Automated digital twin generation of manufacturing systems with complex material flows: graph model completion

Giovanni Lugaresi, Andrea Matta

Automated Digital Twin Generation of Manufacturing Systems with Complex Material Flows: Graph Model Completion

Giovanni Lugaresi[1*] and Andrea Matta[2]

[1] Laboratoire de Génie Industriel, CentraleSupélec – 3 Rue Joliot Curie, 91190 Gif-sur-Yvette, FRANCE

[2] Department of Mechanical Engineering, Politecnico di Milano – Via La Masa 1, 20156 Milano, ITALY

*Corresponding author: giovanni.lugaresi@centralesupelec.fr

Abstract. Industry 4.0 determined the emergence of technologies that enable data-driven production planning and control approaches. A digital model can be used to make decisions based on the current state of a manufacturing system, and its efficacy strictly depends on the capability to correctly represent the physical counterpart at any time. Automated model generation techniques such as process mining can significantly accelerate the development of digital twins for manufacturing systems. However, complex production environments are characterized by the convergence of different material and information flows. The corresponding data logs present multiple part identifiers, resulting in the wrong finding of the system structure with traditional process mining techniques. This paper describes the problem of discovering manufacturing systems with non-linear material flows, such as assembly lines. An algorithm is proposed for the proper digital model generation, aided by the new concept of object-centric process mining. The proposed approach has been applied successfully to two test cases and a real manufacturing system. The results show the applicability of the proposed technique to realistic settings.

Keywords: model generation · discrete event simulation · process mining · assembly · manufacturing · digital twin.

1 Introduction

Recently, the complexity of manufacturing environments has been significantly increasing in order to meet the rising demand for customized products. Meanwhile, production enterprises have been pushed to invest in new technologies toward higher digitization and automation (Rao et al., 2008). Information systems are now essential for the successful day-to-day operations of modern organizations. These systems must be able to meet the new, more demanding requirements and enable the efficient online management of shop floors. The design of modern decision support tools is based on the coexistence of the real system with its digital counterpart, often referred to as *digital twin* (DT) (Negri et al., 2020). With regards to production planning and control phases, DTs can be based on discrete event simulation models with the addition of real-time data streams. DTs can help production planners to evaluate optimal solutions

for the current system state at any time (Tavakoli et al., 2008). However, if the model is not correctly representing the system structure and parameters, any conclusion derived from experimental results is likely to be erroneous and may lead to expensive decisions (Lugaresi and Matta, 2021b).

Meanwhile, Industry 4.0 has contributed to the spreading of new technologies for data handling. Such equipment can provide up-to-date information on the shop floor status (Tao et al., 2018). The increased integration of functionalities allows for a closer coupling of previously separated decisional levels (Rossit and Tohmé, 2018). Faster data transfer speeds allow for quicker decisions to be made based on current data from the shop floor, leading to better online decisions (Monostori et al., 2016). The availability of real-time data enables the generation of models from manufacturing system datasets. This way, the development phase may be significantly shortened, enabling the alignment of digital twins with their real counterparts (Reinhardt et al., 2019).

Recent research adopted process mining techniques for model generation (van der Aalst, 2016, 2018). However, the automated development of digital models for more complex manufacturing systems reaches the limitations of the available methodologies. Indeed, most approaches based on process mining are limited by the so-called *flat data*. Namely, a unique part identifier is used to identify material flows, while in realistic context several different object types may be involved in certain production phases (e.g., batches, packages, orders). The relationships among different objects are disregarded by the traditional methodologies. The result is that certain types of systems cannot be modeled entirely with an automated approach. Assembly processes are commonly used in production industries, such as automotive and electronics. At these assembly stations, multiple material flows intersect. Hence, it is typical for new part identifiers to be given to the newly-assembled parts, either during or after the assembly. Traditional process mining techniques make use of a single part identifier, which limits their ability to identify logical flows (e.g., activity precedences). As a consequence, production systems with non-linear material flows are discovered as a collection of separate, independent models.

The goal of this work is to provide automated support in the development of discrete event simulation models for complex manufacturing systems. The contribution of the paper is threefold: (1) it describes the problem of discovering manufacturing systems with non-linear material flows, (2) it introduces a method based on object-centric process mining to generate a complete graph model of a manufacturing system starting from available datasets, and (3) it introduces quantitative indicators to evaluate the fitness of the generated graph models. The problem of automated model generation is strictly linked to the quality of input data. Indeed, several issues arise in terms of data availability, data cleaning, and preparation for automatic processing. These issues are not within the scope of this paper, and in the remainder it is assumed that complete and clean datasets are available.

The rest of the manuscript is structured as follows. Section 2 introduces the related works on the automated simulation model generation in manufacturing systems, with a focus on the assembly phases. Section 3 describes the characteristics of the problem for the discovery and model generation, pointing

out the information that is available for the discovery and the criteria to guide the mining. Section 4 outlines the proposed method to generate a graph model. Section 5 proposes a solution procedure. Section 6 presents the experiments that have been used to investigate the applicability of the approach and the numerical results. Section 7 contains the authors' final remarks.

2 Related Work

This section lists the significant contributions for generating simulation models of complex manufacturing systems, it outlines how object-centric process mining can support the generation of models for more complex systems, and it introduces the research gap.

2.1 Automated Generation of Simulation Models for Manufacturing Applications

Process mining is a recent research field that includes a plethora of techniques to discover, monitor, and improve real processes by extracting knowledge from data logs available from modern information systems. It is mainly composed by three activities (van der Aalst, 2016): *system discovery* is finding the main logical relationships between activities, *conformance checking* refers to the collection of approaches that find deviations between the discovered model and a standard one, and *enhancement* that is linked to the idea of improving some model properties of interest. Process mining can detect not only parameters and causal relationships, but also logical relationships such as activity precedences. This allows to infer the topological structure of a manufacturing system and to generate a simulation model (Rozinat et al., 2009). Several works in the field of process mining have used it to develop models that can predict the performance of a manufacturing system (i.e. simulation). Rozinat et al. (2009) were the first to propose this approach. A novel methodology for identifying dispatching rules in a manufacturing system has been proposed by Bergmann et al. (2015). Milde and Reinhart (2019) conceived a technique for identifying the material flow, calculating the parameters, and recognizing the control policies from manufacturing systems datasets. Lugaresi and Matta (2021b) devised a method for constructing simulation models with an appropriate level of detail. Process mining has also been used to discover specific parameters of a manufacturing system, such as interarrival times (Martin et al., 2015), determinants of process delays (Ferreira and Vasilyev, 2015), resource availabilities (Martin et al., 2016), batching operations (Martin et al., 2017), and production system structures (Denno et al., 2018).

2.2 Discovery of Systems with Non-linear Material Flows

The available model generation methodologies suffer limitations in the discovery of complex manufacturing systems. Indeed, most approaches based on process mining are relying on the assumption of a single part identifier in the datasets, while in most realistic environments multiple object types may be involved

in a production step (e.g., packaging, batching, assembly). Assembly processes are very common in production enterprises, for instance in the automotive or consumer electronics industries. In the literature, few studies have addressed the application of digital model generation techniques to assembly operations. In order to select significant contributions with which to relate, papers have been collected from Scopus database. The papers are the results from the following query: *("model generation" OR "process mining") AND (manufactur* OR produc*) AND (assembl* OR aggregat* OR merg* OR join*)*. No time-based limitations have been applied, and only papers in the English language have been considered. The query has been executed on 2023-02-06 and resulted in 163 papers. From a title- and abstract-based screening, 17 papers have been selected based on the following exclusion criteria: (1) the paper does not regard the assembly of any type of materials, (2) the paper talks about non-engineering topics, (3) the paper does not mention any type of data-based model generation or analysis. Table 1 lists the selected contributions, which are described in the following.

van der Aalst (2021) introduced the concept of federated process mining and presented a formal framework for using process mining in the analysis of cross-organizational event logs. The author introduces the concept of Merging Abstractions, which consists in the idea of merging different process mining results into a single representation. The paper also highlights the fact that in case of multiple object types, the resulting abstraction is a set of separate graph models. Brockhoff et al. (2022) examined the integration of the structural information of multi-level Bills of Materials into a top-down operational process analysis framework. The goal is to automatically discover Bills of Materials to aid the performance analysis of a complex manufacturing system. The effectiveness of the proposed approach has been tested using data from a real industrial printer assembly factory. Xu et al. (2016) proposed an approach to combine different process event logs using genetic algorithm, effectively automating the process of flattening data. The approach is promising in finding relations within different cases.

Petschnigg et al. (2020) introduced a systematic approach for automatically creating a factory simulation model from a raw point cloud obtained from laser scanning and supervised by a deep-learning-based object segmentation module. The approach has been demonstrated using an automotive assembly plant. The material flow dynamics have not been investigated. Sjarov et al. investigate the modeling phases of a digital twin of production resources. The modeling approach is demonstrated using an assembly cell in a laboratory. The main purpose of the digital twin is the visualization of process performance online. Ng et al. introduced a toolset that integrates model abstraction, automatic model generation, and simulation-based optimization within an internet platform. The toolset features a unique model aggregation and generation technique, which combined with optimization engines can accelerate the time-intensive model building, experimentation, and optimization processes to aid decision-making. The generation method assumes assembly points have already been identified. Rashid and Louis (2020) proposed a framework that combined RFID tracking and process mining techniques to create a digital model of an assembly line and detect any deviations from the predefined plan. Yang et al. presented a general process mining

approach that addresses the diversity issue by classifying cases into sub-logs, which are then processed by multiple process miners to generate separate sets of process models. A genetic algorithm is used to combine the process models into a comprehensive process model, balancing four quality dimensions. The approach has been tested on synthetic and real-life logs from a telecommunications company. Lugaresi and Matta (2021a) described the problem of discovery of manufacturing systems with assembly phases. The authors have proposed an algorithm to automatically generate a graph model including non-linear material flows. A flow shop with synthetic data has been used as case study. The approach is limited to the discovery of assembly steps that correspond to starting points of the respective graph models (i.e., no predecessor activities).

Krenczyk et al. (2017) proposed a heuristics and simulation-based approach for balancing mixed and multi-model assembly lines. The proposed approach consists in a combination of data-driven automatic simulation model generation and heuristic line balancing methods. The implementation is demonstrated using FlexSim simulation software within a practical example. Limère et al. (2013) proposed a mixed integer linear programming that provides an optimal assignment of individual parts to either kitting or bulk feeding, with the goal of minimizing in-plant logistics costs. Denno et al. (2018) used process mining to optimize the production schedule for an automotive under-body assembly system.

Dong et al. (2019) presented a method that integrates process mining and complex network theory to analyze the relationship between resource nodes and process nodes in a manufacturing workflow. The method combines information about processes and resources to identify critical nodes and improve workflow analysis. Knoch et al. (2020) proposed a method to derive process traces from videos that depict assembly procedures recorded from above. The method is based on an algorithm to generate trajectories of a operator hand movements by using input from a neural network-based real-time object detector. The trajectories are automatically classified and linked to work steps through hierarchical clustering of similar behavior patterns using dynamic time warping. Knoll et al. (2019a) developed a methodology to apply process mining to internal logistics for a mixed-model assembly line, which included multi-dimensional process mining to automate and improve the Value Stream Mapping methodology, and to identify areas of the plant where the most waste is produced. Knoll et al. (2019b) developed an internal logistic ontology to facilitate the pre-processing of manufacturing event logs. The authors highlight the multiple-objects nature of shop floor data given by the natural interrelation of several components within production phases. Similarly, Schuh et al. (2020) created a complete data model to enable the application of process mining in end-to-end order processing in production enterprises.

The discovery of multiple-object phases such as assembly has not received enough attention from the literature, although being among the most common production steps. Indeed, with the sole exeption of Lugaresi and Matta (2021a), the aforementioned papers do not specifically focus on model generation including the modeling of assembly phases. This shortfall may be due to the problem of flattening data (van der Aalst, 2019), which is particularly acute for assembly processes, in which different material flows

Table 1: Features of this paper with respect to significant contributions in the literature.

Reference	**Application Field**	**Scope**	**Event Log**	**Blocking Conditions**	**Real Case Study**	**Fitness Testing**
Ng et al.	Manufacturing	Model Generation	-	-	-	-
Limère et al. (2013)	Manufacturing	Assembly Kitting Optimization	-	-	-	-
Yang et al.	Telecommunications	Model Generation	OCPM	-	•	-
Krenczyk et al. (2017)	Manufacturing	Assembly Line Balancing	Traditional	-	-	-
Denno et al. (2018)	Manufacturing	Structure Discovery	Traditional	-	-	-
Xu et al. (2016)	Agnostic	Automated Data Logging	Unstructured	-	-	-
Knoll et al. (2019a)	Manufacturing	Value Stream Mapping	OCPM	-	-	-
Dong et al. (2019)	Manufacturing	Resource Modeling	Traditional	-	-	-
Knoll et al. (2019b)	Manufacturing	Data Modeling	-	-	-	-
Knoch et al. (2020)	Manufacturing	Structure Discovery	Unstructured	-	•	-
Rashid and Louis (2020)	Construction	Structure Discovery	Traditional	-	-	-
Petschnigg et al. (2020)	Manufacturing	Model Generation	Unstructured	-	•	-
Schuh et al. (2020)	Manufacturing	Data Modeling	Unstructured	-	-	-
Lugaresi and Matta (2021a)	Manufacturing	Model Generation	OCPM	•	-	-
van der Aalst (2021)	Agnostic	Merging Graph Models	Traditional	-	-	-
Sjarov et al.	Manufacturing	Digital Twin Modeling	-	-	•	-
Brockhoff et al. (2022)	Manufacturing	Discovery of Bills of Materials	Traditional	-	•	-
this paper	Manufacturing	Model Generation	OCPM	•	•	•

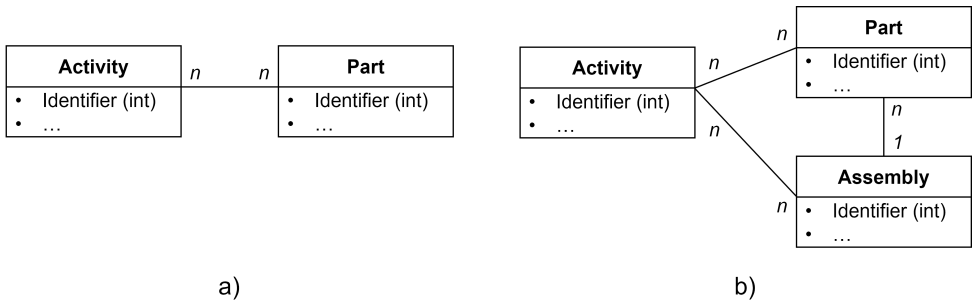

Fig. 1: Entity relationship diagrams between manufacturing activities and parts identifiers: a) traditional process mining, b) object-centric process mining.

converge toward assembly stations where the part identifiers are likely to change. For instance, in the automotive market it is common to assign an identifier to a car frame, and other dedicated IDs to its sub-components, such as the doors. When the car is assembled, it may have either a new identifier or the same ID of one of the sub-components. All events in the production system will refer to the assembled product, which is linked to multiple sub-components IDs. This means that there are multiple ways to flatten the data, leading to different views that are not connected. As a result, it becomes difficult to obtain an overview of the real system structure from a discovery approach, since the event data must be extracted multiple times for the each requested view.

2.3 Research Gap

van der Aalst (2019) discussed the discrepancy between actual event data and the flattened event logs used by traditional process mining techniques. To address this, the author proposed a new mining approach called Object-Centric Process Mining (OCPM) and a specific logging format. The object-centric event log is a collection of events related to objects of different types (e.g., tools, packages, shipments). The paper also presents notations and a baseline discovery approach to help facilitate OCPM comprehension and usage. Figure 1 graphically explains the difference between the two mining views. In traditional mining approaches, only a single part identifier is allowed, while OCPM allows to use several, thereby representing objects that can be linked to other items with many-to-many relationships. Hence, OCPM can be applied effectively to describe production systems with assembly operations.

An OCPM representation of a production system dataset allows to represent the relations among different object types that flow in a manufacturing system. However, the sole recognition of the objects is not sufficient for identifying the specific locations where they interact one another. With reference to assembly operations, OCPM can recognise and represent the relationships among products and their components, while it cannot associate the assembly locations. As a consequence, the positions where assembly operations occur can only be assumed or discarded from the digital model, unless further manual inspections and modifications are affordable. The consequent limitation is evident in the generated

model. The simulation model must take into account the assembly phase in order to accurately depict the system's behavior. This means that a specific condition must be present in the model, indicating that all required materials must be available at the assembly points for the assembly phase to be successful. A model generation technique that neglects this condition will produce a biased estimation of the real system performance[1]. For example, Figure 2 graphically shows two different modeling options using Petri Nets (Peterson, 1977). The transitions 1 and 2 indicate the last operations that must be done on the sub-components, while transition 3 expresses the assembly step. In Figure 2a, the assembly operation represented by transition 3 may be executed even if only one of the upstream steps has been completed. Differently, in Figure 2b, the assembly step is enabled only by the availability of all necessary components. It is worth to notice that assembly operations are not the only blocking conditions that may be discarded from a model representation. For instance, the operations in a manufacturing system may be linked to managerial processes that control production based on specific indicators or rules. Also, kanban manufacturing is notably dependent on the synergy between material and information flows.

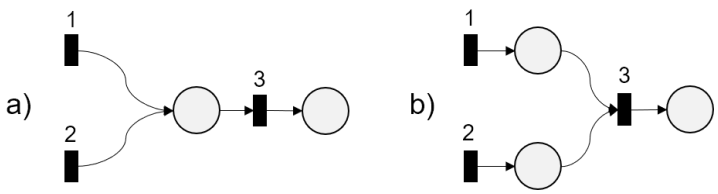

Fig. 2: Example of assembly process modeled by a portion of a Petri Net: a) improper modeling results in a non-blocking condition, b) the blocking condition representing assembly steps is properly modeled: all the previous production steps must be completed before assembly. The figure has been taken from Lugaresi and Matta (2021a).

Table 1 shows that most available approaches do not explicitly consider blocking conditions such as the one in Figure 2. The sole work that does is Lugaresi and Matta (2021a), whilst relying on strong assumptions on the graph models. Differently from the latter and previous approaches, this paper aims to formalize and generalize the system discovery and model generation phases. The main novelty elements of this work are the following: (1) it proposes a generalized procedure for the generation of complete graph models of systems with non-linear material flows, with no assumptions on the assembly phase locations; (2) it introduces indicators to evaluate the fitness of the generated solutions. Also, the proposed approach in this work is applied to a real manufacturing system case for testing in realistic industrial conditions.

[1] Evidence of the biased estimation of the real system performance is provided in Table 8.

3 Problem Description

In this section is described the problem of discovering manufacturing systems with non-linear material flows. Without loss of generality, this work takes as reference the material flows that determine systems with assembly operations.

Let us consider as illustrative example the manufacturing system depicted in Figure 3. The system consists of stations 1 and 2 that produce components of type A, while stations 3 and 4 are dedicated to components of type B. A conveyour brings parts of type A and B to an assembly station 5. The parts are assembled into products of type C on station 5 and sent to station 6 for packaging. Each item is traceable through a unique identifier (e.g., bar code, quick response code). Each station is equipped with sensors and contributes to the creation of an *event log*. Event logs are datasets that include data about items going through the system, such as serial numbers associated with the items and the time stamps of their activities, as well as various details such as resource identifiers and results of quality controls. Process mining techniques typically assume to start from an event log with at least three information types: the activity identifier $n \in \mathbb{N}$, the work-piece identifier $i \in \mathbb{I}$, and the timestamps $t_S(n, i)$ and $t_F(n, i)$ indicating the moment at which the n-th activity has been started and finished by the i-th work-piece, respectively. It is reasonable to assume that such data types are available and can be gathered from modern manufacturing systems.

3.1 Object-centric Event Logs

An OCPM-compliant representation of a production system dataset is feasible, provided the availability of relational tables connecting the components to the assembled products, which are necessary to define the object relationships. In production environments, such information is typically retrieved from the Bill of Materials (BOM). The BOM includes the component-product relationships among all part types. For instance, the BOM of the products produced in the system in Figure 3 can be written as $\{C : [A, B]\}$. Table 2 shows an extract of the object-centric log in which components of type A and B are assembled into a part type D. The components are coded 1 and 2, respectively, while the assembled part is coded with 3. In the object-centric representation, the part identifier column has been substituted by two object columns: components, and assembled products.

Note on Bill of Materials Representations. The BOMs are tables widely used and commonly available within tools such as Product Life-Cycle Management or Enterprise Resource Planning. A BOM can be represented by a tree model, in which each part type is represented by a node, and it is connected with an arc to the nodes of its component part types. Two main cases are introduced and depicted in Figure 4:

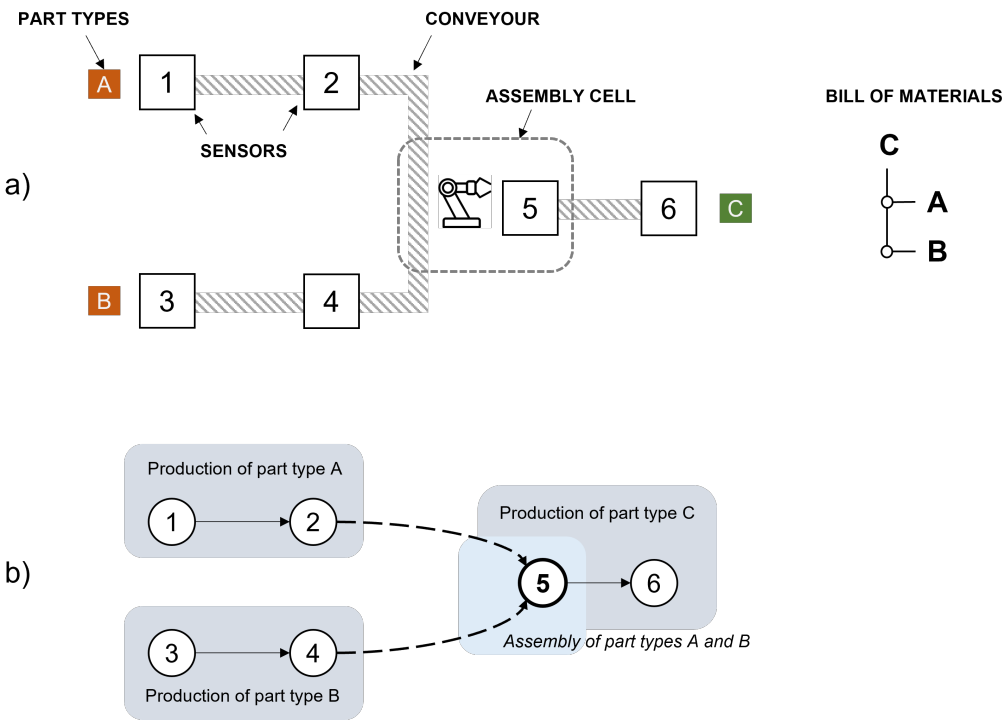

Fig. 3: Illustrative example – a) Logical schema of a flow shop manufacturing system with assembly operations; b) Graph model of the flow shop system.

Table 2: Example of object-centric event log produced by the system in Figure 3.

		Activities		**Involved Objects**	
Event	**Time-stamp**	***Name***	***Type***	***Components***	***Assembly***
1	0.00	1	start	1	-
2	0.00	3	start	2	-
3	0.35	1	finish	1	-
4	0.35	2	start	1	-
5	0.45	3	finish	2	-
6	0.45	4	start	2	-
7	0.51	4	finish	2	-
8	0.62	2	finish	1	-
9	0.62	5	start	{1,2}	3
10	0.76	5	finish	{1,2}	3
11	0.76	6	start	{1,2}	3
12	0.88	6	finish	{1,2}	3

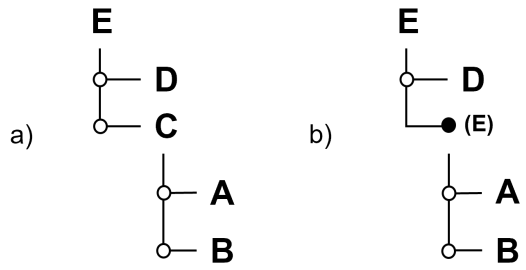

Fig. 4: Bill of Materials representations: a) type 1: full traceability of product types, b) type 2: partial part type traceability.

- *BOM Type 1: Full part type traceability.* In the first case, each node of the BOM identifies a distinct part. Hence, each assembly step identifies a new part type. In Figure 4a, part types A and B are assembled in a component of type C, which is then used together with part type D for the production of the final product, E.
- *BOM Type 2: Partial part type traceability.* In the second case, multiple assembly operations may be performed on a part without changing its part type. Such practice is typical for smaller components or consumables. As example in Figure 4b, part types A and B are assembled in a work-in-progress component which is already tagged the same way as the final product, i.e. part type E. A further assembly stage simply adds the sub-component of type D.

It is worth to notice that the second BOM representation is less formally valid. However, in the industrial practice it often happens to encounter such situations. This can happen for several reasons, and is easy to be observed among Small and Medium Enterprises. The method proposed in this paper is compatible with both types of BOM representations as support to define the objects relationships in an event log.

3.2 Digital Model Generation

A simulation model structure can be represented by the collection of relational properties (i.e., precedence relationships) in a directed graph composed by nodes that indicate the activities and arcs that represent the material flows. A graph model is defined as a tuple $\Omega = (\mathbb{N}, \mathbb{E})$ where $\mathbb{N}$ is the set of nodes and $\mathbb{E} \subseteq \mathbb{N} \times \mathbb{N}$ is the set of arcs. For example, the graph model of the system in Figure 3 is defined by the set of nodes $\mathbb{N} = \{1, 2, 3, 4, 5, 6\}$ and the set of arcs $\mathbb{E} = \{(1, 2), (3, 4), (2, 5), (4, 5), (5, 6)\}$. For convenience, a matrix $\Gamma = \{\gamma_{ij}\}$ can be defined, where each element γ_{ij} is 1 if there is an arc between nodes i and j, 0 otherwise. The model generation procedure from (Lugaresi and Matta, 2021b) produces the graph model $\Omega_0 = (\mathbb{N} = \{1, 2, 3, 4, 5, 6\}, \mathbb{E} = \{(1, 2), (3, 4), (5, 6)\})$. The result is a model with sectors treated as separate graph models. The graph model is not complete. The same result is obtained with traditional process mining software tools (e.g., Disco, ProM). This is because the model generation technique is based on the assumption of a single part identifier. Differently, assembled parts and components have distinct

identifiers, resulting in the omission of assembly relationships. Hence, after the system discovery step has been completed, the graph model must be enriched with elements accounting for assembly operations. The problem consists in the addition of arcs to the graph model obtained by traditional mining steps, Ω_0. In the following, such addition is called *Graph Model Completion*. Any node in the graph model Ω_0 can be an assembly node, and each added arc is dedicated to a product type that is produced in the system. Let us define z_{ijp} as the boolean variable that defines if the directed arc (i, j) is added to the graph model for representing the assembly of product type p. The *Graph Model Completion* corresponds with the addition of elements to the Γ matrix. Namely, the complete graph model can be defined by a Γ' matrix, in which each element γ'_{ij} is defined as follows:

$$\gamma'_{ij} = I(\gamma_{ij} + \sum_{p} z_{ijp}) \tag{1}$$

where $I(x)$ is 1 if $x > 0$, 0 otherwise.

For instance, referring to the illustrative example of Figure 3, $\gamma_{1,2} = \gamma_{3,4} = \gamma_{5,6} = 1$, and $z_{2,5,C} = z_{4,5,C} = 1$. Hence, the Γ' matrix will include the additional elements $\gamma'_{2,5} = \gamma'_{4,5} = 1$. Thanks to such addition, assembly nodes (i.e., $n = 5$) can be recognized as such, and the correct modelling of the blocking conditions is enabled.

Figure 5 summarizes the main steps of the model generation framework. The goal is to generate a complete digital model of a manufacturing system. The information system of a production environment is exploited to gather information, namely an event-log, the BOM, and additional information such as work-force availability and maintenance policies. The event log is exploited in a graph model generation procedure as in (Lugaresi and Matta, 2021b). Then, the BOM is exploited to complete the graph model with the assembly-related arcs. Further adjustments can be done to model parameters taking into account additional information (e.g., statistical distribution fitting, maintenance policies, availability models). Indeed, since a digital twin should follow the evolution of the system along its life cycle, it is necessary to keep updating the parameters to guarantee the physical-digital alignment (e.g., buffer sizes, processing time distributions). The adjustments can occur in several ways: among others, a manual or automated adjustment of a known parameter (e.g., spindle speed), an estimation based on an available dataset (i.e., distribution fitting), or an estimation based on expert knowledge. Also, the graph model may be tuned toward the users' specifications (Lugaresi and Matta, 2021b). Finally, the graph model can be converted into a simulation model through a specific formalism such as Petri Nets (Peterson, 1977) or Event Relationship Graphs (Schruben, 1983). Essentially, model conversion means translating the graph models into simulation code (Peñarroya et al., 2006). Each element in the model finds a correspondent in a executable language (e.g., a node becomes a station object instance). Further information on model conversion is available in related literature (Passarin and Verucchi, 2022). Model tuning, parameter estimation, and

model conversion are out of the scope of this paper. The remainder of this paper concentrates on the first two steps of the model generation framework, with a focus on graph model completion.

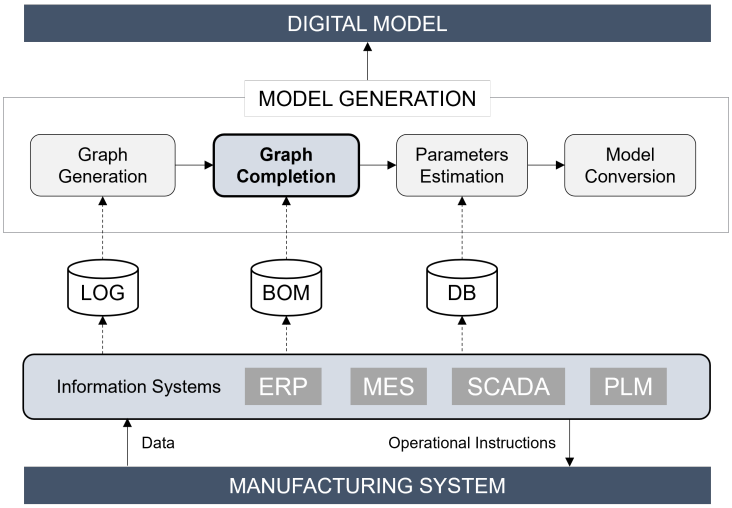

Fig. 5: General overview of the model generation approach and the scope of this paper (LOG: Event Log, BOM: Bill of Materials, DB: Additional Databases).

4 Proposed Model Generation Method

The first step of a model generation procedure consist in creating a graph model from the available event log. Then, in order to take into account for non-linear material flows, the graph model is enriched with the arcs that represent multiple objects relationships. Once a graph model is created, nodes and arcs can be enriched with properties of the system. For instance, the finite capacity of the conveyor between two generic activities i and j can be indicated as a property of the arc (i, j).

4.1 Generation of a Graph Model

Model generation links an event log with its corresponding simulation model. This work takes as reference the model generation procedure that has been developed in (Lugaresi and Matta, 2021b). Firstly, a set of unique identifiers of activities is created and indicated by $\mathbb{N}$. The specific route that each part followed in the system is called *trace* and can be represented by a series of activity identifiers. Each i-th part has its corresponding trace $\theta_i = \{n^{(1)}, n^{(2)}, \ldots, n^{(N_i)}\}$, where N_i is the number of the activities performed by the part. The traces can also be used to fetch precedence relationships between activities. In summary, a node is created when a specific activity has been performed by at least one object in the system, and an arc indicates that at least one trace shows an activity has followed another. Namely, arc (m, n) exists if $\exists i \in \mathbb{I}, m, n \in \mathbb{N} | t_F(n, i) < t_S(m, i)$.

4.2 Non-linear Material Flows

The problem described in section 3 can be expressed as a matching of components and assembled parts. To this end, the temporal proximity of operations can be exploited as indicator of assembly locations. The general idea is that the added arcs have to be such that the temporal difference between the production of components and assembled products is minimized. Such assumption is suitable for a significant subset of manufacturing systems. For instance, flow lines in which relevant components are produced in a nearby machining area, or group technology manufacturing, in which a set of production cells coordinate to produce the work-in-progress that will converge downstream. A typical example is the automotive sector: doors are usually formed and welded in the same plant as the chassis, and converge to the main assembly line with very limited time differences (Uysal et al.). Meanwhile, other production systems may not be suitable for a time-proximity-based approach. This is the case for systems relying on batch production (e.g., foundry, moulding) or with several out-sourced activities. In such systems, the completion timestamps of components will be equal for all the components of the batch.

In order to base the Graph Completion Problem (GCP) on the temporal distance between each element production time, it is essential to maintain both parts and sub-components in the mathematical representation. Multiple combinations of parts and sub-components instances are possible, hence an assignment is necessary before considering the temporal distance. The next section elaborates on the mathematical formulation of the problem.

4.3 Mathematical Formulation

The GCP can be expressed through a mathematical programming formulation. In the following, d_{ca} is defined as the time difference between the instant in which the c-th component is produced and the moment it is assembled with the a-th product. The mean temporal proximity can be expressed by the Mean Square Error (MSE) which is used as objective function of the problem. $\mathbb{C}$ is defined as the set of components, $\mathbb{A}$ the set of assembled products, $\mathbb{S}$ the set of stations (i.e. the nodes of the original graph model), and $\mathbb{P}$ the set of part types.

Assumptions. It is assumed that a production system is supplied with data collection devices and it is possible to assemble an event log. Also, it is assumed that the log is clean from incomplete traces, and each trace belongs to a part that is either a component or an assembled product. Further, the BOM-related cardinalities are respected: for instance, if one product requires two components, then the corresponding traces must be in the log. Finally, it is considered the case in which no batching is present along the production activities, and that products and components are processed in sequential order, i.e. the initial sequence is identical to the final sequence.

Parameters. The following parameters can be derived by the joint pre-processing of event log and BOM:

- τ_c is the time instant at which component c has been produced: from the log, $\tau_c = \max_{n \in \mathbb{N}} t_F(n, c) \, \forall c \in \mathbb{C}$.
- t_{sa} is the time instant at which assembled product a is produced on station s, 0 otherwise. Namely, $t_{sa} = t_S(s, a) \, if \, \exists \, t_S(s, a) \, \forall a \in \mathbb{A}, \, s \in \mathbb{S}$.
- η_s is 1 if the s-th station is compatible with assembly, 0 otherwise. A station is compatible for assembly if it produces at least one part type from sub-components.
- ρ_{cp} is equal to 1 if the c-th component is of part type p, 0 otherwise.
- B_{ap} is an integer value representing the number of components of type p required for the fabrication of the assembled part a.
- M is a very large number.

Decision Variables. The decision variables of the problem are as follows:

- x_{cas} is 1 if the c-th component is assigned to assembled part a on station s, 0 otherwise.
- d_{ca} is the temporal distance between the production of component c and the a-th assembled product.

Graph Completion Problem (GCP):

$$\min \quad y \tag{2}$$

subject to:

$$y \geq \frac{\sum_a \sum_c d_{ca}^2}{|\mathbb{C}||\mathbb{A}|} \tag{3}$$

$$d_{ca} \geq \quad t_{sa} - \tau_c - (1 - x_{cas})M \qquad \forall c, a, s \tag{4}$$

$$d_{ca} \leq \quad t_{sa} - \tau_c + (1 - x_{cas})M \qquad \forall c, a, s \tag{5}$$

$$\sum_a \sum_s x_{cas} \leq \quad 1 \qquad \forall c \tag{6}$$

$$\sum_c \sum_s x_{cas} \rho_{cp} = B_{ap} \qquad \forall a, p \tag{7}$$

$$\sum_c \sum_a x_{cas} \leq \quad M \, \eta_s \qquad \forall s \tag{8}$$

$$\sum_c x_{cas} \leq \quad M \, t_{sa} \qquad \forall a, s \tag{9}$$

$$d_{ca} \in \mathbb{R}; x_{cas} \in \{0, 1\}. \tag{10}$$

The objective function (2) aims at the minimization of the Mean Square Error y, defined by the temporal distance between the components and assembled products, as stated in constraint (3). Constraints (4) and (5) indicate that the temporal distance d_{ca} is to be accounted only for the component-assembly pairs that are selected. The constraints are activated only for the component-product combination on the selected

station, which is indicated by $x_{cas} = 1$. Constraints (6) state that each component can be assigned to either one assembled product, on maximum one station. Constraints (7) state that each assembled product has to be assigned to the number of components corresponding to the BOM requirements. Constraints (8) guarantee that only stations compatible with assembly operations are selected. Constraints (9) ensure that assembly locations are identified in accordance to where the production is recorded in the log, hence that an assembled product is not assigned to unvisited stations. Constraints (10) indicate the nature of the decision variables. In total, GCP counts $|\mathbb{C}||\mathbb{A}|(1+|\mathbb{S}|)$ variables and $1+|\mathbb{C}|+2|\mathbb{A}|+|\mathbb{S}|(|\mathbb{C}||\mathbb{A}|+|\mathbb{A}|+1)$ constraints.

Retrieving the Graph Model. Once GCP is solved, the solution in terms of graph model can be retrieved with a post-processing step. Indeed, the corrections to the model defined by the variable z can be derived with the simple procedure listed in Algorithm 3 described in Appendix C.

5 Solution Procedure

This section proposes a solution procedure for the problem formulated in section 4.3. It is assumed that the graph model generation method described in (Lugaresi and Matta, 2021b) is applied once at the beginning of the procedure. The result is a graph model Ω. Given that the size of the problem depends on $|\mathbb{C}|$, $|\mathbb{S}|$, and $|\mathbb{A}|$, which are generally not large in a production system, the proposed solution method is based on the complete enumeration of feasible assembly stations in the existing graph model. Then, the selection of the specific assembly locations is done following the idea of temporal proximity. Figure 7 summarizes the procedure steps.

5.1 Step 1: Define the BOM Levels.

It is assumed to analyze a subset of nodes and arcs such that one level of BOM is explored at a time. Indeed, a generic BOM $\mathbb{B}$ can be separated in a collection of levels, $\mathbb{B} = \{\mathbb{B}_1, \ldots, \mathbb{B}_i\}$. For instance, the BOM of the products produced in the system of Figure 10 has two levels: the first one is $\mathbb{B}_1 = \{D : [C, D]\}$, while the second is $\mathbb{B}_2 = \{D : [A, B]\}$. Hence, the problem is separated in two parts, so that only one level of BOM is present in each part. For each level, the GCP is solved. Referring to Figure 10, the GCP is firstly solved for the system composed of stations $s \in \{3, 4, 5\}$. Then, the system composed by stations $s \in \{1, 2, 3, 5\}$ is analyzed. In summary, Step 1 is dedicated to the identification and separation of the BOM levels. The remaining steps take as input one BOM level at a time, and one GCP is solved for each level.

5.2 Step 2: Define the Set of Candidate Stations.

In this step, a subset of stations $\mathbb{S}_p \subseteq \mathbb{S}$ is identified. Each node in the graph model Ω is a station $s \in \mathbb{S}$. By exploiting the information from the BOM, it is possible to identify the stations that produce parts

with sub-components. All such stations are candidate assembly locations. Let B_i be the i-th level of BOM selected at Step 1. Each station $s \in \mathbb{S}_p$ is a candidate station if it has produced an assembly product of type p included in B_i, namely $\exists t_S(s,a)|p_a \in B_i$. Figure 6 represents the logic of this step, which is performed by Algorithm 1 in Appendix B (complexity: $O(n)$).

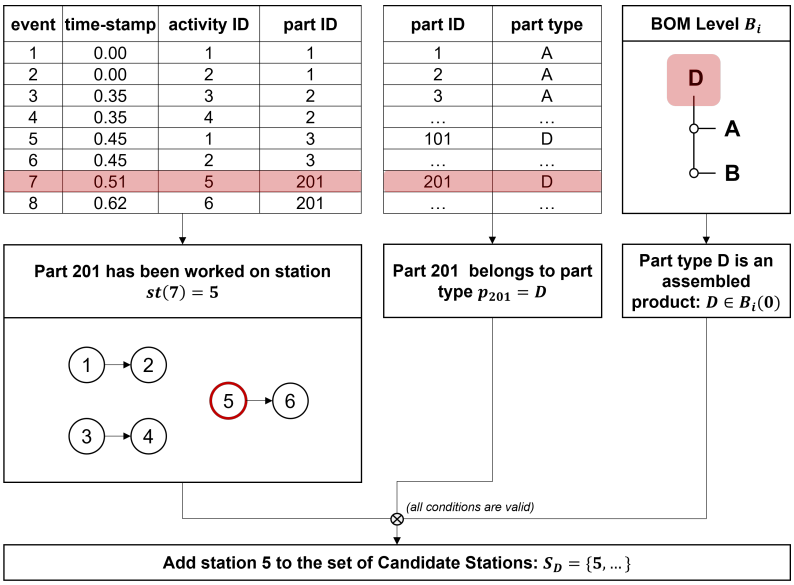

Fig. 6: Selection of candidate stations (step 2).

5.3 Step 3: Define the Set of Combinations.

In this step, the possible combinations of assembly stations are identified. Starting from the results of model generation, the obtained graph model can be divided in a collection of G disjunct subgraphs $\Omega = \{\Omega_1, \dots, \Omega_G\}$. Since different product types may be produced on different stations, there is no guarantee that they will be produced on nodes from the same disjunct graph model. Further, for each product type, only one node is a candidate assembly station. As a result, multiple combinations of candidate stations are identified. Let us define $\mathbb{V}$ a collection of tuples. Each tuple corresponds to a possible combination of assembly stations for the considered part types. For instance, referring to the system in Figure 3, the subcomponents of product type C can either be assembled on station 5 or station 6 Hence, $\mathbb{V} = \{(5), (6)\}$. Step 3 is performed by Algorithm 2 in Appendix B (complexity: $O(n)$).

5.4 Step 4: Assignment Sub-Problem.

The remaining part of the problem regards the assignment of the components to the corresponding assemblies. Time proximity is used as indicator of the best matchings. Notice that this problem corresponds

to a *job assignment problem* (Pentico, 2007), in which the cost matrix is determined by the square time differences between the production time of components and assemblies. Namely, for each combination $v \in \mathbb{V}$, $\delta_{ca}^{(v)}$ is defined as the difference of the timestamps of the a-th assembled product and the c-th component:

$$\delta_{ca}^{(v)} = (t_{s_v a} - \tau_c)^2 \quad \forall c, a \tag{11}$$

Differently from d_{ca}, which depends on the assignment given by x_{cas}, $\delta_{ca}^{(v)}$ considers the temporal distance with the product assembled on the station from the specific combination v. Hence, in step 4, the following problem is solved for each combination of assembly stations v:

Component-Assembly Assignment Problem (CAAP):

$$\min \quad y \tag{12}$$

subject to:

$$y \geq \frac{\sum_a \sum_c (\delta_{ca}^{(v)})^2 x_{cas_v}^{(v)}}{|\mathbb{C}||\mathbb{A}|} \tag{13}$$

constraints (6), (7), (10).

The objective function (12) represents the minimization of the time proximity between the components and assemblies through the Mean Square Error y, which is defined in constraint (13). The remaining constraints are (6), (7), and (10). The CAAP is a sub-problem of the GCP in which the stations' assignments have been established.

5.5 Step 5: Identify and Evaluate the Solution.

Finally, among the $|\mathbb{V}|$ combinations, the algorithm selects the one that guarantees the lowest MSE, indicated by v^*. The corresponding solution of the assignment problem can be used to retrieve the GCP solution. Starting from the selected solution, the corresponding nodes and arcs can be added to the graph model. The procedure to complete the graph model starting from a GCP solution x_{cas}^* is listed in Algorithm 3 in Appendix C (complexity: $O(n)$).

5.5.1 Assignment Scores Once a solution is obtained by the procedure, it is of interest to understand how the solution performs with respect to the underlying system. The algorithm provides information on both the component-product assignment and the location of the assembly operation. These assignments may be subject to deviations, due to several reasons: the assignments are driven by time differences and small deviations may cause a different assignment from the real one (e.g., due to a data collection error), the components registered in the log may be sent to other sectors of the system that are not traced, the

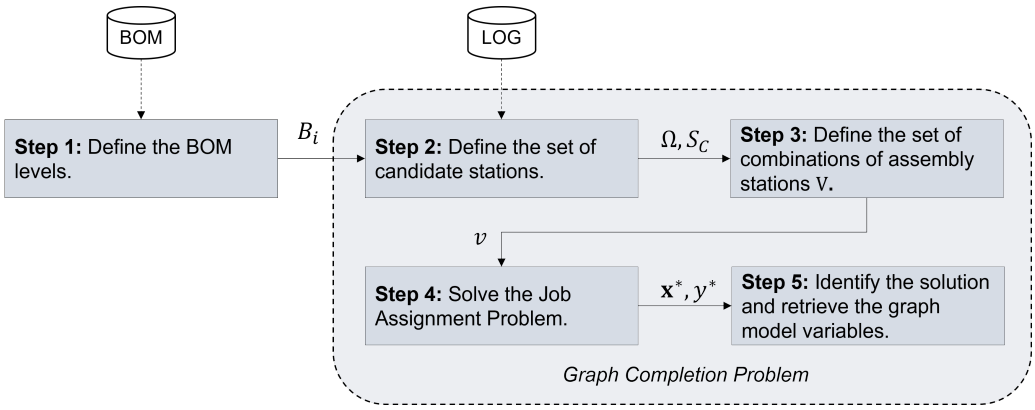

Fig. 7: Proposed procedure for solving the Graph Completion Problem.

real production order may not be sequential for all the components. Hence, it is important to evaluate the capability to provide good assignments and to spot the correct assembly locations. Let us define x^*_{cas} as the solution of the algorithm, and w_{cas} the matrix representing the correct assignments in the system.

The **assignment score** α is an indicator of the goodness of the assignments on the a-th assembled product, as follows:

$$\alpha_a = \frac{\sum_c (\sum_s |x^*_{cas} - w_{cas}|)}{|\mathbb{C}|}. \tag{14}$$

For each assembled product, α_a assumes values in the interval $[0, 1]$, with 1 representing the completely correct assignments. Additionally, $\bar{\alpha} \in [0, 1]$ is defined as the **average assignment score**:

$$\bar{\alpha} = \frac{\sum_a \alpha_a}{|\mathbb{A}|}. \tag{15}$$

λ_c is the **location assignment indicator** on the c-th assembled product, measuring how well the method has assigned components on the assembly stations:

$$\lambda_c = 1 - \sum_s \frac{|\sum_a x^*_{cas} - w_{cas}|}{|\mathbb{A}||\mathbb{S}|}. \tag{16}$$

For each component, λ_c assumes values in the interval $[0, 1]$, with 1 representing the completely correct location assignment. Similarly to α, $\bar{\lambda} \in [0, 1]$ is defined as the **average location score**:

$$\bar{\lambda} = \frac{\sum_c \lambda_c}{|\mathbb{C}|}. \tag{17}$$

Also the global behavior of the obtained solution can be assessed. This can be done with a **reproducibility score**, defined as:

$$\phi = \frac{\sum_c \sum_a \sum_s |x^*_{cas} - w_{cas}|}{|\mathbb{C}||\mathbb{A}||\mathbb{S}|}. \tag{18}$$

The score ϕ assumes values in the interval $[0, 1]$, with 1 representing a completely correct assignment, both in terms of assembly station and component-assembly assignment.

6 Numerical Experiments

The solution procedure described in section 5 has been applied in three test cases. Each test case has the goal to test a different aspect of the GCP. Table 3 summarizes the characteristics of the experiments that are listed as follows:

- **Test Case 1.** A multi-stage production system with five part types, namely two assembled products and three components (single-level BOM of type 1). The scope of this experiment is to verify that the proposed approach can correctly be executed in a system with multiple assembled parts.
- **Test Case 2.** A multi-stage production system with four part types, with a multi-level BOM of type 2. The scope is to show the behavior of the proposed method with respect to a multi stage production system, i.e. in which the BOM is composed by multiple levels.
- **Test Case 3.** A real production system with assembly operations within the manufacturing of tier-1 automotive components (single-level BOM of type 1). The scope of this test is twofold: (1) to verify the applicability of the proposed approach in a realistic scenario, and (2) the quantitative observation of the performance obtained with a complete model, to understand the difference with a model generated with the standard approach.

For all the experiments, the solution procedure (section 5) has been implemented in ILOG CPLEX v12.6 and it is solved using a PC equipped with an i7-6600U CPU at 2.6 GHz and 16 GB memory.

Table 3: Characteristics of the test cases selected for the experiments.

Case	BOM Levels	Product Types	Components Number	Processing Time Distribution
1	Single	Multiple	2000	Exponential
2	Multiple	Single	3000	Exponential
3	Single	Single	3000	Fitted from data

In the following are described the test cases, the experimental settings, and the numerical results.

6.1 Test Case 1: Flow Shop

The first test case is a flow shop which produces two final product types, D and E. The system is depicted in Figure 8.

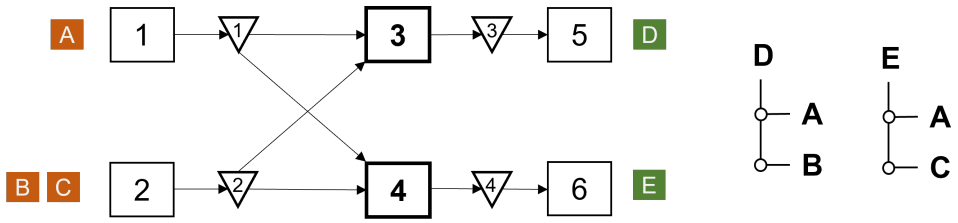

Fig. 8: Test Case 1 – Flow shop manufacturing system. Squares represent stations and triangles represent inter-operational buffers. Stations 2 and 4 produce sub-components, while stations 5 and 7 assemble them into products of type D and E, respectively.

6.1.1 Production System. The system is composed by six stations. Stations 1 and 2 produce sub-components of type A, B, and C. Station 3 assembles parts of type A and B into products of type D, while station 4 assembles B and C into E. Each station s has a downstream buffer of size H_s and the buffer capacities are equal for all stations: $H_s = 10\,\forall s$, except from stations 5 and 6 which produce products D and E as soon as the needed sub-components are available in the corresponding upstream buffers. Both inter-arrival times and processing times are distributed according to an exponential distribution with mean 1 min. The choice of such distribution is due to its ability to describe variability (i.e., high variance) which characterizes a significant set of production environment.

6.1.2 Experimental Setting. The manufacturing system has been modeled in Arena Simulation Software. Five event logs have been generated, each corresponding to an independent replication. Each replication represents the production of 1000 final products, 600 of type D and 400 or type E. The resulting event log contains 7168 events. Since the BOM contains a single level for both part types, one GCP has to be solved over the entire set of nodes.

6.1.3 Results. Given that two part types are being produced, the set of possible combinations of assembly stations is given by the permutations of 2 elements from the sets $\mathbb{S}_D = \{3, 5\}$ and $\mathbb{S}_E = \{4, 6\}$. Hence, for this case the set of candidate assembly stations are the tuples $\mathbb{V} = \{(3, 4), (3, 6), (5, 4), (5, 6)\}$. For each replication, the GCP solution method described in section 5 has been used to generate a graph model. Table 4 summarizes the results obtained for the first test case. It can be noticed that the correct combination of assembly stations $v^* = (3, 4)$ has been selected in each replication as the one guaranteeing the minimum MSE value. Figure 9 shows the graph model obtained with the proposed procedure. The arcs $(1, 3)$ and $(2, 3)$ have been added to represent the production of part type D, while the arcs $(1, 4)$ and $(2, 4)$ model the assembly of part type E. The obtained scores are: average assignment score $\bar{\alpha} = 0.949 \pm 0.0192$, average location score: $\bar{\lambda} = 1.0 \pm 0.0$, average reproducibility score: $\bar{\phi} = 0.931 \pm 0.0196$.

Note on computation time. The requirements in terms of computation time have been tested by executing the first test case while varying the problem dimension in terms of input components, i.e.

Table 4: Test Case 1 - Objective function values obtained for each candidate combination of assembly stations v.

	Combinations v				Assignment Scores		
Replication	(3,4)	(3,6)	(5,4)	(5,6)	α	λ	ϕ
1	**10.706**	25.327	13.721	29.690	0.939	1	0.911
2	**18.299**	46.527	21.413	51.048	0.968	1	0.951
3	**17.242**	45.971	19.646	49.371	0.945	1	0.937
4	**10.316**	32.593	11.865	35.303	0.932	1	0.919
5	**14.398**	33.623	16.816	37.206	0.963	1	0.935
Mean	**14.192**	36.808	16.692	40.524	0.949	1	0.931

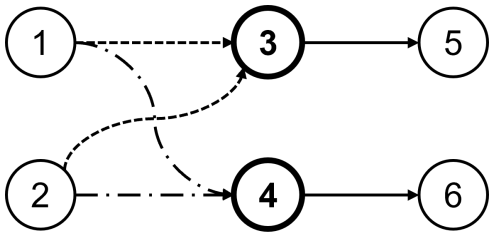

Fig. 9: Test Case 1 – Graph model obtained by the proposed approach: the dashed lines represent the added arcs as solution of the GCP.

$|\mathbb{C}| \in \{10, 100, 500, 1000, 2000, 5000\}$. Table 5 shows the behaviour of the average computation time, which is exponentially increasing and it exceeds five minutes with 2000 components.

Table 5: Test Case 1 – Average computation time with respect to the number of input components (5 replications).

Nr. of components	10	100	500	1000	2000	5000
Nr. of traces	34	282	1390	2792	5589	13989
Average Computation Time [s]	7	8	26	122	342	1380

6.2 Test Case 2: Multi-Level BOM

In this case, a production system with a two-level BOM is analysed.

6.2.1 Production System. Figure 10 shows the structure of production system under study. Station 1 and 2 produce components of type A and B, respectively. Such components are assembled in station 3 into product type D. On station 5, the component type C is assembled on product D. Hence, the Bill of Materials is of type 2 and it consists in two separate levels. The first level consists in the assembly of D

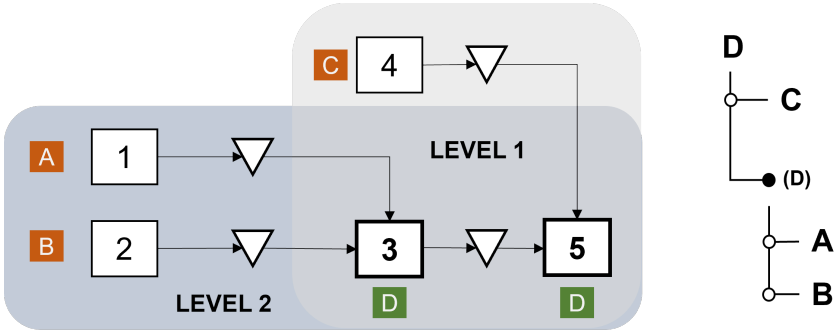

Fig. 10: Test Case 2 – Production system and BOM structure. Squares represent stations and triangles represent inter-operational buffers. Stations 3 and 5 are the assembly stations.

and C, while the second one describes the assembly of components A and B. Each station has a processing time p_s which is distributed according to an exponential distribution with mean 1 min. Inter-operational buffers have 10 slots each.

6.2.2 Experimental Setting. The goal of this test case is to show the behavior of the developed approach in a multi-level system. For the experiments, five event logs have been generated with a discrete-event simulation model in Arena Simulation Software, each representing the production of 1000 components for each part type. The resulting event log contains 9169 events. In the following are listed the results of the first replication.

6.2.3 Results. As first step, subsets of stations are selected based on the levels of BOM. Accordingly, the set of stations for the first level is $\mathbb{N}_1 = \{3, 4, 5\}$, and for the second level, $\mathbb{N}_2 = \{1, 2, 3, 5\}$. Each node set is used in the remaining steps of the method.

Level 1. The set of candidate stations is the subset of $\mathbb{N}_1$ in which the assembled components are produced, hence $\mathbb{S}_1 = \{3, 5\}$. Given that the system produces a single product, both stations are candidate. Hence, the set of possible combinations is $\mathbb{V} = \{(3), (5)\}$. For each combination, the assignment problem defined in section 5.4 has been solved. For both sub-problems, a total of 1792 components and 792 assembled products have been assigned one another. The objective function is $y_3 = 0.00381$ and $y_5 = 0.00323$. Accordingly, station 5 is selected as assembly station. Figure 11a shows the resulting graph model. The computation times are 63.14 s and 58.23 s, respectively. Further, the scores defined in section 5.4 have been calculated: average assignment score: $\bar{\alpha} = 0.99833 \pm 0.000741$, average location score: $\bar{\lambda} = 1.0 \pm 0.0$, average reproducibility score: $\bar{\phi} = 0.9995 \pm 0.0001141$.

Level 2. In the second level of the BOM, the components are types A and B and the assembled product is D. The subset of $\mathbb{N}_2$ in which D is produced is $\mathbb{S}_2 = \{3, 5\}$. Again, both stations are candidate assembly locations, and the set of possible combinations is $\mathbb{V} = \{(3), (5)\}$. For both sub-problems, a total of 2000 components and 792 assembled products have been assigned. The objective function is $y_3 = 0.02101$ and

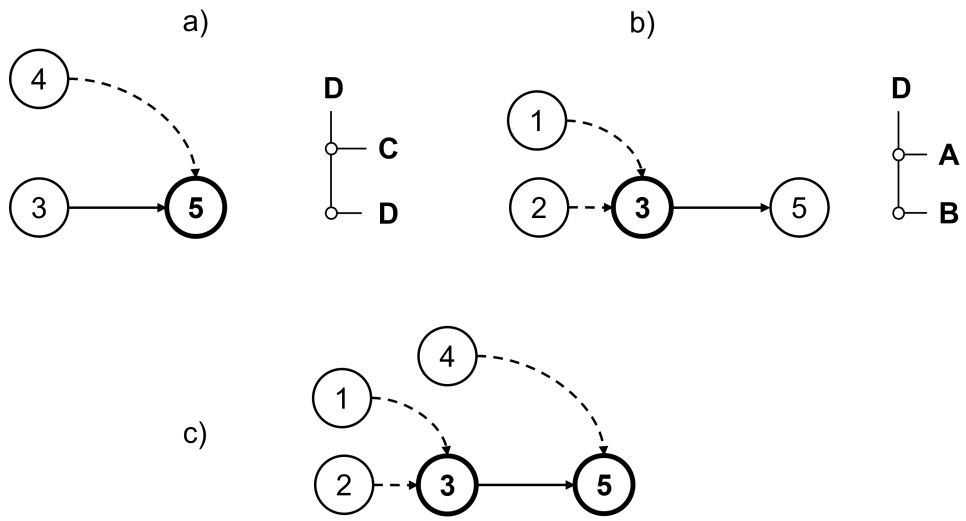

Fig. 11: Test Case 2 – Resulting graph models depending on the levels of the BOM: a) level 1, b) level 2, c) complete model. The dashed lines represent the added arcs as solution of the GCP.

$y_5 = 0.02957$. Hence, station 3 is selected as candidate assembly station. Figure 11b shows the resulting graph model. The computation times are $65.03\,s$ and $78.29\,s$, respectively. Further, the scores defined in section 5.4 have been calculated: average assignment score: $\bar{\alpha} = 0.99802 \pm 0.000913$, average location score: $\bar{\lambda} = 1.0 \pm 0.0$, average reproducibility score: $\bar{\phi} = 0.928 \pm 0.000489$.

The final complete graph model is visible in Figure 11c.

6.3 Test Case 3: Real Production System

The problem to be investigated is a multi-cell production system from a tier-1 supplier of road vehicle injectors. This system has been analysed within an industrial project, hence a validated discrete event simulation model was available and has been used to generate the input data.

6.3.1 Production System. The system consists of eight stations, as shown in Figure 12. Stations 1, 2, and 3 are dedicated to the production of sub-components, which are placed in buffers 1 and 3. The remaining stations produce the main part of the injector, and in station 7 the sub-components are assembled. Table 6 presents the detailed information about each manufacturing cell (process information is not disclosed for confidentiality reasons). The distributions of the processing times are fitted from collected field data or outputs of a more detailed and validated simulation model of each cell in isolation. Let us denote with H_s the buffer capacity of the downstream product store of each cell. The manufacturing system works with a pull production planning using kanban, hence the buffer capacities depend on the specific number of production cards issued in the shop floor. For the sake of simplicity, herewith this

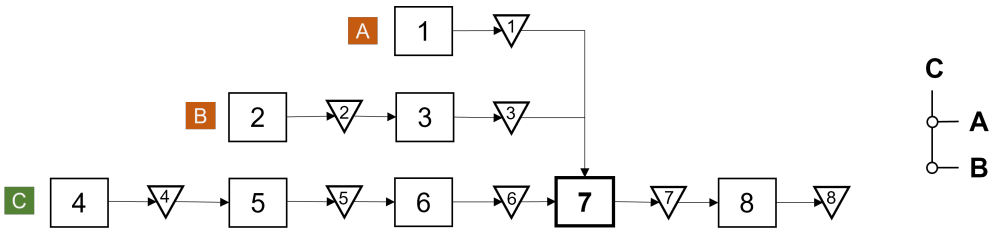

Fig. 12: Test Case 3 – Automotive tier-1 supplier production system. Squares represent stations and triangles represent inter-operational buffers. Station 7 is the assembly station.

phenomenon is not considered and it is assumed $H_s = 10 \, \forall s$; injectors can be produced in station 7 whenever all required sub-components are available in the corresponding upstream buffers.

Table 6: Test Case 3 – Parameters of the manufacturing system used for the experiments (u is a random number between 0 and 1). The processing times refer to the production of 1000 work-pieces.

Station s	Buffer H_s	Processing Time r_s [s]
1	10	$Logn(4.96, 0.52)$
2	10	$195 + Tria(30, 45, 90) + Logn(4.58, 0.542)$
3	10	$100 + Gamma(0.89, 111.91) + 240$
4	10	$156 + Gamma(1.13, 34.21)$
5	10	$Tria(80, 140, 500)$
6	10	$42 + Logn(1.57, 0.542), if\ u \leq 0.53$
		$62 + Logn(2.10, 0.632), if\ u > 0.53$
7	10	$170 + Gamma(5.94, 6.48)$
8	10	$196 + Gamma(5.75, 5.86), if\ u \leq 0.93$
		$277 + Gamma(0.94, 22.47), if\ u > 0.93$

6.3.2 Experimental Setting. The production system has been modeled in Arena Simulation Software. Five event logs have been generated, each corresponding to an independent replication in which 1000 injectors are produced. For each replication, the procedure described in section 5 has been applied. Then, the result has been used to build a simulation model. For the sake of simplicity, the graph model has been converted manually into a simulation model in Arena. The obtained model has been used to generate five independent event logs, and the corresponding performance in terms of throughput and system time has been calculated.

6.3.3 Results. Table 7 lists the objective function values obtained for each replication. Notice that in this case, since a single product type is produced, the assembly station combinations $\mathbb{V}$ are represented

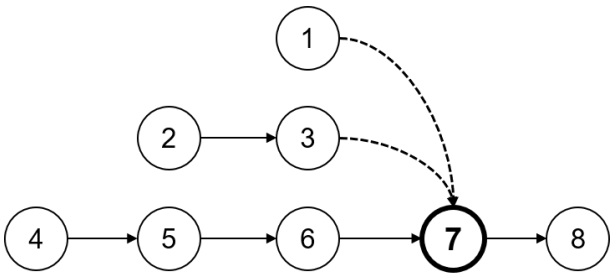

Fig. 13: Test Case 3 – Graph model obtained by the proposed approach. The dashed lines represent the added arcs as solution of the GCP.

by one-element tuples. Also in this case, the results table shows that in each replication the minimum objective function value corresponds to components-assembly associations on the correct station: $s = 7$. Figure 13 represents the graph model that has been obtained. Station 7 is identified as the assembly location and arcs are added to the model accordingly. The obtained scores are: average assignment score $\bar{\alpha} = 0.884 \pm 0.0307$, average location score: $\bar{\lambda} = 1.0 \pm 0.0$, average reproducibility score: $\bar{\phi} = 0.860 \pm 0.0206$. Such result allows to model correctly the assembly process on station 7, hence enabling the automated generation of the digital replica of the system of Figure 12.

Table 7: Test Case 3 - Objective function values obtained for each candidate assembly station.

	Combinations v					Assignment Scores		
Replication	(4)	(5)	(6)	(7)	(8)	α	λ	ϕ
1	0.227	0.164	0.054	**0.014**	0.016	0.890	1	0.872
2	0.085	0.074	0.038	**0.016**	0.017	0.844	1	0.843
3	0.051	0.050	0.049	**0.048**	0.053	0.910	1	0.877
4	0.151	0.114	0.049	**0.042**	0.046	0.879	1	0.841
5	0.042	0.043	0.042	**0.037**	0.039	0.895	1	0.865
Mean	0.111	0.089	0.046	**0.032**	0.034	0.884	1	0.860

6.3.4 Comparison with the Standard Approach. The obtained graph model has been validated by comparing the performance between two scenarios: (1) model with assembly operations obtained with the proposed method, (2) model obtained with a method that does not include non-linear flows (i.e., model generated with the method in (Lugaresi and Matta, 2021b)). Specifically, two performance indicators have been chosen for the comparison: IDT_7 is the vector of inter departure times from the assembly station 7: $IDT_7 = t_F(7, i) - t_F(7, i-1)$, and the system time, defined as the time a part of type C takes to flow in

the system: $ST = t_F(8, i) - t_S(4, i)$. Table 8 summarizes the results obtained by simulation experiments replicated 5 times each. The results show that the exclusion of assembly stations from the representation causes an over estimation of the system performance. The system time is significantly lower than the one of the real system. This is because the exclusion of the assembly blocking condition allows for several products to proceed and exit the system before their corresponding sub-components are available.

Notice that such difference depends on the particular system configuration and parameters.

Table 8: Test Case 3 - Performance validation depending on addition of assembly-related arcs out of 1000 samples (CI-HW = 95% Confidence Interval Half Width).

Blocking Conditions	Replication	IDT_7 (mean)	CI-HW	ST (mean)	CI-HW
original (with)	-	60.9	4.0	2249.5	24.9
	1	58.5	3.6	2114.1	21.3
	2	61.7	3.9	2231.1	23.0
with	3	62.0	3.8	2251.2	25.6
	4	60.2	3.7	2201.6	25.8
	5	60.3	3.7	2175.3	22.8
	1	15.7	0.2	680.6	10.2
	2	15.7	0.2	626.7	10.9
without	3	15.6	0.2	661.6	10.1
	4	15.6	0.2	627.7	11.2
	5	15.5	0.2	617.7	11.8

7 Final Remarks

This paper has introduced an approach that allows for the discovery and modeling of manufacturing processes with non-linear material flows. An algorithm identifies stations in which components are assembled into final or work-in-progress products, and the corresponding material flows. With the addition of the GCP and the corresponding solution procedure, the blocking condition related to the availability of component parts can be added to a simulation model, allowing for the proper estimation of the performance of systems with multiple part identifiers and non-linear material flows. The developed technique can also be used to generate models of disassembly and de-manufacturing systems, since such environments present material flow dynamics comparable to assembly processes and are subject to the same data flattening issues. In addition, this work can be exploited within multi-dimensional model generation techniques, in which manufacturing steps may be characterized by the convergence of different material or information

flows (e.g., kanban). This work is beneficial for digital twins dedicated to production planning and control. Indeed, the proposed algorithm can be applied online, for instance updated with a fixed frequency, and used to guarantee that the digital twin is always a correct representation of the corresponding physical system.

Several limitations still need to be overcome. The proposed approach assumes complete data availability and perfect traces, while realistic datasets are more unreliable. Proper adjustments could be required. Further, the exclusion of batched operations limits the applicability of the approach. In production systems which operate in batches such as thermal treatments or painting processes, the timestamps of completion for several components are equal. Since the approach relies on temporal proximity, the presence of batches would cause several equivalent solutions, along with a high risk of improper identifications. Assuming the absence of batched operations causes the exclusion of certain types of manufacturing systems, for instance, semi-conductor production systems. In general, this approach is ideal for systems with a certain degree of production coordination, in which components are produced in-house and not stored for long times before their usage. Moreover, the computation times suffer greatly from the problem dimensions. Within a real-time framework, if the simulation model is built using as reference a short time window, hence dimensionality issues remain minor. However, higher production rates or longer acquisitions may still determine large datasets. Hence, future developments of this work should also address the optimization of the solution procedure, for instance with the development of meta heuristic approaches or ad-hoc branch-and-bound algorithms. Since the discovery of models with multiple interacting elements is receiving increased interest by researchers, the proposed approach could benefit from the comparison and integration with techniques that share the same goals (Esser and Fahland, 2021). Last but not least, future work should aim at providing a direct comparison between the resources required for ()1) manually building models by simulation experts and (2) employing the proposed technique. Specific experiments must be designed, in which several systems can be identified and for each of them the automated model building phases are compared with traditional, manual-intensive approaches.

Bibliography

S. Bergmann, N. Feldkamp, and S. Strassburger. Approximation of dispatching rules for manufacturing simulation using data mining methods. In *2015 Winter Simulation Conference (WSC)*, pages 2329–2340. IEEE, 2015.

T. Brockhoff, M. S. Uysal, I. Terrier, H. Göhner, and W. M. van der Aalst. Analyzing multi-level bom-structured event data. In *Process Mining Workshops: ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31–November 4, 2021, Revised Selected Papers*, pages 47–59. Springer International Publishing Cham, 2022.

P. Denno, C. Dickerson, and J. A. Harding. Dynamic production system identification for smart manufacturing systems. *Journal of manufacturing systems*, 48:192–203, 2018.

C. Dong, X. Zheng, and J. Yu. Resource modeling of manufacturing process and critical nodes recognition based on the integration of process mining and complex network. *Jixie Gongcheng Xuebao/Journal of Mechanical Engineering*, 55(3):169–180, 2019.

S. Esser and D. Fahland. Multi-dimensional event data in graph databases. *Journal on Data Semantics*, 10(1):109–141, 2021.

D. R. Ferreira and E. Vasilyev. Using logical decision trees to discover the cause of process delays from event logs. *Computers in Industry*, 70:194–207, 2015.

S. Knoch, S. Ponpathirkoottam, and T. Schwartz. Video-to-model: Unsupervised trace extraction from videos for process discovery and conformance checking in manual assembly. In *International Conference on Business Process Management*, pages 291–308. Springer, 2020.

D. Knoll, G. Reinhart, and M. Prüglmeier. Enabling value stream mapping for internal logistics using multidimensional process mining. *Expert Systems with Applications*, 124:130–142, 2019a.

D. Knoll, J. Waldmann, and G. Reinhart. Developing an internal logistics ontology for process mining. 79:427–432, 2019b. ISSN 22128271.

D. Krenczyk, B. Skolud, and A. Herok. A heuristic and simulation hybrid approach for mixed and multi model assembly line balancing. In *International conference on intelligent systems in production engineering and maintenance*, pages 99–108. Springer, 2017.

V. Limère, K. De Cock, and E.-H. Aghezzaf. Generic simulation model for assembly line supply. In *11th Annual Industrial Simulation Conference (ISC-2013)*, pages 192–197. EUROSIS-ETI, 2013.

G. Lugaresi and A. Matta. Discovery and digital model generation for manufacturing systems with assembly operations. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 752–757. IEEE, 2021a.

G. Lugaresi and A. Matta. Automated manufacturing system discovery and digital twin generation. *Journal of Manufacturing Systems*, 59:51–66, 2021b.

N. Martin, B. Depaire, and A. Caris. Using process mining to model interarrival times: investigating the sensitivity of the arpra framework. In *2015 Winter Simulation Conference (WSC)*, pages 868–879. IEEE, 2015.

N. Martin, F. Bax, B. Depaire, and A. Caris. Retrieving resource availability insights from event logs. In *2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 1–10. IEEE, 2016.

N. Martin, M. Swennen, B. Depaire, M. Jans, A. Caris, and K. Vanhoof. Retrieving batch organisation of work insights from event logs. *Decision Support Systems*, 100:119–128, 2017.

M. Milde and G. Reinhart. Automated model development and parametrization of material flow simulations. In *2019 Winter Simulation Conference (WSC)*, pages 2166–2177. IEEE, 2019.

L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. Cyber-physical systems in manufacturing. *Cirp Annals*, 65(2):621–641, 2016.

E. Negri, S. Berardi, L. Fumagalli, and M. Macchi. Mes-integrated digital twin frameworks. *Journal of Manufacturing Systems*, 56:58–71, 2020.

A. H. Ng, J. Bernedixen, M. U. Moris, and M. Jagstam. Factory flow design and analysis using internet-enabled simulation-based optimization and automatic model generation. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 2176–2188. IEEE. ISBN 978-1-4577-2109-0 978-1-4577-2108-3 978-1-4577-2106-9 978-1-4577-2107-6.

E. Passarin and F. Verucchi. Online synchronisation of digital twins: a control-based methodology for manufacturing systems applications. 2022.

A. Peñarroya, F. Casado, and J. Rosell. A computer-aided simulation analysis tool for siman models automatically generated from petri nets. In *International Mediterranean Modelling Multiconference, Barcelona, Spain*, pages 57–62, 2006.

D. W. Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, 2007.

J. L. Peterson. Petri nets. *ACM Computing Surveys (CSUR)*, 9(3):223–252, 1977.

C. Petschnigg, S. Bartscher, and J. Pilz. Point based deep learning to automate automotive assembly simulation model generation with respect to the digital factory. In *2020 9th International Conference on Industrial Technology and Management (ICITM)*, pages 96–101. IEEE, 2020. ISBN 978-1-72814-306-4.

Y. Rao, F. He, X. Shao, and C. Zhang. On-line simulation for shop floor control in manufacturing execution system. In *International Conference on Intelligent Robotics and Applications*, pages 141–150. Springer, 2008.

K. M. Rashid and J. Louis. Process discovery and conformance checking in modular construction using rfid and process mining. In *Construction Research Congress 2020: Computer Applications*, pages 640–648. American Society of Civil Engineers Reston, VA, 2020.

H. Reinhardt, M. Weber, and M. Putz. A survey on automatic model generation for material flow simulation in discrete manufacturing. *Procedia CIRP*, 81:121–126, 2019.

D. Rossit and F. Tohmé. Scheduling research contributions to smart manufacturing. *Manufacturing Letters*, 15:111–114, 2018.

A. Rozinat, R. S. Mans, M. Song, and W. van der Aalst. Discovering simulation models. *Information systems*, 34(3):305–327, 2009.

L. Schruben. Simulation modeling with event graphs. *Communications of the ACM*, 26(11):957–963, 1983.

G. Schuh, A. Gutzlaff, S. Cremer, S. Schmitz, and A. Ayati. A data model to apply process mining in end-to-end order processing processes of manufacturing companies. In *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 151–155. IEEE, 2020. ISBN 978-1-5386-7220-4.

M. Sjarov, T. Lechler, E. Russwurm, J. Fuchs, F. Faltus, E. Schäffer, M. Brossog, and J. Franke. Life cycle of a digital resource twin: Meta-modeling and application example. 104:1644–1649. ISSN 22128271.

F. Tao, Q. Qi, A. Liu, and A. Kusiak. Data-driven smart manufacturing. *Journal of Manufacturing Systems*, 48:157–169, 2018.

S. Tavakoli, A. Mousavi, and A. Komashie. A generic framework for real-time discrete event simulation (des) modelling. In *2008 Winter Simulation Conference*, pages 1931–1938. IEEE, 2008.

M. S. Uysal, S. J. van Zelst, T. Brockhoff, A. Farhang, M. P. Ghahfarokhi, R. Schumacher, S. Junglas, G. Schuh, and W. van der Aalst. Process mining for production processes in the automotive industry.

W. van der Aalst. *Process Mining: Data science in action*. Springer, 2016.

W. van der Aalst. Process mining and simulation: a match made in heaven! In *SummerSim*, pages 4–1, 2018.

W. van der Aalst. Object-centric process mining: Dealing with divergence and convergence in event data. In *International Conference on Software Engineering and Formal Methods*, pages 3–25. Springer, 2019.

W. M. van der Aalst. Federated process mining: Exploiting event data across organizational boundaries. In *2021 IEEE International Conference on Smart Data Services (SMDS)*, pages 1–7. IEEE, 2021. ISBN 978-1-66540-058-9.

Y. Xu, Q. Lin, and M. Q. Zhao. Merging event logs for process mining with hybrid artificial immune algorithm. In *Proceedings of the International Conference on Data Science (ICDATA)*, page 10. The Steering Committee of The World Congress in Computer Science, 2016.

L. Yang, G. Kang, W. Cai, and Q. Zhou. An effective process mining approach against diverse logs based on case classification. In *2015 IEEE International Congress on Big Data*, pages 351–358. IEEE. ISBN 978-1-4673-7278-7.

Appendix A: Selection of Candidate Stations

This section elaborates on the method described in step 2 of the procedure in section 5. This step requires as input three information tables: the event log ($\mathbb{L}$), the BOM level selected at step 1, B_i, and the part types table (i.e., p_a is the part type of assembled product a). Let us define $\mathbb{E}$ as the set of events in the log. $id(e)$ is the part identifier corresponding to the e-th event in the log. Similarly, $st(e)$ is the station – or node – at which event e occurred. Algorithm 1 lists the steps to identify the set of candidate stations $\mathbb{S}_p$. Figure 6 explains this step graphically using an example.

Algorithm 1: Selection of candidate stations (step 2).

Data: Event log $\mathbb{L}$, BOM i-th level B_i, part types table p_a;
Result: Set of candidate stations: $\mathbb{S}_p$;

```
for e ∈ L do
    k ← id(e);
    if p_k ∈ B_i(0) then
        s ← st(e);
        S_C ← s;
```

Appendix B: Definition of the set of combinations $\mathbb{V}$

This section elaborates on the method described in step 3 of the procedure in section 5. The obtained graph model can be divided in the collection of G sub-graphs $\Omega = \{\Omega_1, \ldots, \Omega_G\}$. In step 3, the goal is to determine a set of tuples. For each tuple, the elements are the candidate assembly nodes for the par types of interest. Let us accept the short notation $N(\Omega_i)$ as the function returning the set of nodes for the i-th sub-graph. Algorithm 2 outlines the procedure of this step.

Appendix C: Graph Model Retrieval

Once the GCP problem (section 4.3) has been solved, the solution in terms of graph model can be retrieved with a post-processing step. Namely, the additions defined by the variable z can be derived with the simple procedure listed in Algorithm 3. The algorithm analyses all the components of the solution matrix x^*_{cas}. If $x^*_{cas} = 1$, it means the c-th component has been assigned to the a-th assembly on station s. In this case, the algorithm searches for the last station where a production record exists for component c. Namely, where the last station l recording the time-stamp $t_F(l, c)$. Such station is the starting node of

Algorithm 2: Definition of the set of combinations $\mathbb{V}$.

Data: collection of G sub-graphs Ω, Candidate Stations $\mathbb{S}_p$;

Result: Set of station combinations $\mathbb{V}$;

```
for Ω_i ∈ Ω do
    for n ∈ N(Ω_i) do
        if n ∈ S_p then
            v ← n;
            S_C ← s;
V ← v;
```

Algorithm 3: Graph model retrieval.

Data: GCP solution: x^*_{cas};

Result: Graph model addition variables: z_{ijp};

```
z_ijp ← 0   ∀i, j, p;
for c ∈ C do
    for a ∈ A do
        for s ∈ S do
            if x*_cas = 1 then
                l = arg max_{n∈S} t_F(n, c);
                z_{l,s,p_a} ← 1.
```

the arc that represents the convergence of components of type p_a, toward station s. The arc is represented by $z_{l,s,p_a} = 1$.