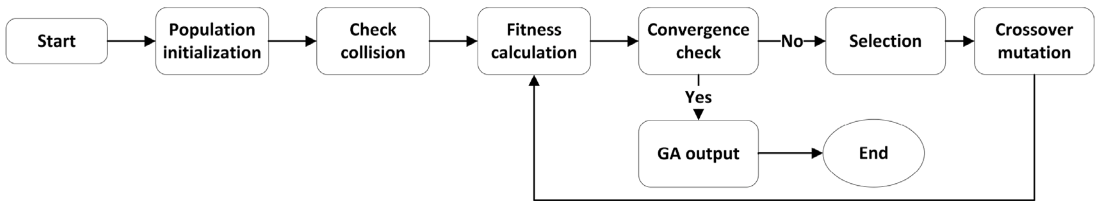


Fig. 1. Flowchart of GA.

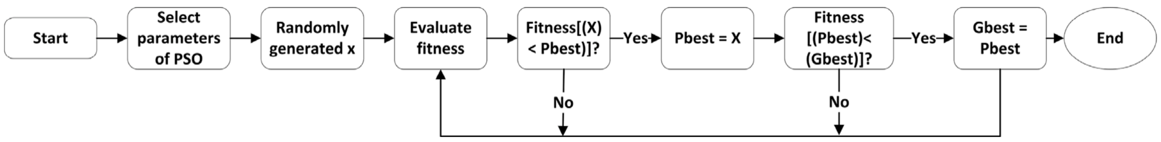


Fig. 2. Flowchart of PSO algorithm.

based on robotics assets [43,44], and can be used in parallel with artificial intelligence approaches to reconfiguring human–robot collaborative assembly lines [45,46]. Novel digital twin-driven approaches that achieve rapid reconfiguration can be developed for automated manufacturing systems that use open-architecture machine tools to achieve the desired reconfiguration [40] by integrating customisable process planning.

The DT approach is used in the present study to simulate manufacturing processes involving numerous industrial robots that simultaneously perform different tasks. Embedded in the DT is an artificial intelligence-driven data pipeline that allows the asset model, and consequently the real one, to improve the decision-making capabilities of the system and the related performances.

Despite the definition of a digital twin requiring autonomous bidirectional information exchange between the physical and digital aspects, changing a manufacturing set-up may still require manual data entry new equipment is deployed, often requiring re-measuring with laser measurements, 3D point cloud scans or similar systems to accurately position items in the virtual space [47]. This manual data entry approach is prone to human errors and to the risk of variation between virtual and physical environments. The need for an automatic and consistent pairing approach to combine virtual and physical twins was recently addressed with an objective evaluation of the connection between physical and digital counterparts [48].

2.3. Intelligent systems and intelligent decision making

To meet fluctuating customer and market demands, production systems must be as responsive as possible, and intelligent system architectures such as multi-agent systems allow for “on the fly” reconfiguration of manufacturing system control, exploiting asset flexibility and control to change the behaviour of the manufacturing system [49]. These capabilities can be enhanced through an ontology-based architecture, which increases the system’s intelligence by allowing it to reason about concepts, and has been applied to increase productivity while reducing energy consumption [50].

Human–machine collaboration is an increasingly important concept in flexible and reconfigurable manufacturing systems, but considerations of safety in such environments are needed [32]. Another related topic where decision-making based on data is important is the self-repair ability of smart systems; an artificial intelligence approach selects the best strategy, based on product and module swapping, operation rescheduling, and reconfiguration, to significantly reduce the capacity loss [51]. The reconfigurability of a manufacturing system can be achieved by applying hybrid agent-based and discrete-event simulation modelling techniques [52], bringing benefits such as decentralised control and collaborative decision-making, enabling a system to react to system variability and faults. Intelligent systems have been used to evaluate the priorities of production and redistribute tasks and jobs between agents with a negotiation approach; these distributed artificial intelligence systems are proving competitive in dynamic environments [53].

These intelligent systems approach offer frameworks for integrating decision-making into the manufacturing process. Many different decision-making methods can be used for a variety of goals, including achieving the reconfigurability of the manufacturing system. Here we focus on three algorithms that are later used as validation for the reconfiguration framework. They are the Genetic Algorithm (GA), the Particle Swarm Optimisation Algorithm (PSO) and the Simulated annealing algorithm (SA).

2.3.1. Genetic algorithm

A genetic algorithm (GA) is a search heuristic that is inspired by Charles Darwin’s theory of natural selection. These algorithms reflect the process of natural selection, where the fittest algorithms are selected for reproduction and mutation to produce offspring of the next generation [54]. The generic flowchart of these genetic algorithms is listed in Fig. 1.

A challenge of using GA is when the number of elements exposed to mutation is large, there is often an exponential increase in search space size. The complexity of evaluating the fitness of solutions is also a problem, making it difficult to apply to complex problems with large search spaces [55].

Genetic algorithms have been applied frequently in the manufacturing industry. Zhou et al. [56] propose a cloud-forging resource service optimisation strategy based on a genetic algorithm. Bensmaine et al. [57] proposed a new technique based on a genetic algorithm to solve the problem of machine selection in reconfigurable manufacturing systems design.

2.3.2. Particle swarm optimisation algorithm

Kennedy and Eberhart suggested PSO in 1995, inspired by schools of fish or flocks of birds moving in a group. While a bird may seemingly fly about looking for food at random, all of the birds in the flock will share information indirectly and assist the flock as a whole [58].

PSO’s answer is often close to the best global solution [59]. The generic flowchart of the typical PSO algorithm is shown in Fig. 2. Each particle has its own velocity and position, which are randomly initialized at the start. Each particle maintains a record of its positions $Pbest$ (the local best position) and $Gbest$ (the global best position among all the particles). The main challenge of the PSO algorithm is that it is easy to fall into local optimum in high-dimensional space and has a low convergence rate in the iterative process.

PSO algorithms have been applied successfully as optimisation tools in the manufacturing industry. PSO integrated with Memetic Algorithms (called Modified Memetic Particle Swarm optimisation Algorithm or MMPSO) is applied to yield initial feasible solutions for

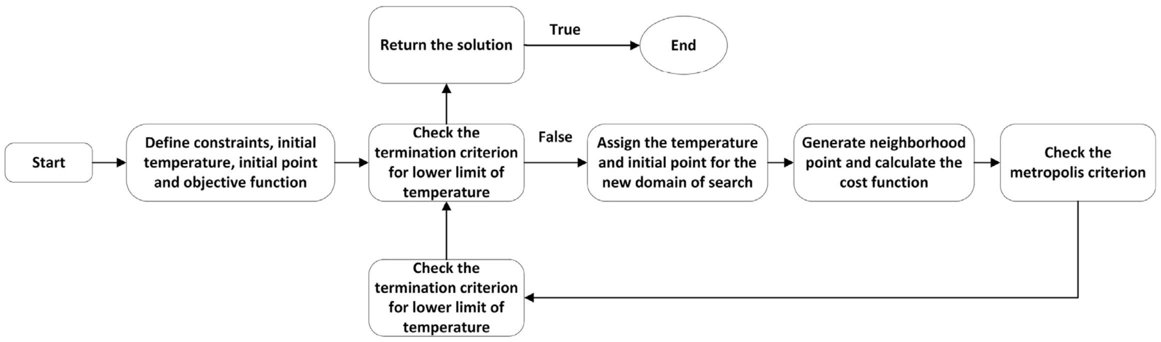

Return the solution
True
End
Start
Define constraints, initial temperature, initial point and objective function
Check the termination criterion for lower limit of temperature
False
Assign the temperature and initial point for the new domain of search
Generate neighborhood point and calculate the cost function
Check the metropolis criterion
Check the termination criterion for lower limit of temperature

Fig. 3. Flowchart of SA.

the scheduling of multi-load automated ground vehicles (AGVs) for minimum travel and waiting time in flexible manufacturing systems (FMSs) [60]. Han et al. [61] propose an effective hybrid particle swarm optimisation algorithm to solve the deadlock-free scheduling problem of FMSs that are characterised by lot sizes, resource capacities, and routing flexibility.

2.3.3. Simulated annealing algorithm

The simulated annealing (SA) algorithm is one of the most preferred heuristic methods for solving the optimisation problems [62]. The annealing process refers to cooling metals gradually after being exposed to high heat and defines the ideal molecular configurations of metal particles where the mass's potential energy is reduced. These SA algorithms typically adopt an iterative movement based on a changeable temperature parameter that mimics the metals' annealing process [63]. The generic flowchart is listed in Fig. 3.

The challenge of using SA is that if the cooling process is slow, then the convergence speed is slow, and the algorithm performance is highly related to the initial value. Besides, the parameters are sensitive. If the cooling process is too fast, the optimal global solution may not be obtained [64].

SA has also been applied successfully in the manufacturing industry. Wang et al. [65] formulate a model solving both inter-cell and intra-cell facility layout problems for cellular manufacturing systems. While these methods can give an optimal solution to small-scale problems, they are often inefficient when applied to larger-scale problems. Koren et al. [66] explore how to model reconfigurable manufacturing activities from an optimisation perspective and how to develop and select appropriate non-conventional optimisation techniques for reconfigurable process planning with the help of a simulated annealing algorithm.

2.4. Knowledge graph storage

In 2012, Google introduced its Knowledge Graph(KG) project and used it to improve query result relevancy and users' search experience. Due to the increasing amount of internet resources and the release of linked open data (LOD) projects, many knowledge graphs have been constructed. KG consist of two layers: the schema layer which represents the abstract concepts, and the entity layer which represents the specific entities that are instantiated from ontological concepts from the schema layer. KG can be applied to many applications, such as question-answering systems [67,68], recommendation systems [69–71], and information retrieval [72–74].

There are two main types of storage for knowledge graphs. One is Resource Description Framework (RDF)-based storage; the other is graph database-based storage. RDF approaches allow for ease of data distribution and sharing, while graph databases focus on efficient graph queries and search. RDF approaches also store data in triples and does not contain attribute information, but graph databases generally use attribute graphs as the basic representation, so entities and relationships can contain attributes, which means it is easier to express realistic business scenarios [75]. Among them, the Neo4j system is a widely used graph database [76].

Tang et al. [77] compare the storage and query performance between the general storage format of RDF and graph database based on a constructed power equipment management knowledge graph. Ma et al. [78] propose a method of knowledge graph-based manufacturing capability service optimal selection for ICRs. Different families of data management methods of RDF graphs and property graphs have been separately developed in each community for over a decade, which hinders the interoperability in managing large knowledge graph data.

3. Methodology - Reconfiguration framework

Motivated by the literature review and the current limitations of the techniques therein, including challenges of manufacturing reconfiguration in managing its scalability and flexibility and difficulties in collecting data for the manufacturing reconfiguration, this section proposes a framework for managing manufacturing system reconfiguration optimisation which provides a general methodology to achieve reconfiguration in different cases and utilises the benefit of digital twins to increase the performance and the accuracy of the reconfiguration. Furthermore, this framework enables the system to select the most suitable modular AI optimisation method based on the required key performance indicator(s) (KPIs).

This reconfiguration framework covers two levels of the automation pyramid [79]: the manufacturing execution system level and the controller level. The manufacturing execution system level interacts in real-time with the controller level to deal with changing demands on the manufacturing system and makes decisions. The controller level will receive the information from the manufacturing execution level and send the related control signals to the physical asset.

The reconfiguration optimisation process consists of three stages. Reconfiguration stage 1 is capability matching and resource selection, a matching process between the current system configuration and the product requirements to determine if the current system is capable of meeting the needs and, if not, what available assets could be used instead. All the related information for this selection process must be made available to the algorithm; in our case, a knowledge graph is used to store this data [80]. Reconfiguration stage 2 is the trade-off and optimisation process, which takes the selected assets from stage 1 and determines their best locations and parameters. Reconfiguration stage 3 is the configuration update required to instantiate the reconfiguration, which can include Programmable Logic Controller (PLC) and robot controller code updates. Below is a detailed description of the framework components, and Fig. 4 describes the flowchart of our proposed framework:

A **Experience data bank:** The experience data bank is a database for storing data both from the current system configuration

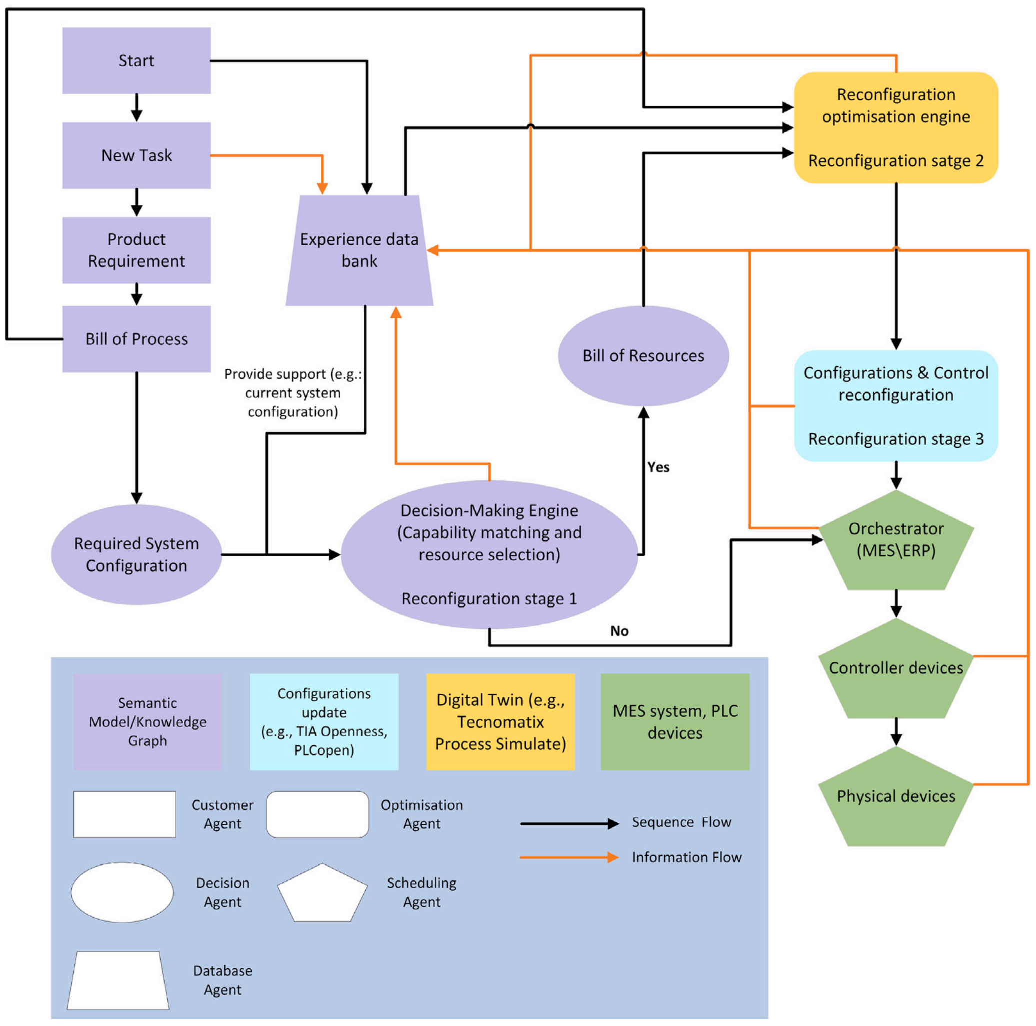

Start
New Task
Product Requirement
Bill of Process
Experience data bank
Provide support (e.g.: current system configuration)
Reconfiguration optimisation engine
Reconfiguration satge 2
Bill of Resources
Configurations & Control reconfiguration
Reconfiguration stage 3
Yes
Orchestrator (MES\ERP)
Required System Configuration
Decision-Making Engine (Capability matching and resource selection)
Reconfiguration stage 1
No
Controller devices
Physical devices
Semantic Model/Knowledge Graph
Configurations update (e.g., TIA Openness, PLCopen)
Digital Twin (e.g., Tecnomatix Process Simulate)
MES system, PLC devices
Customer Agent
Optimisation Agent
Sequence Flow
Information Flow
Decision Agent
Scheduling Agent
Database Agent

Fig. 4. An overview of the system reconfiguration framework.

and from previous system configurations. The current system configuration consists of information about the current production line, including process information, equipment information, layout information, machine runtime condition, and value-chain information. Industry-standard models for these are used to represent the current system configuration information [81]. The assets in the production line will be digitalised and represented by using the digital model [82,83], and a knowledge graph is flexible enough to store the information and the relationships between them. The experience data bank also contains the digital twin for the current configuration, digital twins and simulation results from the previous system configurations, optimisation algorithms, and libraries of PLC code for updating the control programs. The offline data of the digital twin, the simulation model, the interfaces to access data in the live physical system, and other information such as asset information, operation information, and signal information are all stored in the database for future use. What data is stored will depend on the simulation tool being used.

B **Product requirements:** The product requirements specify the operations the system must perform and include the bill of process, which specifies the processes and their order to create the product. The source of the bill of process depends on the use case and could be the customer specifying the bill of process themselves, the manufacturer creating the bill of process in response to customer requests, or utilising a semantic model and matching algorithm to generate these automatically. We assume for the purposes of this framework, that the bill of process is provided to us by an existing manufacturing execution or product life-cycle management system. To ensure the generalisation of our proposed framework, different manufacturing process models can be utilised to represent the bill of process, based on the customer requirement, such as AutomationML [84], process-oriented information model (PIM) [85], Petri Nets [86], or Core Product Model (CPM) [87].

C **Decision-making engine (Reconfiguration stage 1):** The decision-making engine decides the level of reconfiguration required. It performs a capability matching process and the candidate resource selection process. The needs of the bill of process will be compared with the current system configuration to determine if any reconfiguration is required at all. If yes, the system will recommend adding or updating assets to meet the requirements, generating the bill of resources. The incurred extra cost will be updated in the experience data bank. The decision-making engine will also provide further reconfiguration suggestions to the reconfiguration optimisation engine. In the decision-making engine, other decision criteria such as cost, operation time, energy consumption, and capacity are not yet considered and will be addressed in later stages of reconfiguration.

D **Reconfiguration optimisation engine (Reconfiguration stage 2):** The reconfiguration optimisation engine is introduced to perform the second stage of reconfiguration, which is optimisation

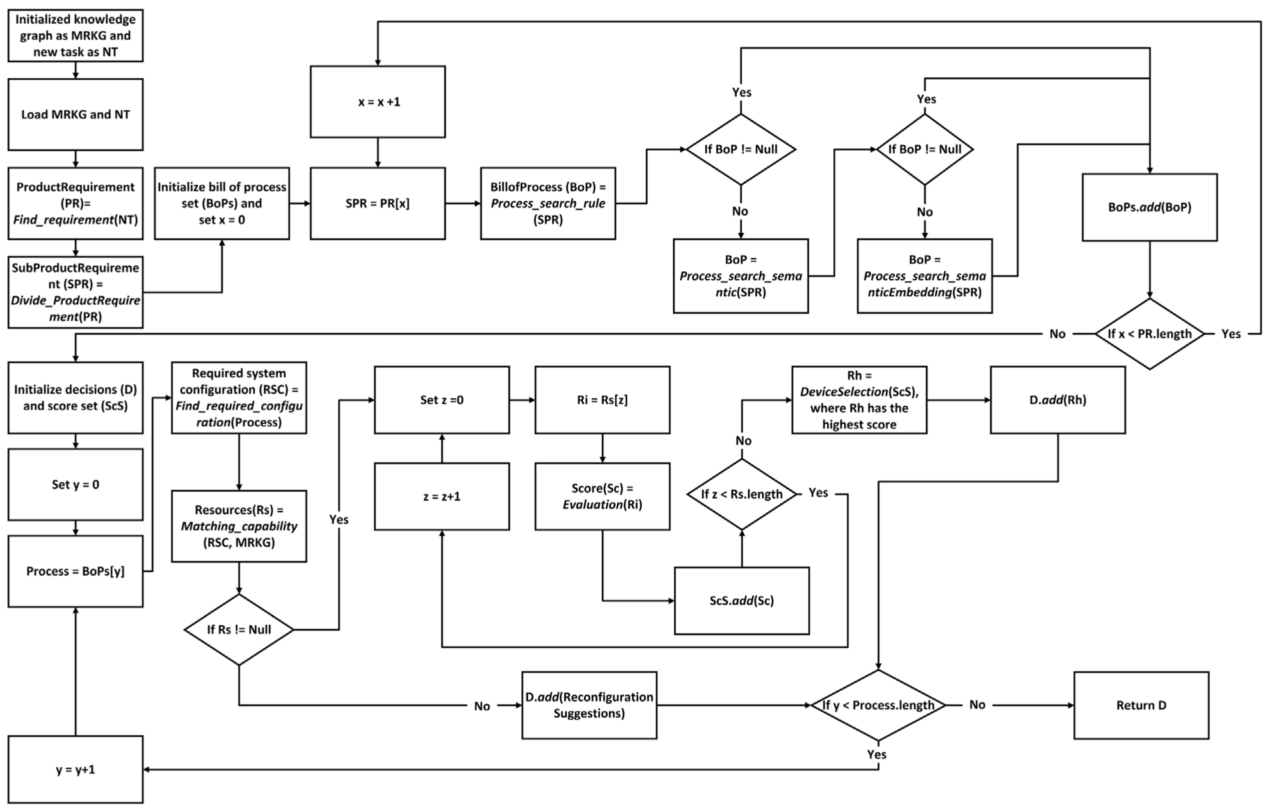

Initialized knowledge graph as MRKG and new task as NT
Load MRKG and NT
ProductRequirement (PR)= Find_requirement(NT)
SubProductRequirement (SPR) = Divide_ProductRequirement(PR)
Initialize bill of process set (BoPs) and set x = 0
x = x +1
SPR = PR[x]
BillofProcess (BoP) = Process_search_rule (SPR)
If BoP != Null
Yes
No
BoP = Process_search_semantic(SPR)
BoP = Process_search_semanticEmbedding(SPR)
BoPs.add(BoP)
If x < PR.length
Initialize decisions (D) and score set (ScS)
Set y = 0
Process = BoPs[y]
Required system configuration (RSC) = Find_required_configuration(Process)
Resources(Rs) = Matching_capability (RSC, MRKG)
If Rs != Null
Set z =0
z = z+1
Ri = Rs[z]
Score(Sc) = Evaluation(Ri)
ScS.add(Sc)
If z < Rs.length
Rh = DeviceSelection(ScS), where Rh has the highest score
D.add(Rh)
D.add(Reconfiguration Suggestions)
If y < Process.length
Return D
y = y+1

Fig. 5. A flow chart of the decision-making process for reconfiguration.

based on different KPIs set by the user or customer. Additionally, the optimised information and the reconfiguration use case will also be stored in the data bank for future reference.

E **Configuration & control reconfiguration (Reconfiguration stage 3):** Once the optimisation in the simulation environment is finished, as well as a physical layout, the reconfiguration optimisation engine will recommend a corresponding software and control reconfiguration based on the optimised setup. For example, if the position and operation sequence of the robots have changed, the PLC and robot control code should also be updated, ideally automatically. The optimised solution and the updated configuration information will first be tested with the simulation environment and then sent to the orchestrator.

F **Orchestrator:** The orchestrator is a system of software and hardware elements that allows industrial organisations to control industrial processes. Depending on context and application, it can correspond to the manufacturing execution system (MES), enterprise resource planning system (ERP), a human–machine interface, or a combination. It will receive information from the decision-making engine and the reconfiguration optimisation engine. Then it will send the corresponding reconfiguration information to the equipment controllers and then to the real equipment.

The following subsection will discuss the decision-making engine (reconfiguration stage 1) and reconfiguration optimisation engine (reconfiguration stage 2) in detail.

3.1. Reconfiguration stage 1 - Decision-making engine

The execution of the decision-making engine utilises a Manufacturing Reconfiguration Knowledge Graph (MRKG) established in the experience data bank step (step A) of the framework and will determine the assets required to enable the creation of the product (or the decision that creation of the product is not possible). At this stage, the product requirements in the form of the bill of process have been decomposed into sub-product requirements. The core procedures of product requirement decomposition include rule-based methods, semantic-based methods, and semantic-embedding-based methods, and they are performed in the product requirements step (step B) of the framework. The decision engine will begin by searching the MRKG to discover any existing bill of resources that could be used, either for the whole product or sub-product requirements.

If no suitable existing bill of resources can be found (or gaps exist), then the MRKG is used to conduct capability matching between the current system configuration and the required processes to determine the bill of resources. For the capability matching, the required system configuration node will query from the current system configuration node to check if there are capabilities that can fulfil the product requirement. If there are more assets, which fulfil the capability requirement at the same time, the next process is to select the most suitable candidate devices. Algorithm 1 and Fig. 5 show detailed information about the decision-making process with the help of the evaluation method and the knowledge graph. A graph neural network-based embedding algorithm [88] is proposed based on the established MRKG, to achieve the decision-making process in a methodical way. The final decisions can be:

1. The current system is able to produce the product. In this case, either one combination of the assets or more than one combination of the assets can be selected during the capability matching process. If more than one combination of the assets can be applied to produce the product, the selection of the most suitable assets will be executed.
2. The current system is not able to produce the product. In this case, the decisions will be adding or updating assets to match the product requirement. Information about other potential assets to be added to the MRKG will be suggested to the user.

Algorithm 1: Stage 1 - Decision-making process for reconfiguration

```
Input: Initialized knowledge graph as $MRKG$, New Task as $NT$
Output: Decisions at the reconfiguration stage 1 as $D$
1: Load $MRKG$ and $NT$
2: ProductRequirement ($PR$)=$Find\_requirement(NT)$
3: SubProductRequirement ($SPR$)=$Divide\_ProductRequirement(PR)$
4: Initialize bill of process set ($BoPs$)
5: for each $SPR \in PR$ do
6:     BillofProcess ($BoP$)=$Process\_search\_rule(SPR)$
7:     if $BoP$ ! = $Null$ then
8:         $BoPs.add(BoP)$
9:         Continue
10:    else $BoP = Process\_search\_semantic(SPR)$
11:        if $BoP$ ! = $Null$ then
12:            $BoPs.add(BoP)$
13:            Continue
14:        else $BoP = Process\_search\_semanticEmbedding(SPR)$
15:            $BoPs.add(BoP)$
16:            Continue
17:        end if
18:    end if
19: end for
20: Initialize decisions ($D$) and score set ($ScS$)
21: for each $Process \in BoPs$ do
22:     Required system configuration ($RSC$) =
        $Find\_required\_configuration(Process)$
23:     Resources ($Rs$) = $Matching\_capability(RSC, MRKG)$
24:     if $Rs$ ! = $Null$ then
25:         for each $R_i \in R_s$ do
26:             Score ($Sc$) = $Evaluation(R_i)$
27:             $ScS.add(Sc)$
28:         end for
29:         $R_h = DeviceSelection(ScS)$, ▷ where $R_h$ has the highest score
30:         $D.add(R_h)$
31:         Continue
32:     else $D.add(ReconfigurationSuggestions)$
33:     end if
34: end for
35: return $D$
```

Fig. 6 shows the resource selection process in the decision-making engine. The assumption is that there are in total j process $(p_1, p_2, \ldots, p_j)$ which are potential candidates to be used to perform a process. Below is how to describe the number of candidate assets D for each process. For process m (m is the numbering of the process, $m = [1, j] \cap \mathbb{Z}$) where $\mathbb{Z}$ is the integer set, there are in total k_m candidate resources. C_m is the candidate device sets at the process m which can be described in (1). In this way, the potential candidate assets for each process can be expressed.

$$C_m = [D_{m_1}, D_{m_2}, \ldots, D_{mk_m}], (\text{where } m = [1, j] \cap \mathbb{Z}) \tag{1}$$

The weights of each candidate device at every process can be calculated in (2). w_{imd} describes the weights of the candidate device sets with the numbering d about evaluator i at process m. h is the total number of evaluators. Evaluators are different in different candidate device sets. For example, if the candidate devices are robots, then features to differentiate them include gripper type, capability model, reachability, and repeatability. These features are used in our proposed approach as the evaluators. The evaluators can be described in the form of the vectors [89,90] or defined in the form of numbers with the criteria designed by the engineer according to the experience [91]. $\sigma_{md\_eva\_i}$ is the deviation between evaluator set i and the corresponding features of the device with the numbering d at process m.

$$w_{imd} = \frac{\sigma_{md\_eva\_i}}{\sum_{n=1}^{k_m} \sigma_{mn\_eva\_i}},$$
$$(\text{where } d = [1, k_m] \cap \mathbb{Z}, k_m = [k_1, k_j] \cap \mathbb{Z}, m = [1, j] \cap \mathbb{Z}, i = [1, h] \cap \mathbb{Z}) \tag{2}$$

If the candidate device matches more closely to the evaluators, the deviation will be smaller, which means the weights of this candidate device should also be smaller. And according to the Algorithm 1, the most suitable candidate device will have the highest score, so the reciprocal of w_{imd} will be used for further evaluation. The updated weights can be described below in (3).

$$W_{imd} = \frac{1}{w_{imd}} = \frac{\sum_{n=1}^{k_m} \sigma_{mn\_eva\_i}}{\sigma_{md\_eva\_i}},$$
$$(\text{where } d = [1, k_m] \cap \mathbb{Z}, k_m = [k_1, k_j] \cap \mathbb{Z}, m = [1, j] \cap \mathbb{Z}, i = [1, h] \cap \mathbb{Z}) \tag{3}$$

The weights sets W_{md} of the device with the numbering d at process m can be described in (4).

$$W_{md} = [W_{1md}, \ldots, W_{hmd}], \tag{4}$$

Take process p_1 as an example, there are in total k_1 candidate sets and the weights of the candidate sets will be calculated. W_{111} describes the weights of evaluator 1 of the candidate device D_{11} at process 1. $\sigma_{11\_eva\_1}$ is the deviation between evaluator set 1 and the device accuracy information of the above-mentioned device.

After the weights of different devices at each process are calculated, the scores S_d^i of the candidate devices under the evaluator U_i should be calculated, where $d = [1, k_m] \cap \mathbb{Z}$ and $i = [1, h] \cap \mathbb{Z}$. This score aims to show the matching ability of the candidate devices under a single evaluator. The score can be defined in many different ways, as long as it shows the matching ability of the candidate devices.

Taking the consistent fuzzy matrix as an example, the scores can be calculated in the following way [92]. Considering that there is a question of the selection of the most suitable devices from k_m candidate devices under h evaluators. Under this condition, $h \cap \mathbb{Z}$ of single-evaluator fuzzy priority relations can be built.

$$B_i = (b_{ef}^i)_{k_m \times k_m}, (\text{where } k_m = [k_1, k_j] \cap \mathbb{Z}, i = [1, h] \cap \mathbb{Z}) \tag{5}$$

where b_{ef}^i is called the coefficient of the preferred relationship of device D_{me} to device D_{mf} under the evaluator U_i. The value of b_{ef}^i is listed in (6).

$$b_{ef}^i = \begin{cases} 0, & \text{if } D_{me} \text{ is worse than } D_{mf} \text{ under the factor } U_i \\ 0.5, & \text{if } D_{me} \text{ is equal to } D_{mf} \text{ under the factor } U_i \\ 1, & \text{if } D_{me} \text{ is better than } D_{mf} \text{ under the factor } U_i \end{cases} \tag{6}$$

Then $B_i (i = [1, h] \cap \mathbb{Z})$ will be converted to the consistent fuzzy matrix as listed in (7) below.

$$R_i = (r_{ef}^i)_{k_m \times k_m} \tag{7}$$

where

$$r_{ef}^i = \frac{r_e^i - r_f^i}{2k_m} + 0.5; r_e^i = \sum_{l=1}^{k_m} b_{el}^i \tag{8}$$

Because for $\forall l = 1, 2, \ldots, k_m$,

$$r_{el}^i - r_{fl}^i + 0.5 = \frac{r_e^i - r_l^i}{2k_m} + 0.5 - (\frac{r_f^i - r_l^i}{2k_m} + 0.5) + 0.5 = \frac{r_e^i - r_f^i}{2k_m} + 0.5 = r_{ef}^i \tag{9}$$

So the $R_i (i = [1, h] \cap \mathbb{Z})$ is the consistent fuzzy matrix [91]. The score s_d^i of the candidate device D_{mi} under evaluator U_i can be calculated in (10):

$$s_d^i = \frac{\bar{s_d}}{\sum_{l=1}^{k_m} \bar{s_l}}, (\text{where } \bar{s_d} = (\prod_{l=1}^{k_m} r_{el}^i)^{\frac{1}{k_m}} \text{ and } d = [1, k_m] \cap \mathbb{Z}) \tag{10}$$

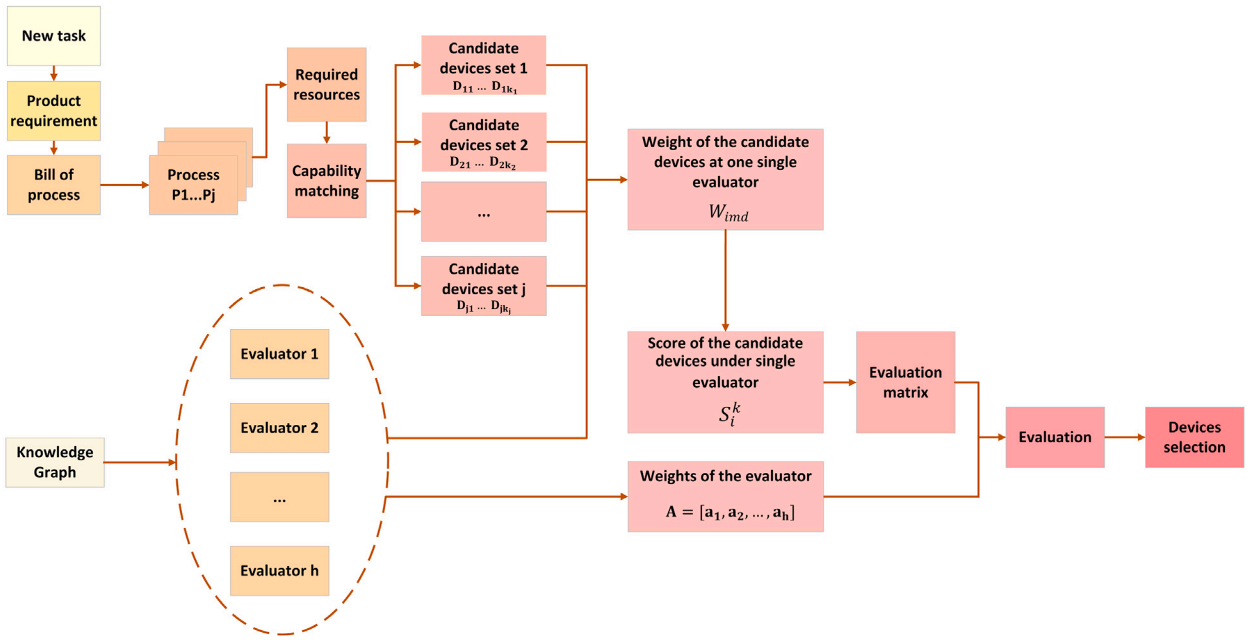

New task
Product requirement
Bill of process
Process P1...Pj
Required resources
Capability matching
Candidate devices set 1 $D_{11} \ldots D_{1k_1}$
Candidate devices set 2 $D_{21} \ldots D_{2k_2}$
...
Candidate devices set j $D_{j1} \ldots D_{jk_j}$
Weight of the candidate devices at one single evaluator W_{imd}
Score of the candidate devices under single evaluator S_i^k
Evaluation matrix
Evaluation
Devices selection
Knowledge Graph
Evaluator 1
Evaluator 2
...
Evaluator h
Weights of the evaluator $A = [a_1, a_2, \ldots, a_h]$

Fig. 6. Capability matching and device selection process.

After that, the fuzzy set of the weights of each evaluator factor should be calculated [93]. This information comes from the rich relatedness information of the generated knowledge graph and also from the engineer's experience. A is the weight set of each evaluator in the fuzzy evaluation.

$$A = [a_1, a_2, \ldots, a_h] \tag{11}$$

Then the evaluation matrix should be calculated. There are many methods to define the evaluation matrix, such as using the weights sets W_{md}, consistent fuzzy matrix [91], the definition of the membership degree function based on the experts' knowledge [94], use of historical data, and the help of the knowledge graph [95]. The fuzzy matrix, in general, can be described in :

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1g} \\ r_{21} & r_{22} & \cdots & r_{2g} \\ r_{31} & r_{32} & \cdots & r_{3g} \\ \vdots & \vdots & \ddots & \vdots \\ r_{h1} & r_{h2} & \cdots & r_{hg} \end{bmatrix} \tag{12}$$

Where h is the total number of evaluator sets, and g is the total number of evaluation criteria. In terms of utilising a consistent fuzzy matrix to select the most suitable devices from k_m candidate devices under h evaluators, the evaluation criteria can be understood as the candidate devices, then the fuzzy matrix can be described in (13):

$$R = \begin{bmatrix} S_1^1 & S_2^1 & \cdots & S_{k_m}^1 \\ S_1^2 & S_2^2 & \cdots & S_{k_m}^2 \\ S_1^3 & S_2^3 & \cdots & S_{k_m}^3 \\ \vdots & \vdots & \ddots & \vdots \\ S_1^h & S_2^h & \cdots & S_{k_m}^h \end{bmatrix} \tag{13}$$

The total score of each candidate device S_d can be calculated in (14):

$$S_d = \sum_{k=1}^{h} A_k \cdot s_d^i,$$
$$\text{(where } d = [1, k_m] \cap \mathbb{Z}, \text{and } A \text{ is the evaluator weights set)} \tag{14}$$

The ranking of the candidate devices in order of $S_d (d = 1, 2, \ldots, k_m)$ from largest to smallest can be obtained by combining the influence of h factors. The most suitable device will have the highest score according to Algorithm 1. If no suitable device is available, the system will recommend the addition of new devices to the user.

Table 1
Membership degree defined by the engineer.

	Device 1	Device 2	Device 3
Evaluator 1	0.4	0.6	0.7
Evaluator 2	0.3	0.7	0.2

As an example, assume that there is a situation where there are three candidate devices (device 1, device 2 and device 3) and two evaluators in the use case. The membership degree for each evaluator can be defined by the engineer or be defined from the knowledge graph with the help of weights sets W_{md}. In this use case, the membership degree is shown in Table 1.

Then the fuzzy matrix can be defined in (15).

$$R = \begin{bmatrix} 0.4 & 0.6 & 0.7 \\ 0.3 & 0.7 & 0.2 \end{bmatrix} \tag{15}$$

The weight sets of each evaluator are assumed and listed in (16).

$$A = [0.3, 0.7] \tag{16}$$

So the result according to (14) are:

$$Result = A \cdot R = [0.33, 0.67, 0.35] \tag{17}$$

Thus the most suitable device is device 2, as it has the highest score.

3.2. Reconfiguration stage 2 - reconfiguration optimisation engine

In the first stage of reconfiguration, the assets to use have been selected, but their layout and process parameters have not been selected. In the second stage of reconfiguration, a simulation environment will be utilised for process optimisation and reconfiguration. The system will be optimised based on user-selected KPIs such as operation time, energy consumption, and cost. In addition, there may be some constraints, such as the total reconfiguration cost not exceeding a given budget. The system will find the best solutions based on the KPIs and constraints by selecting one or more modular optimisation algorithms, and the optimised scenario will then be updated in the MRKG. The MRKG will store the selected algorithm(s), the algorithm's result, and the

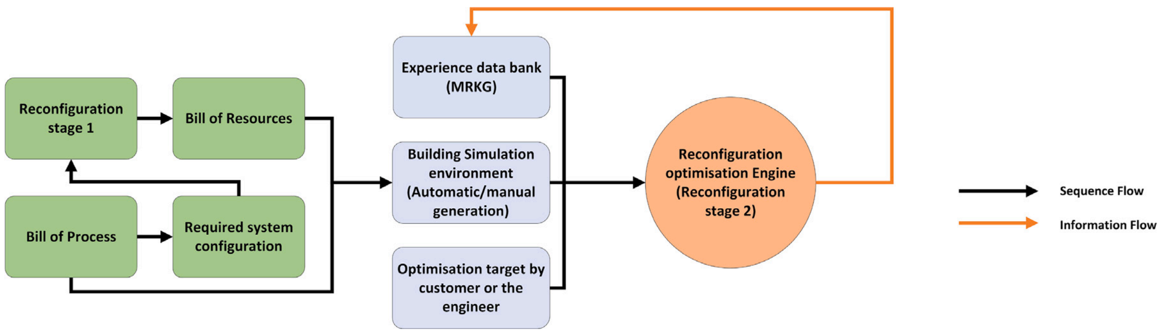

Reconfiguration stage 1
Bill of Resources
Bill of Process
Required system configuration
Experience data bank (MRKG)
Building Simulation environment (Automatic/manual generation)
Optimisation target by customer or the engineer
Reconfiguration optimisation Engine (Reconfiguration stage 2)
Sequence Flow
Information Flow

Fig. 7. Workflow of utilising the reconfiguration optimisation engine.

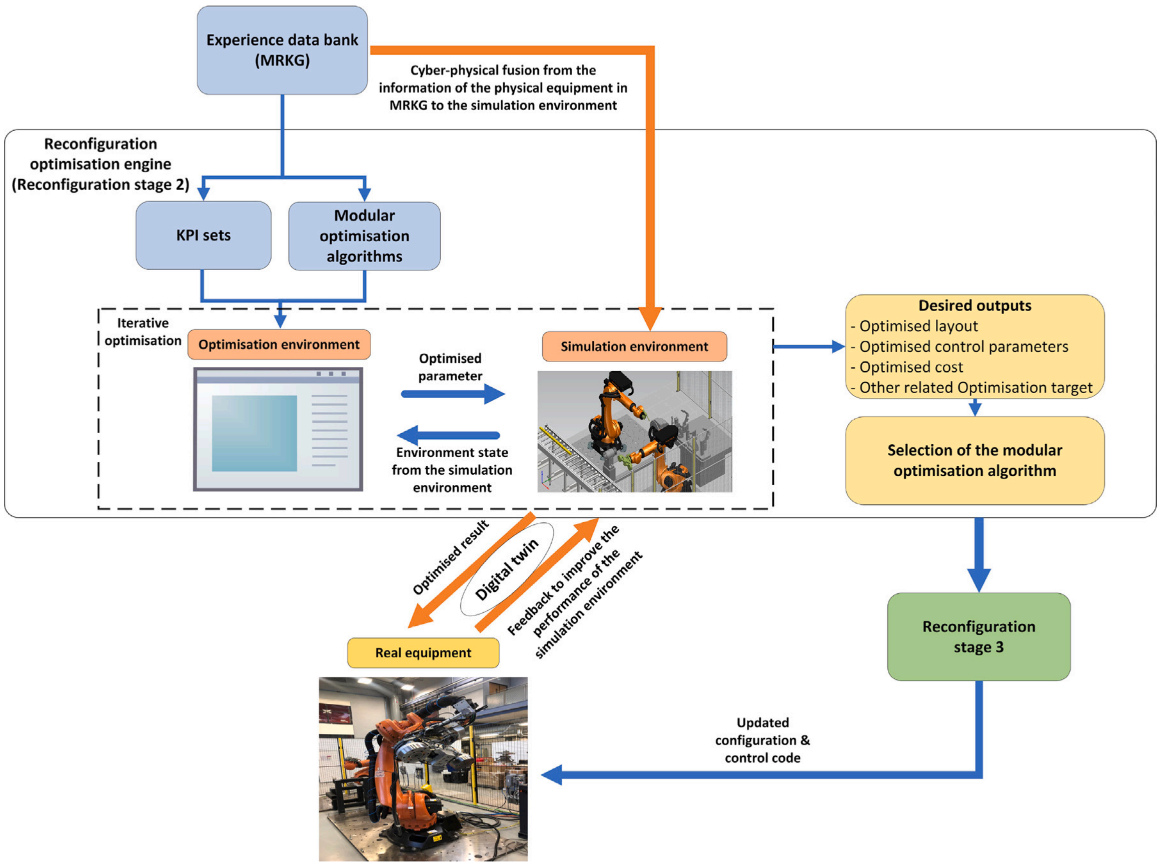

Experience data bank (MRKG)
Cyber-physical fusion from the information of the physical equipment in MRKG to the simulation environment
Reconfiguration optimisation engine (Reconfiguration stage 2)
KPI sets
Modular optimisation algorithms
Iterative optimisation
Optimisation environment
Optimised parameter
Environment state from the simulation environment
Simulation environment
Desired outputs
- Optimised layout
- Optimised control parameters
- Optimised cost
- Other related Optimisation target
Selection of the modular optimisation algorithm
Optimised result
Digital twin
Feedback to improve the performance of the simulation environment
Real equipment
Reconfiguration stage 3
Updated configuration & control code

Fig. 8. Details of utilising the reconfiguration optimisation engine.

related simulation environment information used in the optimisation. The reconfiguration optimisation engine can select multiple algorithms to determine the system configuration for a given product and select the results to utilise based on which algorithm performed best. Optimisation algorithms are (currently manually) stored and labelled in the MRKG. A common method for optimisation is based on discrete event simulation, such as resource allocation optimisation [31]. However, in complex manufacturing use cases, many types of optimisation decisions are needed to specify the reconfiguration. These include asset locations, asset programs, and the operation sequence.

Fig. 7 details the process of utilising the reconfiguration optimisation engine. From this figure, we can see that, at first, a simulation environment should be generated based on the bill of process from the product requirement and the bill of resources from reconfiguration stage 1, either automatically or manually. The target of the optimisation in the reconfiguration optimisation engine will be set by the customer or the engineer based on the product requirement and the historical data. At the same time, the experience data bank (which, in our case: MRKG) will be queried to provide KPI sets and modular optimisation algorithms based on the customer requirement and the previous experience data. Once the KPIs are selected, the reconfiguration optimisation engine will execute the optimisation process. Based on the desired outputs, one or more modular optimisation algorithms will be selected. After the optimisation process, the desired outputs will be given, such as the optimised layout, control parameters, cost, or another related optimisation target.

The detailed information on the reconfiguration optimisation engine is shown in Fig. 8. The reconfiguration optimisation engine consists of four parts: the simulation environment, the optimisation environment, KPI sets, and modular optimisation algorithms. The optimisation environment will get information about the KPI sets and modular optimisation algorithms from the experience data bank and execute them, and the simulation environment is used to test the results. Below is a detailed description of these four parts.

(1) **Simulation environment**
Digital twins are introduced as a combination of three primary components: a virtual twin, a physical twin that corresponds to it, and the data flow component. The data flow component is used to feed data from a physical twin to its virtual twin and returns information from the virtual twin. In a more complicated use case, many types of data representations are needed, such as device location, robot path, operation sequence, cycle time, collision status, and utilisation of the devices. A simulation environment is needed to simulate the process. Besides, the simulation environment should be compatible and powerful enough to simulate the manufacturing process. The simulation environment is created based on the information stored in the MRKG from the real equipment. The entities in the simulation environment are mapped with the entity on the real equipment.

(2) **Optimisation environment**
The optimisation environment is a software library which can process the data from the simulation environment. This optimisation environment can be either deployed on the cloud or locally, depending on customer requirements. Cloud-based optimisation is recommended in this system because it facilitates the modularity of the pipeline components. As for the cloud platform, the KPIs are sent to the cloud data lake. And it will be updated in every iteration. There exists a trigger script that continuously listens for the event of the data lake population. As the data lake becomes populated with the data from the simulation environment, it instantiates another script that validates the input data against the schema (for valid requirements). The model endpoint is queried in terms of input data to get optimised values. These values are sent back to the simulation environment. Then the process will be repeated. An enabler is needed to enable the communication between the cloud platform (optimisation environment) and the simulation environment, such as REST API [96]. The optimisation process is an iterative process with the simulation environment. The iteration time is dependent on the customer requirement and the experience data.

(3) **Modular optimisation algorithms**
The experience data bank will provide a modular optimisation algorithm to fulfil the optimisation targets. In the optimisation environment, different modular artificial intelligence algorithms and approaches are used to optimise the simulation environment to meet the different optimisation targets. The selection process will utilise the same evaluation methods proposed in Section 3.1, but for matching algorithms to problems. For example, the evaluators for evaluating different methods can be the best results they can reach, the efficiency and the cost. The information on the desired outputs and the results of the selection of the modular optimisation algorithm will be stored in the experience data bank as proposed at the beginning of Section 3. For example, if the optimisation target is multi-objective optimisation, then algorithms such as NSGA-II (non-domination-based genetic algorithm for multi-objective optimisation) [97] and MOGA (Multi-Objective Genetic Algorithm) [98] are selected. If the manufacturing system looks to optimise a single parameter with the new product – such as faster operation or lower cost – then reinforcement learning can be used in the optimisation environment. For highly complex or difficult problems, deep reinforcement learning will be suggested by the decision engine.

(4) **KPI sets**
KPIs in production are common ways to assess the performances of manufacturing systems. They provide manufacturing systems with the ability to define targets, evaluate performance and make operational decisions. Being directly connected to the automation level, they allow timely control and quantification of production related to a specific system input, such as the utilisation of the devices, the cost, the cycle time and the energy consumption. In our proposed reconfiguration optimisation engine, the KPI sets the specific optimisation target as defined by the information from the experience data bank and the knowledge from the engineer.

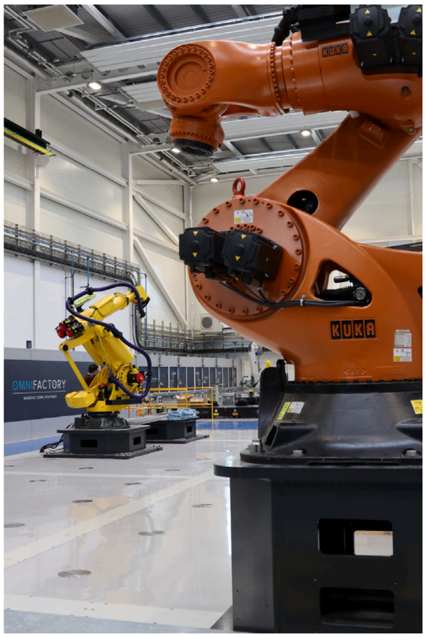


Fig. 9. The reconfigurable floor of the Omnifactory at the University of Nottingham.

Once the reconfiguration stage 2 is finished, the optimised result is found and evaluated successfully in the simulation environment. The optimised result (such as the optimised robot path or the structure of the production line) should be applied to the physical device via the orchestrator. Any updated information from reconfiguration stage 2 will be utilised to update the configuration and control code, which in our proposed framework is reconfiguration stage 3. The orchestrator may make direct changes to PLC code or robot controllers to instantiate the change or make recommendations to human operators to make the changes manually. Then the real equipment will receive the updated configuration information. The corresponding changes will also happen in the real equipment, such as relocation of the robots and sequence change of the operations. The real equipment data can also be used to improve the performance of the simulation environment.

The simulation environment, the optimisation environment and the real equipment can be connected via different communication methods. The communication methods can be sockets, OPC-UA, MQTT, TCP/IP (to connect to external systems such as a PLC simulator), or other communication protocols depending on the application domain and can be dynamically chosen by the reconfiguration optimisation engine. This communication method will be suggested to be used as the middleware between the simulation environment and the optimisation environment. And it can also be used as the communication channel between different devices in the simulation environment and the real devices. For instance: if a cloud-based environment is needed, then the MQTT approach is the better approach to enable communication. If PLC controls the real physical system, then a PLC simulator is a good solution to optimise the system because it can simulate the process of PLC control. If the workstations in the manufacturing system use different operating systems, then OPC UA is a better solution.

4. Validation

This section describes a demonstrator used to verify the effectiveness of the proposed approach in achieving optimised system reconfiguration. We begin by describing the use case, describe how the steps

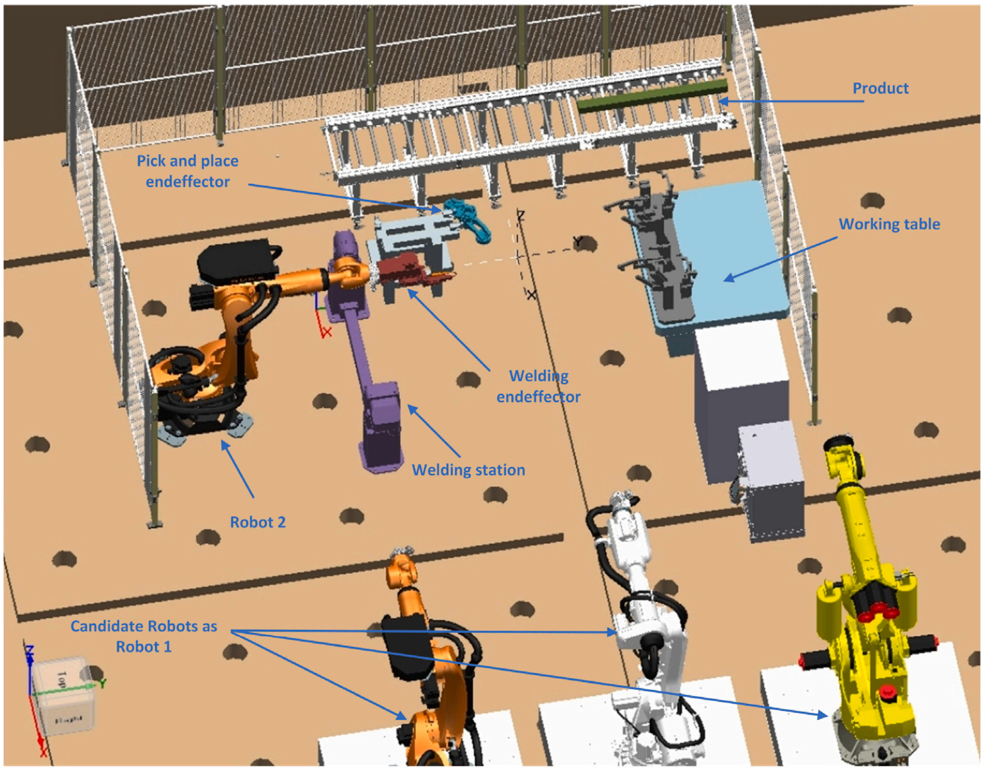


Fig. 10. Layout of the simulation environment.

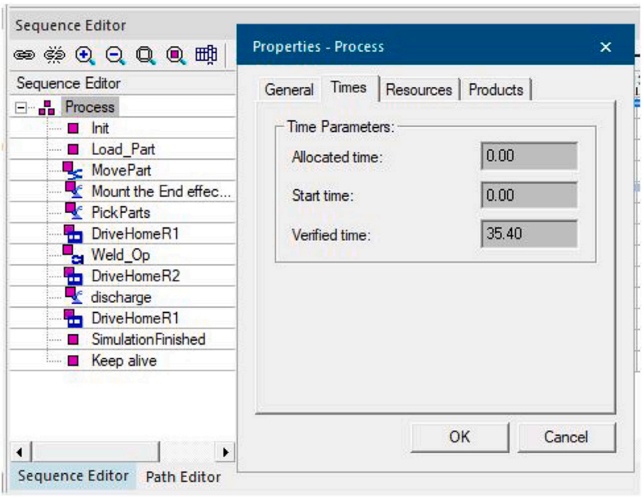
of the reconfiguration optimisation methodology are implemented, and then describe the results of the optimisation process.

4.1. Use case description

The methodology proposed in this paper can be used to determine the optimum locations for robots in simulation before they are permanently located. However, the presented use case here applies to a modern reconfigurable manufacturing systems approach that allows the robot to be relocated rapidly, either with a reconfigurable flooring solution (which can be seen in Fig. 9), or with a robot placed on an AGV that can be moved to the new position after it receives the optimised location. KUKA matrix production is one example of a highly reconfigurable robotic manufacturing solution [99].

The following example illustrates a use case based on a multi-purpose robotic manufacturing cell, using Siemens Tecnomatix Process Simulate as the digital twin enabler alongside a knowledge graph to store data. As illustrated in Fig. 10, a simulation environment is built in Siemens Tecnomatix Process Simulate, which consists of two KUKA KR 270 R2700 ultra robots, one FANUC M-900iB/360 robot, one ABB IRB 6700-150 robot, one conveyor to transport the product from the previous step in the manufacturing process, one welding station, one work table to store parts after welding, one tool stand to store the end effectors, and three Automated Ground Vehicles (AGVs) to transport the robots to new locations. The end effectors are one pick-and-place end effector and one welding end effector, with both initially put on the tool stand. This scenario is based on a real reconfigurable robotic manufacturing cell at the University of Nottingham.

The goal of this use case is to have new tasks assigned to a manufacturing cell, and for the system to determine if a reconfiguration is necessary and what the optimum reconfiguration should be. In our use case, it is assumed that the customer proposes a new task, and the task is that the customer wants a welded rod product with a specific dimension. The tasks that will be assigned are pick-and-place and welding. There will be a selection process between one of the



Sequence Editor
- Process
  - Init
  - Load_Part
  - MovePart
  - Mount the End effec...
  - PickParts
  - DriveHomeR1
  - Weld_Op
  - DriveHomeR2
  - discharge
  - DriveHomeR1
  - SimulationFinished
  - Keep alive

Time Parameters:

Allocated time:	0.00
Start time:	0.00
Verified time:	35.40

Fig. 11. The baseline of the total cycle time which needs to be improved.

KUKA KR270 ultra robots, the FANUC M-900iB/360 robot and the ABB IRB 6700-150 robot for executing the pick-and-place task. The selected robot will be named robot 1. The other KUKA KR270 ultra robot is already selected as the welding robot with the name of robot 2. The locations of robot 1 and robot 2 should then be optimised. The baseline of the optimisation target is listed and set with the condition of the product requirement. Robot 1 and robot 2 are at the customer-defined position initially where robot 1 stays in the middle between robot 2 and the welding station. At these positions, robot 1 and robot 2 can execute all the operations required. As depicted in Fig. 11, the baseline of the total operation time is 35.40 s.

The bill of process to achieve this is listed below:

(1) **AGV transport:** An AGV will transport the selected robot to the initial position.

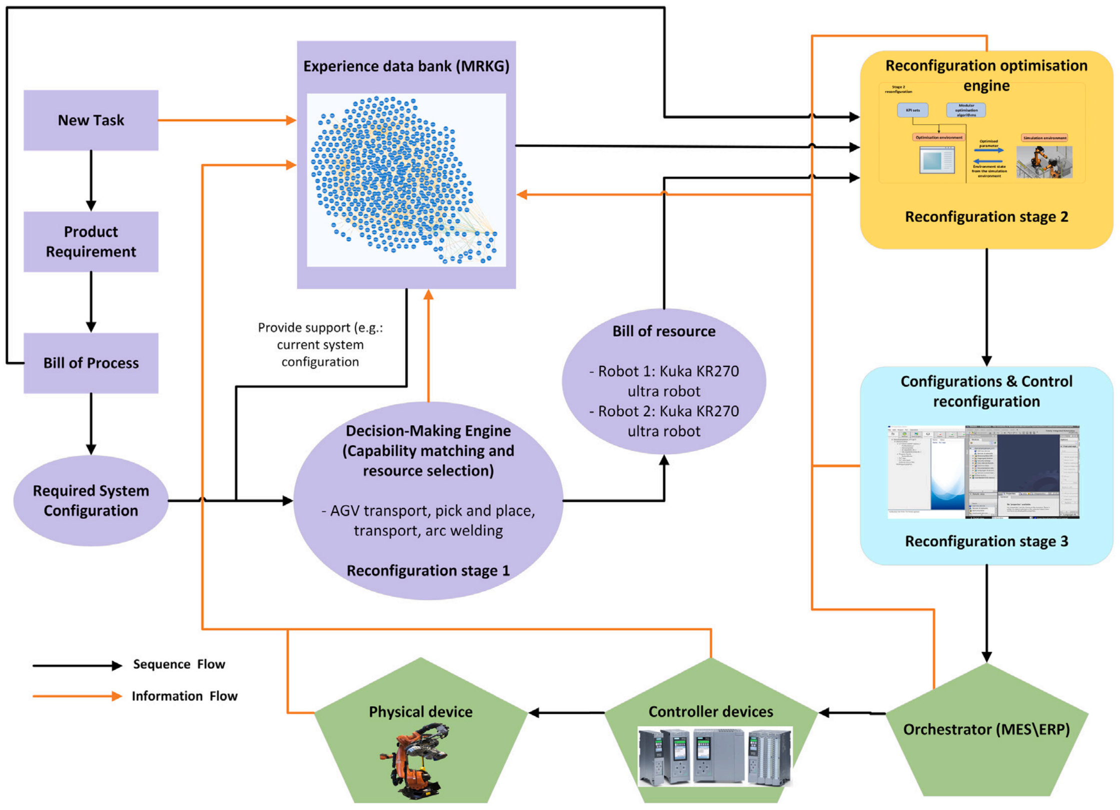

Experience data bank (MRKG)
New Task
Product Requirement
Bill of Process
Required System Configuration
Provide support (e.g.: current system configuration
Decision-Making Engine (Capability matching and resource selection)
- AGV transport, pick and place, transport, arc welding
Reconfiguration stage 1
Bill of resource
- Robot 1: Kuka KR270 ultra robot
- Robot 2: Kuka KR270 ultra robot
Reconfiguration optimisation engine
Reconfiguration stage 2
Configurations & Control reconfiguration
Reconfiguration stage 3
Sequence Flow
Information Flow
Physical device
Controller devices
Orchestrator (MES\ERP)

Fig. 12. Reconfiguration Process for the specific use case.

(2) **Mounting end effector:** The robot will mount the pick-and-place end effector.
(3) **Pick-and-place:** The product is placed on the conveyor by the previous process in the production line.
(4) **Transport:** The part will be transported by conveyor to robot 1 for picking.
(5) **Pick-and-place:** Robot 1 will pick the part up and move it to the welding station.
(6) **Arc welding:** After the part is successfully put on the welding station, robot 2 will perform arc welding on the product.
(7) **Pick-and-place:** After welding, robot 1 will pick up the product, put it on the work table, and return to the home position.

As with the proposed framework described in Fig. 12, there are three stages to intelligent reconfiguration. In reconfiguration stage 1, the decision-making engine should decide if the currently available capability of the production line can meet the production requirement. There will also be an asset selection process if there is more than one candidate asset. For reconfiguration stage 1, it is necessary to build the initial knowledge graph, understand the product requirement, and find the bill of process and bill of the resources to produce the expected product from the customer.

For reconfiguration stage 2, the reconfiguration optimisation engine will receive the decision from stage 1 and optimise the configuration to maximise (or minimise) user-selected KPIs. As described in Section 3.2, the corresponding optimisations are done with the help of the simulation environment and modular optimisation algorithms, such as PSO, GA and SA algorithms. Here, total operation time and collision status are used as the KPIs to minimise. An additional constraint in this use case is to ensure that neither robot collides with other parts during operations and that the robots can reach the required parts.

Stage 3 of the intelligent reconfiguration process is for the corresponding control code (e.g. the PLC code and robot programs) to be automatically reconfigured and updated on the hardware controllers, thus enabling the reconfiguration process to be as automated as possible.

After the three stages of reconfiguration optimisation are finished, the physical reconfiguration begins. Both robots need to relocate to their optimal locations (based on optimal locations from the reconfiguration optimisation engine). Lastly, the results from the simulation environment and the real equipment will be stored and updated in the original knowledge graph for future reference as shown in Fig. 13 .

In our framework, the first step is to build the MRKG to enable the decision-making engine. As depicted in Fig. 13, the generated knowledge graph is stored as a graph format in Neo4j. We will now detail the implementation of stage 1 and stage 2 of the intelligent reconfiguration process.

4.2. Decision-making engine implementation (reconfiguration stage 1)

The decision-making engine will take information from the MRKG. As described in Section 4.1, the product's required system configuration (i.e. the capabilities which must be available) is "Pick", "Transport", "Arc welding", and "Place". All the manufacturing data (current system configuration and historical data) is stored in graph format in Neo4j as the experience data bank in our proposed framework. With the "match" function of Neo4j or via the semantic search, the available resources which have the required capability can be selected [100]. If the process is new and unknown, a semantic search method will be utilised [101]. After the candidate assets are selected (i.e. those which could perform the task), there will be a process to select the most suitable asset. As shown in Section 3.1, evaluators are used to choose the most suitable assets.

Based on the MRKG and the engineer's experience, evaluators are defined and used to select the most suitable assets for our applications. Our research uses fuzzy evaluation to select the optimal resources [102].

The implementation of our proposed method for this specific use case is divided into the following steps. (1) An evaluation matrix R is defined by the engineer at first. In future cases, the matrix R

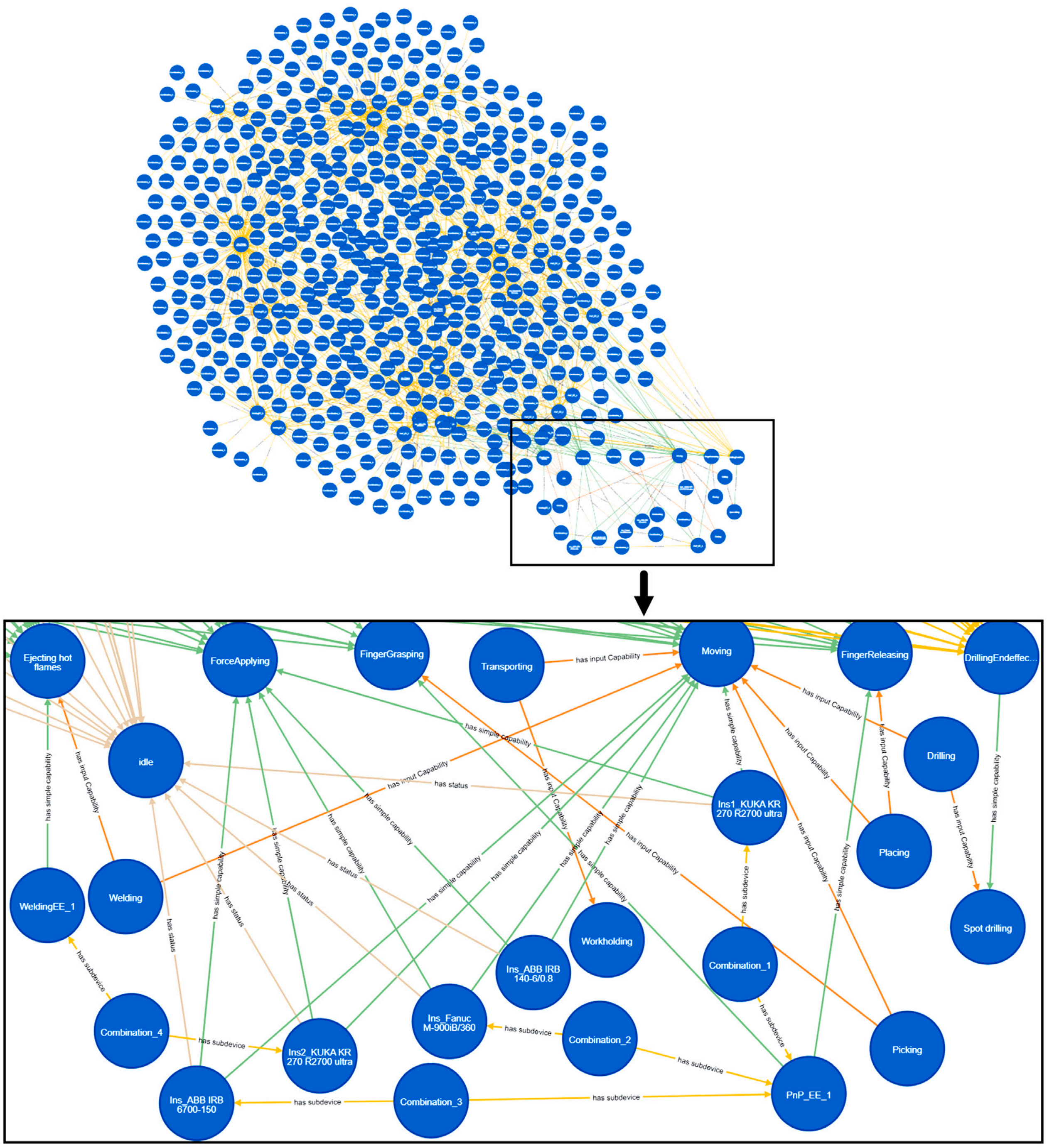


Fig. 13. Data stored in graph format.

Table 2
Requirements.

Repeatability	> 0.05
Asset utilisation	> 50%
Payload	> 140
Reachability	> 2600

could be generated with the combination of the human experience and experience from the knowledge graph. In our use case, there are four types of evaluators: repeatability, current asset utilisation, payload, and reachability. The product requirement in our use case is listed in Table 2.

The deviation between the evaluators and the features from the candidate assets (KUKA KR 270 R2700 ultra robot, ABB IRB 6700-150 robot, FANUC M-900iB/360 robot) will be calculated. Based on the method proposed in Section 3.1, Table 3 shows the detailed information of the candidate assets from the experience data bank.

As mentioned in Section 3.1, an evaluation matrix will be defined to evaluate the candidate assets. In our validation case, we have utilised the consistent fuzzy matrix.

In our case, the fuzzy priority relations based on (5) will be built at first. We define that the closer of features of the evaluators of the candidate assets to the evaluators, the higher the score will be. Taking ‘Repeatability’ as an example, the matrix will be defined in (18)

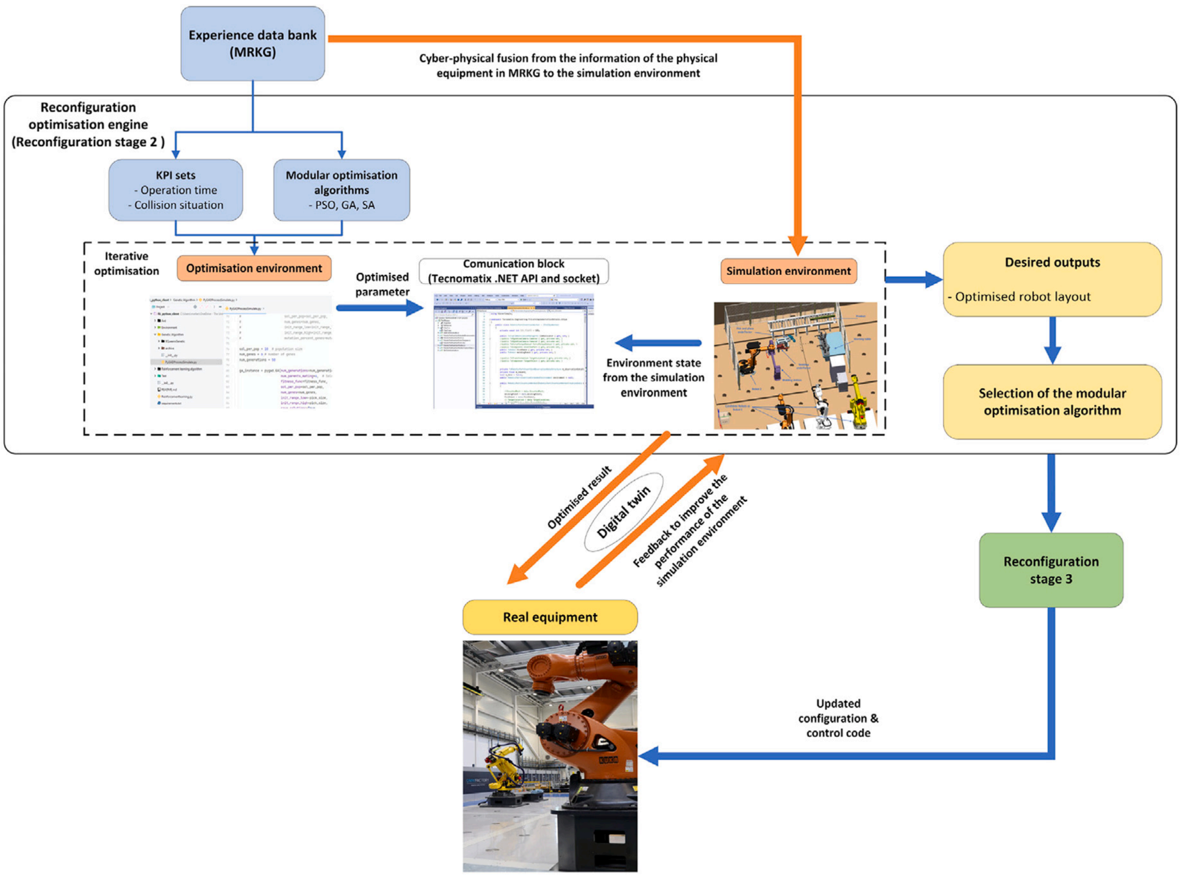

Experience data bank (MRKG)
Cyber-physical fusion from the information of the physical equipment in MRKG to the simulation environment
Reconfiguration optimisation engine (Reconfiguration stage 2)
KPI sets - Operation time - Collision situation
Modular optimisation algorithms - PSO, GA, SA
Iterative optimisation
Optimisation environment
Optimised parameter
Comunication block (Tecnomatix .NET API and socket)
Simulation environment
Environment state from the simulation environment
Desired outputs - Optimised robot layout
Selection of the modular optimisation algorithm
Optimised result
Digital twin
Feedback to improve the performance of the simulation environment
Real equipment
Reconfiguration stage 3
Updated configuration & control code

Fig. 14. Utilisation of the reconfiguration optimisation engine in the application.

Table 3
Candidate asset information.

Evaluators	KUKA KR 270 R2700 ultra	ABB IRB 6700-150	FANUC M-900iB/360
Evaluator 1 - Repeatability	0.06	0.06	0.3
Evaluator 2 - Asset utilisation	80%	60%	70%
Evaluator 3 - Payload	270	150	700
Evaluator 4 - Reachability	2696	3200	2832

according to (5).

$$B_1 = \begin{bmatrix} 0.5 & 0.5 & 1 \\ 0.5 & 0.5 & 1 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{18}$$

The consistent fuzzy matrix can be calculated in (19) according to (10) and (13).

$$R_1 = \begin{bmatrix} 0.5 & 0.5 & 0.75 \\ 0.5 & 0.5 & 0.75 \\ 0.25 & 0.25 & 0.5 \end{bmatrix} \tag{19}$$

The score $s_d^1 (d = 1, 2, 3)$ of the three candidate assets under factor 'Repeatability' can be calculated in (20) according to (10).

$$s_1^1 = 0.392, s_2^1 = 0.392, s_3^1 = 0.216 \tag{20}$$

With the same approach, the score of the three candidate assets under other evaluators can be calculated according to (10). Table 4 shows the scores that each candidate asset got under different single evaluators.

Table 4
Score of each candidate asset under single evaluators.

	KUKA KR 270 R2700 ultra	ABB IRB 6700-150	FANUC M-900iB/360
Repeatability	0.392	0.392	0.216
Asset utilisation	0.431	0.22	0.349
Payload	0.335	0.454	0.211
Reachability	0.454	0.211	0.335

Table 5
Total scores of each candidate asset.

	KUKA KR 270 R2700 ultra	ABB IRB 6700-150	FANUC M-900iB/360
Total score	0.403	0.319	0.278

In this application, we assign equal importance to each evaluator, which means the weight of each evaluator is 0.25. The score of the candidate assets can be calculated by (14). Table 5 shows the total scores for each candidate asset. So in our validation use case, the KUKA KR270 ultra robot is selected as robot 1.

4.3. Reconfiguration optimisation engine implementation (reconfiguration stage 2)

After the asset is selected, the reconfiguration optimisation engine in our proposed framework will be executed. The framework of the reconfiguration optimisation engine for this use case is shown in Fig. 14. The simulation environment is created based on the result from the reconfiguration stage 1 and the information from the real equipment. The simulation environment will send information to the optimisation environment through a socket based on the Tecnomatix .NET API.

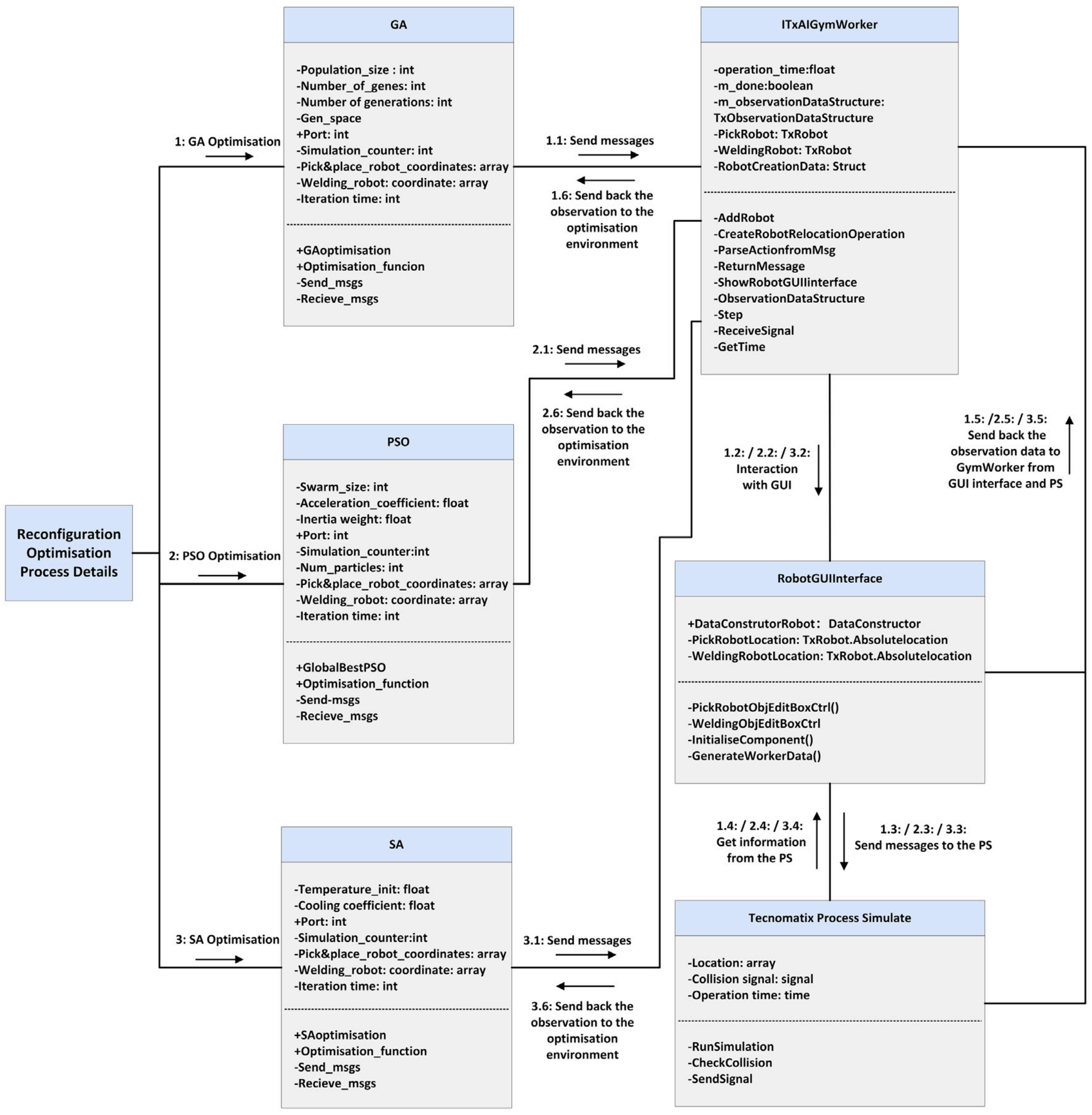

GA
-Population_size : int
-Number_of_genes: int
-Number of generations: int
-Gen_space
+Port: int
-Simulation_counter: int
-Pick&place_robot_coordinates: array
-Welding_robot: coordinate: array
-Iteration time: int
+GAoptimisation
+Optimisation_funcion
-Send_msgs
-Recieve_msgs
ITxAIGymWorker
-operation_time:float
-m_done:boolean
-m_observationDataStructure: TxObservationDataStructure
-PickRobot: TxRobot
-WeldingRobot: TxRobot
-RobotCreationData: Struct
-AddRobot
-CreateRobotRelocationOperation
-ParseActionfromMsg
-ReturnMessage
-ShowRobotGUIInterface
-ObservationDataStructure
-Step
-ReceiveSignal
-GetTime
1: GA Optimisation
1.1: Send messages
1.6: Send back the observation to the optimisation environment
2.1: Send messages
2.6: Send back the observation to the optimisation environment
PSO
-Swarm_size: int
-Acceleration_coefficient: float
-Inertia weight: float
+Port: int
-Simulation_counter:int
-Num_particles: int
-Pick&place_robot_coordinates: array
-Welding_robot: coordinate: array
-Iteration time: int
+GlobalBestPSO
+Optimisation_function
-Send-msgs
-Recieve_msgs
Reconfiguration Optimisation Process Details
2: PSO Optimisation
1.2: / 2.2: / 3.2: Interaction with GUI
1.5: /2.5: / 3.5: Send back the observation data to GymWorker from GUI interface and PS
RobotGUIInterface
+DataConstrutorRobot: DataConstructor
-PickRobotLocation: TxRobot.Absolutelocation
-WeldingRobotLocation: TxRobot.Absolutelocation
-PickRobotObjEditBoxCtrl()
-WeldingObjEditBoxCtrl
-InitialiseComponent()
-GenerateWorkerData()
1.4: / 2.4: / 3.4: Get information from the PS
1.3: / 2.3: / 3.3: Send messages to the PS
SA
-Temperature_init: float
-Cooling coefficient: float
+Port: int
-Simulation_counter:int
-Pick&place_robot_coordinates: array
-Welding_robot: coordinate: array
-Iteration time: int
+SAoptimisation
+Optimisation_function
-Send_msgs
-Recieve_msgs
3: SA Optimisation
3.1: Send messages
3.6: Send back the observation to the optimisation environment
Tecnomatix Process Simulate
-Location: array
-Collision signal: signal
-Operation time: time
-RunSimulation
-CheckCollision
-SendSignal

Fig. 15. Detailed flowchart of the optimisation process based on these three algorithms.

This API is connected to the Tecnomatix Process Simulate simulation environment with Tecnomatix .NET viewers.

In our use case, the operation time and collision status are the “user-defined” KPIs that the customer wants to improve. To evaluate the results of the modular algorithm, the number of iterations and improvements compared to the baselines are KPIs that were stored in the experience databank. The reconfiguration optimisation engine will use these criteria to select the optimisation algorithm which has the best performance.

In the reconfiguration optimisation engine, there is a loop of the optimisation process in the optimisation environment. After one optimisation via the selected artificial intelligence approach completes, the updated robot parameter will be sent to the simulation environment to get a new operation time and then check for collisions. The CEE (Cyclic Event Evaluator) simulation mode in Tecnomatix Process Simulate is used in this example. CEE, which functions as a PLC, is used to control how a typical robotics simulation progresses using logic. Once the start signal is true, the simulation will start. Originally, there were no robot move relocation functions defined. With Tecnomatix .NET API, the move operation can be generated in each simulation. In the first iteration, the optimisation environment will send random coordinates of both robots to the simulation environment, and then two object flow operations for relocating the robots in Tecnomatix Process Simulate will be generated and linked with other operations.

This application uses three algorithms to determine an optimal setup based on the framework proposed in Section 3.2. These three algorithms are recommended algorithms from the experience databank. They are the global PSO, GA, and SA approaches. The flowchart for the optimisation process based on the GA, PSO and SA is illustrated below in Fig. 15. The potential condition describes the range of the initial locations of the robots. Table 6 describes the details of the initial conditions.

The initial coordinates of the robots will be randomly selected within the preset range. After the initial coordinates are sent to the simulation environment, the output value generated from the simulation environment is the process time for the operation or a high failure value (in this case, 60), which means there is either a collision in the simulation environment or objects or not reachable by the robots. As mentioned before, the KPIs in our use case are total operation time and collision status. In more complicated optimisation scenarios, more KPIs can be implemented. The KPIs are progressively optimised in our reconfiguration optimisation engine. Fig. 16 shows the results we receive from the three different algorithms in the reconfiguration optimisation engine. The final optimised robot locations will be sent to

Table 6
The range of the coordinates of the two robots.

Range of the coordinates of pick-and-place robot	Range of the coordinates of welding robot
([−2200, 2200], [−2200, 2200], [0, 0])	([−500, 500], [−500, 500], [0, 0])

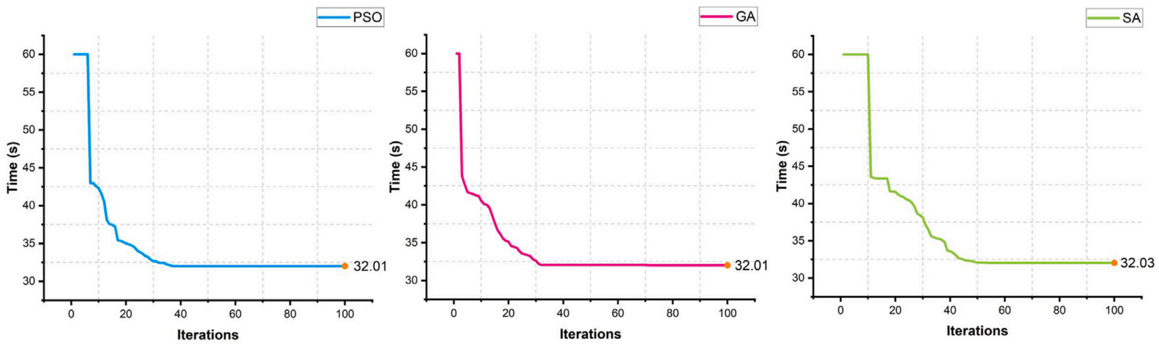


Fig. 16. Comparison between different approaches.

Table 7
Reconfiguration improvement under condition 1.

Algorithm	Reconfiguration efficiency (Iterations needed to find the potential solution)	Improvement compared to baseline
PSO	7	9.58%
GA	3	9.58%
SA	11	9.52%

the orchestrator, and stage 3 of the reconfiguration process will be used to update the controllers with the new configuration.

From Fig. 16 and Table 7, we find that the GA algorithm converges faster than the PSO algorithm approach and the SA approach. The optimised process time found by the GA, PSO and SA is almost the same: 32.01 s for the GA and PSO algorithms and 32.03 s for the SA algorithm. Compared with the baseline (35.40 s), they have reduced the time by 9.58%, 9.58% and 9.52%, respectively. Based on this information, an evaluation will be executed to select the best approach. It can save this information in the experience data bank as described in Fig. 12, to suggest GA as the modular optimisation algorithm for similar future problems because GA found the best time and converged fastest.

There are some limitations to the approach. For instance, if the initial position of the robot is too far from the work-piece where they cannot execute the operations, the result will not easily converge—if the robots cannot execute the operations, the optimisation environment will always receive a penalty (bigger than the operation time) instead of the valid operation time. Convergence speed is also highly dependent on the chosen penalty value.

4.4. Configuration & control code update implementation (reconfiguration stage 3)

Configuration & control code updates will be executed as the last stage of our proposed reconfiguration framework. In our current use case, Siemens TIA Openness API was used to implement the automatic configuration update stage [103].

5. Conclusions and future work

Complex and flexible manufacturing systems are increasingly changeable as the markets require quicker responses to new products, supply disruptions, and volume demands. Reconfiguring and optimising the production process in response to external changes is a difficult challenge for complex and flexible systems. This paper proposes a novel reconfiguration framework to enable the system to find the most suitable devices and find the optimised configurations with different criteria. A knowledge graph-based approach has been used for decision-making in response to changing customer or market needs. With this framework, the manufacturing system reconfiguration can be enabled at first in the simulation environment and deployed to the physical system. A use case with multiple robots and multiple tasks has been used to validate our approach. Compared with the baseline, the optimised KPI has increased by about 10%.

For future work, more key performance indicators will be introduced as optimisation criteria for this framework. Furthermore, more complicated use cases will be considered, including applications where manufacturing is not limited to one workstation. Manufacturing standards will be integrated into this framework, including using the RAMI architecture to describe manufacturing knowledge [104]. Our proposed reconfiguration knowledge graph can be applied to more specific manufacturing domains, such as aerospace, pharmaceuticals, and automotive. Finally, we aim to publish a comprehensive dataset of manufacturing reconfiguration scenarios; with this dataset, further research can be carried out into knowledge graphs, link prediction [105], recommendation systems [106], and reinforcement learning method for knowledge graph reasoning [107].

CRediT authorship contribution statement

Fan Mo: Conceptualization, Methodology, Software, Validation, Writing – original draft. **Hamood Ur Rehman:** Validation, Writing – original draft. **Fabio Marco Monetti:** Writing – original draft. **Jack C. Chaplin:** Conceptualization, Writing – review & editing, Supervision. **David Sanderson:** Conceptualization, Writing – review & editing, Supervision. **Atanas Popov:** Writing – review & editing, Supervision. **Antonio Maffei:** Writing – review & editing, Supervision. **Svetan Ratchev:** Resources, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgements

This research is supported by the DiManD Innovative Training Network (ITN) project funded by the European Union through the Marie Skłodowska-Curie Innovative Training Networks (H2020-MSCA-ITN-2018) under grant agreement no. 814078.

References

[1] S.J. Oks, A. Fritzsche, C. Lehmann, The digitalisation of industry from a strategic perspective, in: Proceedings of Research and Development Management Conference 2016, from Science To Society: Innovation and Value Creation, 2016, pp. 3–6.
[2] T. Kollmann, J. Christofor, International entrepreneurship in the network economy: Internationalization propensity and the role of entrepreneurial orientation, J. Int. Entrepreneurship 12 (1) (2014) 43–66.
[3] M.M. Tseng, R.J. Jiao, C. Wang, Design for mass personalization, CIRP Ann. 59 (1) (2010) 175–178.
[4] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, H. Van Brussel, Reconfigurable manufacturing systems, CIRP Ann. 48 (2) (1999) 527–540.
[5] M.M. Mabkhot, P. Ferreira, A. Maffei, P. Podržaj, M. Mądziel, D. Antonelli, M. Lanzetta, J. Barata, E. Boffa, M. Finžgar, Ł. Paśko, P. Minetola, R. Chelli, S. Nikghadam-Hojjati, X.V. Wang, P.C. Priarone, P. Litwin, D. Stadnicka, N. Lohse, Mapping Industry 4.0 enabling technologies into united nations sustainability development goals, Sustainability 13 (5) (2021) 1–35.
[6] C.A. Tisdell, Economic, social and political issues raised by the COVID-19 pandemic, Econ. Anal. Policy 68 (2020) 17–28.
[7] M.S. Kumar, D.R.D. Raut, D.V.S. Narwane, D.B.E. Narkhede, Applications of industry 4.0 to overcome the COVID-19 operational challenges, Diabetes & Metabolic Syndrome: Clinical Res. Rev. 14 (5) (2020) 1283–1289.
[8] A. Napoleone, L.B. Prataviera, Reconfigurable manufacturing: Lesson learnt from the COVID-19 outbreak, in: B. Lalic, V. Majstorovic, U. Marjanovic, G. von Cieminski, D. Romero (Eds.), Advances in Production Management Systems. the Path To Digital Transformation and Innovation of Production Management Systems, Springer International Publishing, Cham, 2020, pp. 457–465.
[9] M. Rapaccini, N. Saccani, C. Kowalkowski, M. Paiola, F. Adrodegari, Navigating disruptive crises through service-led growth: The impact of COVID-19 on Italian Manufacturing firms, Ind. Mark. Manag. 88 (2020) 225–237.
[10] L. Wright, S. Davidson, How to tell the difference between a model and a digital twin, Adv. Model. Simul. Eng. Sci. 7 (1) (2020) 13.
[11] W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, Digital twin in manufacturing: A categorical literature review and classification, IFAC-PapersOnLine 51 (11) (2018) 1016–1022.
[12] C. da Cunha, O. Cardin, G. Gallot, J. Viaud, Designing the Digital Twins of Reconfigurable Manufacturing Systems: application on a smart factory, IFAC-PapersOnLine 54 (1) (2021) 874–879.
[13] J. Davis, T. Edgar, J. Porter, J. Bernaden, M. Sarli, Smart manufacturing, manufacturing intelligence and demand-dynamic performance, Comput. Chem. Eng. 47 (2012) 145–156.
[14] T. Li, S. Sun, M. Bolić, J.M. Corchado, Algorithm design for parallel implementation of the SMC-PHD filter, Signal Process. 119 (2016) 115–127.
[15] Y. Cui, S. Kara, K.C. Chan, Manufacturing big data ecosystem: A systematic literature review, Robot. Comput.-Integr. Manuf. 62 (2020) 101861.
[16] C. Yin, B. Li, J.X. Chen, Construction method of knowledge graph of manufacturing resources in cloud manufacturing environment, in: 2021 IEEE 4th International Conference on Nanoscience and Technology, ICNST, 2021, pp. 28–35.
[17] A. Singhal, Introducing the knowledge graph: things, not strings, 2012, (Accessed: 2022-05-10).
[18] A. Banerjee, R. Dalal, S. Mittal, K.P. Joshi, Generating digital twin models using knowledge graphs for industrial production lines, in: Proceedings of the 2017 ACM on Web Science Conference, WebSci '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 425–430.
[19] J.M. Gómez-Berbís, A. de Amescua-Seco, SEDIT: Semantic digital twin based on industrial IoT data management and knowledge graphs, in: R. Valencia-García, G. Alcaraz-Mármol, J. Del Cioppo-Morstadt, N. Vera-Lucio, M. Bucaram-Leverone (Eds.), Technologies and Innovation, Springer International Publishing, Cham, 2019, pp. 178–188.
[20] J. Akroyd, S. Mosbach, A. Bhave, M. Kraft, Universal digital twin - A dynamic knowledge graph, Data-Centric Eng. 2 (2021) e14.
[21] N. Sahlab, S. Kamm, T. Müller, N. Jazdi, M. Weyrich, Knowledge graphs as enhancers of intelligent digital twins, in: 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems, ICPS, 2021, pp. 19–24.
[22] Y. Cai, B. Starly, P. Cohen, Y.-S. Lee, Sensor data and information fusion to construct digital-twins virtual machine tools for cyber-physical manufacturing, Procedia Manuf. 10 (2017) 1031–1042.
[23] F. Mo, J.C. Chaplin, D. Sanderson, H.U. Rehman, F.M. Monetti, A. Maffei, S. Ratchev, A framework for manufacturing system reconfiguration based on artificial intelligence and digital twin, in: Flexible Automation and Intelligent Manufacturing International Conference (FAIM 2022), 2022.
[24] A.-L. Andersen, K. Nielsen, T.D. Brunoe, Prerequisites and barriers for the development of reconfigurable manufacturing systems for high speed ramp-up, Procedia CIRP 51 (2016) 7–12, 3rd ICRM 2016 International Conference on Ramp-Up Management.
[25] M. Bortolini, F.G. Galizia, C. Mora, Reconfigurable manufacturing systems: Literature review and research trend, J. Manuf. Syst. 49 (July 2017) (2018) 93–106.
[26] C. Rösiö, K. Säfsten, Reconfigurable production system design – theoretical and practical challenges, J. Manuf. Technol. Manag. 24 (7) (2013) 998–1018.
[27] H.U. Rehman, T. Pulikottil, L.A. Estrada-Jimenez, F. Mo, J.C. Chaplin, J. Barata, S. Ratchev, Cloud based decision making for multi-agent production systems, Prog. Artif. Intell.. EPIA 2021. Lecture Notes in Computer Science 12981 (9) (2021) 673–686.
[28] L.A. Estrada-Jimenez, T. Pulikottil, N.N. Hien, A. Torayev, H.U. Rehman, F. Mo, S.N. Hojjati, J. Barata, Integration of cutting-edge interoperability approaches in cyber-physical production systems and industry 4.0, in: P. Rea, E. Ottaviano, J. Machado, K. Antosz (Eds.), Design, Applications, and Maintenance of Cyber-Physical Systems, IGI Global, 2021, pp. 144–172.
[29] A. Torayev, G. Martínez-Arellano, J.C. Chaplin, D. Sanderson, S. Ratchev, Towards modular and plug-and-produce manufacturing apps, Proc. CIRP 107 (2022) 1257–1262.
[30] J. Tang, K. Salonitis, A deep reinforcement learning based scheduling policy for reconfigurable manufacturing systems, Proc. CIRP 103 (2021) 1–7.
[31] B. Zhou, J. Bao, J. Li, Y. Lu, T. Liu, Q. Zhang, A novel knowledge graph-based optimization approach for resource allocation in discrete manufacturing workshops, Robot. Comput.-Integr. Manuf. 71 (2021) 102160.
[32] M. El-Shamouty, X. Wu, S. Yang, M. Albus, M.F. Huber, Towards safe human-robot collaboration using deep reinforcement learning, in: Proceedings - IEEE International Conference on Robotics and Automation, 2020, pp. 4899–4905.
[33] Y. Lu, C. Liu, I. Kevin, K. Wang, H. Huang, X. Xu, Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues, Robot. Comput.-Integr. Manuf. 61 (2020) 101837.
[34] B.R. Barricelli, E. Casiraghi, D. Fogli, A survey on digital twin: Definitions, characteristics, applications, and design implications, IEEE Access 7 (2019) 167653–167671.
[35] F. Tao, H. Zhang, A. Liu, A.Y.C. Nee, Digital twin in industry: State-of-the-art, IEEE Trans. Ind. Inform. 15 (4) (2019) 2405–2415.
[36] R. Rosen, G. von Wichert, G. Lo, K.D. Bettenhausen, About the importance of autonomy and digital twins for the future of manufacturing, in: 15th IFAC Symposium OnInformation Control Problems In Manufacturing, 48, (3) 2015, pp. 567–572,
[37] M. Zhang, F. Tao, B. Huang, A. Liu, L. Wang, N. Anwer, A.Y.C. Nee, Digital twin data: Methods and key technologies, Digitaltwin 1 (2022) 2.
[38] F. Tao, M. Zhang, Y. Liu, A. Nee, Enabling technologies and tools for digital twin, J. Manuf. Syst. 58 (2021) 3–21, Digital Twin towards Smart Manufacturing and Industry 4.0.
[39] M. Grieves, J. Vickers, Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems, in: F.-J. Kahlen, S. Flumerfelt, A. Alves (Eds.), Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches, Springer International Publishing, Cham, 2017, pp. 85–113.
[40] J. Leng, Q. Liu, S. Ye, J. Jing, Y. Wang, C. Zhang, D. Zhang, X. Chen, Digital twin-driven rapid reconfiguration of the automated manufacturing system via an open architecture model, Robot. Comput.-Integr. Manuf. 63 (2020) 101895.
[41] F. Tao, M. Zhang, Y. Liu, A. Nee, Digital twin driven prognostics and health management for complex equipment, CIRP Ann. 67 (1) (2018) 169–172.
[42] N. Kousi, C. Gkournelos, S. Aivaliotis, C. Giannoulis, G. Michalos, S. Makris, Digital twin for adaptation of robots' behavior in flexible robotic assembly lines, Procedia Manuf. 28 (2019) 121–126.
[43] M.C. Magnanini, T.A. Tolio, A model-based Digital Twin to support responsive manufacturing systems, CIRP Ann. 70 (1) (2021) 353–356.
[44] C. Zhang, W. Xu, J. Liu, Z. Liu, Z. Zhou, D.T. Pham, Digital twin-enabled reconfigurable modeling for smart manufacturing systems, Int. J. Comput. Integr. Manuf. 34 (7–8) (2021) 709–733.
[45] P. Tsarouchi, G. Michalos, S. Makris, T. Athanasatos, K. Dimoulas, G. Chryssolouris, On a human–robot workplace design and task allocation system, Int. J. Comput. Integr. Manuf. 30 (12) (2017) 1272–1279.
[46] N. Kousi, C. Gkournelos, S. Aivaliotis, K. Lotsaris, A.C. Bavelos, P. Baris, G. Michalos, S. Makris, Digital twin for designing and reconfiguring human–robot collaborative assembly lines, Appl. Sci. 11 (10) (2021) 4620.

[47] D. Nåfors, B. Johansson, P. Gullander, S. Erixon, Simulation in hybrid digital twins for factory layout planning, in: 2020 Winter Simulation Conference (WSC), 2020, pp. 1619–1630.
[48] A. Lind, D. Högberg, A. Syberfeldt, L. Hanson, D. Lämkull, Evaluating a digital twin concept for an automatic up-to-date factory layout setup, in: A.H. Ng, A. Syberfeldt, D. Högberg, M. Holm (Eds.), Advances in Transdisciplinary Engineering, IOS Press, 2022.
[49] W. Lepuschitz, A. Zoitl, M. Vallée, M. Merdan, Toward self-reconfiguration of manufacturing systems using automation agents, IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 41 (1) (2011) 52–69.
[50] J. Wan, B. Yin, D. Li, A. Celesti, F. Tao, Q. Hua, An ontology-based resource reconfiguration method for manufacturing cyber-physical systems, IEEE/ASME Trans. Mechatronics 23 (6) (2018) 2537–2546.
[51] B.I. Epureanu, X. Li, A. Nassehi, Y. Koren, Self-repair of smart manufacturing systems by deep reinforcement learning, CIRP Ann. 69 (1) (2020) 421–424.
[52] K. Khedri Liraviasl, H. El Maraghy, M. Hanafy, S.N. Samy, A framework for modelling reconfigurable manufacturing systems using hybridized Discrete-Event and Agent-based simulation, IFAC-PapersOnLine 28 (3) (2015) 1490–1495.
[53] Y.G. Kim, S. Lee, J. Son, H. Bae, B.D. Chung, Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system, J. Manuf. Syst. 57 (October) (2020) 440–450.
[54] S. Mirjalili, Genetic algorithm, in: Evolutionary Algorithms and Neural Networks, Springer, 2019, pp. 43–55.
[55] S. Yuan, B. Skinner, S. Huang, D. Liu, A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms, European J. Oper. Res. 228 (1) (2013) 72–82.
[56] K. Zhou, J. Hu, Y. Li, Q. Wang, Y. Tong, Research on cloud forging resource service selection optimization based on genetic algorithm, in: Journal of Physics: Conference Series, 1812, (1) IOP Publishing, 2021, 012023.
[57] A. Bensmaine, M. Dahane, L. Benyoucef, A non-dominated sorting genetic algorithm based approach for optimal machines selection in reconfigurable manufacturing environment, Comput. Ind. Eng. 66 (3) (2013) 519–524.
[58] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, 4, IEEE, 1995, pp. 1942–1948.
[59] M.-P. Song, G.-C. Gu, Research on particle swarm optimization: a review, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), 4, IEEE, 2004, pp. 2236–2241.
[60] V. Chawla, A. Chanda, S. Angra, Scheduling of multi load AGVs in FMS by modified memetic particle swarm optimization algorithm, J. Proj. Manag. 3 (1) (2018) 39–54.
[61] L. Han, K. Xing, X. Chen, F. Xiong, A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems, J. Intell. Manuf. 29 (5) (2018) 1083–1096.
[62] O. Erdinc, Optimization in renewable energy systems: recent perspectives, 2017.
[63] D. Bertsimas, J. Tsitsiklis, Simulated annealing, Statist. Sci. 8 (1) (1993) 10–15.
[64] Y. Lin, Z. Bian, X. Liu, Developing a dynamic neighborhood structure for an adaptive hybrid simulated annealing–tabu search algorithm to solve the symmetrical traveling salesman problem, Appl. Soft Comput. 49 (2016) 937–952.
[65] T.-Y. Wang, H.-C. Lin, K.-B. Wu, An improved simulated annealing for facility layout problems in cellular manufacturing systems, Comput. Ind. Eng. 34 (2) (1998) 309–319.
[66] Y. Koren, M. Shpitalni, Design of reconfigurable manufacturing systems, J. Manuf. Syst. 29 (4) (2010) 130–141.
[67] L. Zhou, J. Gao, D. Li, H.-Y. Shum, The design and implementation of xiaoice, an empathetic social chatbot, Comput. Linguist. 46 (1) (2020) 53–93.
[68] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on freebase from question-answer pairs, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1533–1544.
[69] H. Wang, F. Zhang, X. Xie, M. Guo, DKN: Deep knowledge-aware network for news recommendation, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1835–1844.
[70] F. Zhang, N.J. Yuan, D. Lian, X. Xie, W.-Y. Ma, Collaborative knowledge base embedding for recommender systems, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 353–362.
[71] V. Bellini, V.W. Anelli, T. Di Noia, E. Di Sciascio, Auto-encoding user ratings via knowledge graphs in recommendation scenarios, in: Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, 2017, pp. 60–66.
[72] J. Dalton, L. Dietz, J. Allan, Entity query feature expansion using knowledge base links, in: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, 2014, pp. 365–374.
[73] H. Raviv, O. Kurland, D. Carmel, Document retrieval using entity-based language models, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016, pp. 65–74.
[74] F. Ensan, E. Bagheri, Document retrieval model through semantic linking, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 181–190.
[75] I. Grangel-González, L. Halilaj, S. Auer, S. Lohmann, C. Lange, D. Collarana, An RDF-based approach for implementing Industry 4.0 components with Administration Shells, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation, ETFA, IEEE, 2016, pp. 1–8.
[76] D. Fernandes, J. Bernardino, Graph databases comparison: AllegroGraph, ArangoDB, InfiniteGraph, No4J, and OrientDB, in: Data, 2018, pp. 373–380.
[77] Y. Tang, T. Liu, M. He, Q. Wang, H. Zhang, G. Liu, R. Dai, Graph database based knowledge graph storage and query for power equipment management, in: 2020 12th IEEE PES Asia-Pacific Power and Energy Engineering Conference, APPEEC, IEEE, 2020, pp. 1–5.
[78] Y. Ma, W. Xu, S. Tian, J. Liu, B. Yao, Y. Hu, H. Feng, Knowledge graph-based manufacturing capability service optimal selection for industrial cloud robotics, in: International Manufacturing Science and Engineering Conference, 84263, American Society of Mechanical Engineers, 2020, V002T07A022.
[79] T.P. Raptis, A. Passarella, M. Conti, Data management in industry 4.0: State of the art and open challenges, IEEE Access 7 (2019) 97052–97093.
[80] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G.d. Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, Synth. Lect. Data, Semantics, and Knowl. 12 (2) (2021) 1–257.
[81] M.A. Pisching, M.A. Pessoa, F. Junqueira, D.J. dos Santos Filho, P.E. Miyagi, An architecture based on RAMI 4.0 to discover equipment to process operations required by products, Comput. Ind. Eng. 125 (2018) 574–591.
[82] S. Cavalieri, M.G. Salafia, Asset administration shell for PLC representation based on IEC 61131–3, IEEE Access 8 (2020) 142606–142621.
[83] H.U. Rehman, J.C. Chaplin, L. Zarzycki, F. Mo, M. Jones, S. Ratchev, Service based approach to asset administration shell for controlling testing processes in manufacturing, 2022.
[84] A. Lüder, L. Hundt, A. Keibel, Description of manufacturing processes using Automation ML, in: 2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010), IEEE, 2010, pp. 1–8.
[85] H. Zhang, B. Zhu, Y. Li, O. Yaman, U. Roy, Development and utilization of a Process-oriented Information Model for sustainable manufacturing, J. Manuf. Syst. 37 (2015) 459–466.
[86] L. Horvath, I.J. Rudas, Knowledge based generation of Petri net representation of manufacturing process model entities, in: 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No. 96CH35929), 4, IEEE, 1996, pp. 2957–2962.
[87] S.J. Fenves, S. Foufou, C. Bock, R. Bouillon, R.D. Sriram, et al., CPM 2: A revised core product model for representing design information, 2004.
[88] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Trans. Knowl. Data Eng. 29 (12) (2017) 2724–2743.
[89] N. Guan, D. Song, L. Liao, Knowledge graph embedding with concepts, Knowl.-Based Syst. 164 (2019) 38–44.
[90] C. Zhao, C. Ma, H. Zhang, Z. Ma, J. Yang, M. Li, X. Wang, Q. Lv, Modeling manufacturing resources based on manufacturability features, Sci. Rep. 12 (1) (2022) 1–12.
[91] W. Chu, Y. Li, C. Liu, W. Mou, L. Tang, A manufacturing resource allocation method with knowledge-based fuzzy comprehensive evaluation for aircraft structural parts, Int. J. Prod. Res. 52 (11) (2014) 3239–3258.
[92] Z. Xu, Q. Da, An approach to improving consistency of fuzzy preference matrix, Fuzzy Optim. Decis. Mak. 2 (1) (2003) 3–12.
[93] Z.-H. Zou, Y. Yi, J.-N. Sun, Entropy method for determination of weight of evaluating indicators in fuzzy synthetic evaluation for water quality assessment, J. Environ. Sci. 18 (5) (2006) 1020–1023.
[94] Y.-Y. Hsu, K.-L. Ho, Fuzzy expert systems: an application to short-term load forecasting, in: IEE Proceedings C (Generation, Transmission and Distribution), 139, (6) IET, 1992, pp. 471–477.
[95] M.-S. Chen, S.-W. Wang, Fuzzy clustering analysis for optimizing fuzzy membership functions, Fuzzy Sets and Systems 103 (2) (1999) 239–254.
[96] F. Gessert, S. Friedrich, W. Wingerath, M. Schaarschmidt, N. Ritter, Towards a scalable and unified REST API for cloud data stores, in: GI-Jahrestagung, 2014, pp. 723–734.
[97] J. Tong, Y. Li, J. Liu, R. Cheng, J. Guan, S. Wang, S. Liu, S. Hu, T. Guo, Experiment analysis and computational optimization of the Atkinson cycle gasoline engine through NSGA II algorithm using machine learning, Energy Convers. Manage. 238 (2021) 113871.
[98] M.H. Nazari, S.H. Hosseinian, E. Azad-Farsani, A multi-objective LMP pricing strategy in distribution networks based on MOGA algorithm, J. Intell. Fuzzy Systems 36 (6) (2019) 6143–6154.

[99] T. Bányai, Optimization of material supply in smart manufacturing environment: A metaheuristic approach for matrix production, Machines 9 (10) (2021) 220.

[100] C. Johnpaul, T. Mathew, A cypher query based NoSQL data mining on protein datasets using Neo4j graph database, in: 2017 4th International Conference on Advanced Computing and Communication Systems, ICACCS, IEEE, 2017, pp. 1–6.

[101] P. Zheng, L. Xia, C. Li, X. Li, B. Liu, Towards self-x cognitive manufacturing network: An industrial knowledge graph-based multi-agent reinforcement learning approach, J. Manuf. Syst. 61 (2021) 16–26.

[102] K. Mittal, P. Tewari, D. Khanduja, On the fuzzy evaluation of measurement system analysis in a manufacturing and process industry environment: A comparative study, Manag. Sci. Lett. 8 (4) (2018) 201–216.

[103] E. Blanco Viñuela, D. Darvas, G. Sallai, et al., Testing solutions for siemens PLCs programs based on PLCSIM advanced, in: 17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, NY, USA, 05-11 October 2019, JACOW Publishing, Geneva, Switzerland, 2020, pp. 1107–1110.

[104] L. Sun, Research on product attribute extraction and classification method for online review, in: 2017 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration, ICIICII, IEEE, 2017, pp. 117–121.

[105] S. Choudhary, T. Luthra, A. Mittal, R. Singh, A survey of knowledge graph embedding and their applications, 2021, arXiv preprint arXiv:2107.07842.

[106] X. Chen, S. Jia, Y. Xiang, A review: Knowledge reasoning over knowledge graph, Expert Syst. Appl. 141 (2020) 112948.

[107] W. Xiong, T. Hoang, W.Y. Wang, Deeppath: A reinforcement learning method for knowledge graph reasoning, 2017, arXiv preprint arXiv:1707.06690.