Simulation of an Agile Automotive Assembly Lane

Tobias Brockhoff and Anahita Farhang Ghahfarokhi and Mahsa Pourbafrani and Merih Seran Uysal and István Koren and Wil M.P. van der Aalst

Abstract

Recently, process mining has been gaining increasing attention in the production domain; yet its application remains difficult. Event data captured at production and assembly sites often contain a large number of distinct activities that are executed concurrently. This results in complex event data that poses a major challenge for existing process discovery or conformance checking techniques. Despite, from an on-site perspective, the production is highly structured, the discovered models are often spaghetti models. Besides, the control flow perspective that plays a major role in process mining research, is often less interesting in the analysis of production processes. The focus is usually put on performance, yet most out-of-the-box process mining algorithms, which coarsely aggregate performance metrics, fall short in providing detailed insights.

In this document, we introduce simulated datasets of an agile automotive assembly lane. To this end, we first introduce a base version of the model describing its control flow and the parameterization of the processing times. Based on this model, we introduce different scenarios and explain their implementation. Considering the utility of the created event log collection, we briefly describe its major short comings (from the authors point of view). Moreover, despite its simplicity, we illustrate a few challenges when applying out-of-the-box process mining techniques on the simulated data.

I. Model Description

The data was simulated in the scope of a project on the application of process mining to flexible automotive production lanes [5]. Starting from a base model, various performance-related changes were modeled. Theses changes concern the processing times of different stations, operator availability, bottlenecks in the supply chain, and operator efficiency.

Model: Figure 1 depicts a schematic illustration of the base model. First, the frame of the vehicle is mounted on an Automatic Guided Vehicle (AGV) in station GA0. The assembly itself is then built around a lane of general assembly stations $\mathrm{GA} = \{\mathrm{GA1}, \ldots, \mathrm{GA28}\}$. Mounted on the AGV, the frame passes through these stations. For each general assembly step, the station optionally requires parts assembled in an incident subassembly lane. We denote the set of all subassembly stations as $\mathrm{SA} = \{\mathrm{SA}\ 1, \ldots, \mathrm{SA}\ 33\}$. All lane operate concurrently. Note that submaterials are assembled following a make-to-order regime. Right from the start of the submaterial's production, it already assigned to a certain frame rather than being dedicated for storage.

We implemented the model in CPN Tools [3] version 4.0.[1] For each devised scenario (see Section II), we provide the colored Petri net as well as additional configuration files. If multiple scenarios share a model, this model is

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germanys Excellence Strategy-EXC-2023 Internet of Production-390621612.

We also thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

[1]Note that the more recently developed CPN IDE does currently not support to include external Standard ML code files.

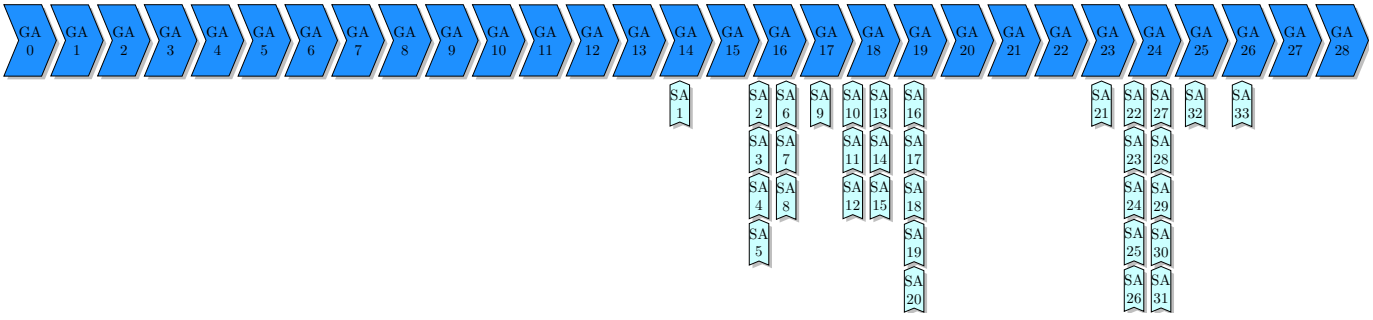


Fig. 1: Schematic illustration of the production lane taken from [5]. Stations of the main assembly lane (GA *) may depend on parts concurrently produced in the incident subassembly lanes (SA *).

TABLE I: Snippet of a dataset. The station is considered the activity (concept:name), resources are numbered (org:resource), start (start_timestamp) and complete (time:timestamp) are logged, the processing time is provided (@@duration), and cases (case:concept:name) are enumerated.

concept:name	org:resource	@@startevent_concept:name	@@startevent_org:resource	start_timestamp	time:timestamp	@@duration	case:concept:name
SA 32	59	SA 32	59.0	2020-06-01 08:00:00+00:00	2020-06-01 08:44:05+00:00	2645.0	1
SA 31	47	SA 31	47.0	2020-06-01 08:00:00+00:00	2020-06-01 08:44:27+00:00	2667.0	1
SA 26	42	SA 26	42.0	2020-06-01 08:00:00+00:00	2020-06-01 08:44:53+00:00	2693.0	1
SA 1	34	SA 1	34.0	2020-06-01 08:00:00+00:00	2020-06-01 08:44:54+00:00	2694.0	1
SA 9	19	SA 9	19.0	2020-06-01 08:00:00+00:00	2020-06-01 08:44:57+00:00	2697.0	1
SA 5	20	SA 5	20.0	2020-06-01 08:00:00+00:00	2020-06-01 08:45:02+00:00	2702.0	1
SA 20	38	SA 20	38.0	2020-06-01 08:00:00+00:00	2020-06-01 08:45:37+00:00	2737.0	1
SA 21	53	SA 21	53.0	2020-06-01 08:00:00+00:00	2020-06-01 08:46:38+00:00	2798.0	1

saved in their shared parent directory. Additionally required utility Standard ML code is contained in the `base` directory. To ensure that CPN Tools can parse any of the provided Petri nets, these file must be copied to the same location as the Petri net.

Data: Each dataset is provided as a csv-file. Each row contains information on typical event data attributes such as *case id*, *activity*, and *resource*. Moreover, the data contain activity start and complete lifecycle transitions and, thus, every row contains a start and complete timestamp. In total, there are 8 columns:

concept:name Name of the station. Can be considered the activity.
org:resource Id of the worker that operated the station.
@@startevent_concept:name Label of the start event (always equals concept:name).
@@startevent_org:resource Label of the start event (always equals concept:name).
start_timestamp Start timestamp of the assembly step.
time:timestamp Complete timestamp of the assembly step.
@@**duration** Processing time.
case:concept:name Id of the vehicle constituting the case id.

Table I shows a snippet of the base log (`base-log.csv`). Irrespective of the floating point representation, which is an artifact from loading the data into [2], the values in the column @@startevent_org:resource are equal to the id in column org:resource. Note that, as common for discrete event simulation, all subassembly lanes start simultaneously (comp. Section III).

Processing Times: We sample the processing time of a vehicle at a station using truncated log-normal distributions. More precisely, consider a station s and let

$$X^s \sim \text{Lognormal}(\mu^s, (\sigma^s)^2) \Leftrightarrow \ln(X^s) \sim \mathcal{N}(\mu^s, (\sigma^s)^2) \tag{1}$$

denote a log-normally distributed random variable. As such, the corresponding probability density function is

$$f_{X^s}(x) = \frac{1}{x\sigma^s\sqrt{2\pi}} \exp\left(-\frac{(\ln(x-\mu^s)^2}{2(\sigma^s)^2}\right). \tag{2}$$

Let PT^s denoted the random variable describing the log-normally distributed processing times for station $s \in \text{GA} \cup \text{SA}$.

To account for unrealistic outliers, we furthermore truncate the processing time distribution. Given a lower bound $\tau_l^s \in \mathbb{R}^+$ and upper bound $\tau_h^s \in \mathbb{R}^+$ on the processing time, our sampled processing times adhere to the probability density

$$f_{X^s}(x|\tau_l^s \leq X^s \leq \tau_h^s). \tag{3}$$

We denote the induced truncated log-normal distribution as

$$\text{Lognormal}(\mu^s, (\sigma^s)^2, \tau_l^s, \tau_h^s). \tag{4}$$

Let PTtr^s denoted the random variable describing the *truncated*, log-normally distributed processing times for station $s \in \text{GA} \cup \text{SA}$.

To better model small differences between stations, the mean and variance were additionally randomly selected from an interval of possible values.

II. Model Configurations

Base Model: The event log obtained from the base model is provided in the file `base-log.csv`. For each general assembly station $s \in \mathrm{GA}$ and subassembly station $s' \in \mathrm{SA}$, we set

$$\mu^s \in [8.593, 8.603], \qquad \sigma^s \in [0.05, 0.051], \tag{5}$$
$$\tau_l^s = 0.95 \cdot e^{\mu^s} > 85\,\mathrm{min}, \qquad \tau_h^s = 1.05 \cdot e^{\mu^s} < 95\,\mathrm{min}, \tag{6}$$
$$\mu^{s'} \in [7.9, 7.921], \qquad \sigma^{s'} \in [0.05, 0.051], \tag{7}$$
$$\tau_l^{s'} = 0.95 \cdot e^{\mu^{s'}} > 43\,\mathrm{min}, \quad \tau_h^{s'} = 1.05 \cdot e^{\mu^{s'}} < 48\,\mathrm{min}. \tag{8}$$

Thereby, the (truncated) mean and standard deviation of the sampled processing times are expected to be

$$\mathrm{E}[\mathrm{PT}^s] \in [90\,\mathrm{min}, 91\,\mathrm{min}], \qquad \mathrm{SD}[\mathrm{PT}^s] \in [4.5\,\mathrm{min}, 4.63\,\mathrm{min}], \tag{9}$$
$$\mathrm{E}[\mathrm{PTtr}^s] \in [90\,\mathrm{min}, 91\,\mathrm{min}], \quad \mathrm{SD}[\mathrm{PTtr}^s] \in [2.4\,\mathrm{min}, 2.5\,\mathrm{min}], \tag{10}$$
$$\mathrm{E}[\mathrm{PT}^{s'}] \in [45\,\mathrm{min}, 46\,\mathrm{min}], \qquad \mathrm{SD}[\mathrm{PT}^{s'}] \in [2.26\,\mathrm{min}, 2.34\,\mathrm{min}], \tag{11}$$
$$\mathrm{E}[\mathrm{PTtr}^{s'}] \in [45\,\mathrm{min}, 46\,\mathrm{min}], \quad \mathrm{SD}[\mathrm{PTtr}^{s'}] \in [1.22\,\mathrm{min}, 1.24\,\mathrm{min}]. \tag{12}$$

A. Processing Time Changes

One type of change that we applied to the baseline model concerns processing times.

Small Static Station Change: In the dataset saved as `small-change-log.csv`, the average processing times of the general assembly station GA11 and the subassembly stations SA17 and SA24 are increased. In particular, the base parameterization was changed as follows:

$$\mu^{\mathrm{GA11}} = 8.698, \quad \mu^{\mathrm{SA17}} = 8, \quad \mu^{\mathrm{SA24}} = 8.1 \tag{13}$$
$$\tau_l^{\mathrm{GA11}} \approx \tau_h^{\mathrm{GA11}} \approx \mu^{\mathrm{GA11}},\ \tau_l^{\mathrm{SA17}} \approx \tau_h^{\mathrm{SA17}} \approx \mu^{\mathrm{SA17}},\ \tau_l^{\mathrm{SA24}} \approx \tau_h^{\mathrm{SA24}} \approx \mu^{\mathrm{SA24}}. \tag{14}$$

Consequently, we expect the following mean processing times

$$\mathrm{E}[\mathrm{PTtr}^{\mathrm{GA11}}] \approx 100\,\mathrm{min}, \quad \mathrm{SD}[\mathrm{PTtr}^{\mathrm{GA11}}] \approx 0.03\,\mathrm{min}, \tag{15}$$
$$\mathrm{E}[\mathrm{PTtr}^{\mathrm{SA17}}] \approx 50\,\mathrm{min}, \quad \mathrm{SD}[\mathrm{PTtr}^{\mathrm{SA17}}] \approx 0.01\,\mathrm{min}, \tag{16}$$
$$\mathrm{E}[\mathrm{PTtr}^{\mathrm{SA24}}] \approx 55\,\mathrm{min}, \quad \mathrm{SD}[\mathrm{PTtr}^{\mathrm{SA24}}] \approx 0.01\,\mathrm{min}. \tag{17}$$

Generally Increased Processing Time Variance: In the dataset `higher-variance-log.csv`, we increased the variance in the processing times for all stations. In particular, for each general assembly station $s \in \mathrm{GA}$ and subassembly station $s' \in \mathrm{SA}$, we used

$$\mu^s \in [8.597, 8.7], \qquad \sigma^s \in [0.1, 0.15], \tag{18}$$
$$\tau_l^s = 0.95 \cdot e^{\mu^s} > 85\,\mathrm{min}, \qquad \tau_h^s = 1.075 \cdot e^{\mu^s} < 107\,\mathrm{min}, \tag{19}$$
$$\mu^{s'} \in [7.9, 8.18], \qquad \sigma^{s'} \in [0.1, 0.15], \tag{20}$$
$$\tau_l^{s'} = 0.95 \cdot e^{\mu^{s'}} > 38\,\mathrm{min}, \quad \tau_h^{s'} = 1.075 \cdot e^{\mu^{s'}} < 64\,\mathrm{min}. \tag{21}$$

Consequently, we expect the following mean processing times

$$\mathrm{E}[\mathrm{PTtr}^s] \in [90\,\mathrm{min}, 100\,\mathrm{min}], \quad \mathrm{SD}[\mathrm{PTtr}^s] \in [3.16\,\mathrm{min}, 3.55\,\mathrm{min}], \tag{22}$$
$$\mathrm{E}[\mathrm{PTtr}^{s'}] \in [40\,\mathrm{min}, 4060], \qquad \mathrm{SD}[\mathrm{PTtr}^{s'}] \in [1.42\,\mathrm{min}, 2.12\,\mathrm{min}]. \tag{23}$$

Increased Processing Time Variance - Slow Start 1: Like in `higher-variance-log.csv`, the variance in the processing times is increased. Moreover, the mean processing time for general assembly stations at the beginning of the assembly lane is larger than for the other stations. The dataset is saved as `higher-variance-slow-start-log.csv`. In particular, the parameterization for $s \in \{\mathrm{GA3}, \mathrm{GA4}, \mathrm{GA5}, \mathrm{GA6}\}$ is as follows:

$$\mu^s \in [8.65, 8.77], \qquad \sigma^s \in [0.1, 0.15], \tag{24}$$
$$\tau_l^s = 0.95 \cdot e^{\mu^s} > 90\,\mathrm{min}, \quad \tau_h^s = 1.075 \cdot e^{\mu^s} < 102\,\mathrm{min}. \tag{25}$$

Consequently, we expect the following mean processing times

$$\mathrm{E}[\mathrm{PTtr}^s] \in [96\,\mathrm{min}, 108\,\mathrm{min}], \quad \mathrm{SD}[\mathrm{PTtr}^s] \in [3.3\,\mathrm{min}, 3.8\,\mathrm{min}]. \tag{26}$$

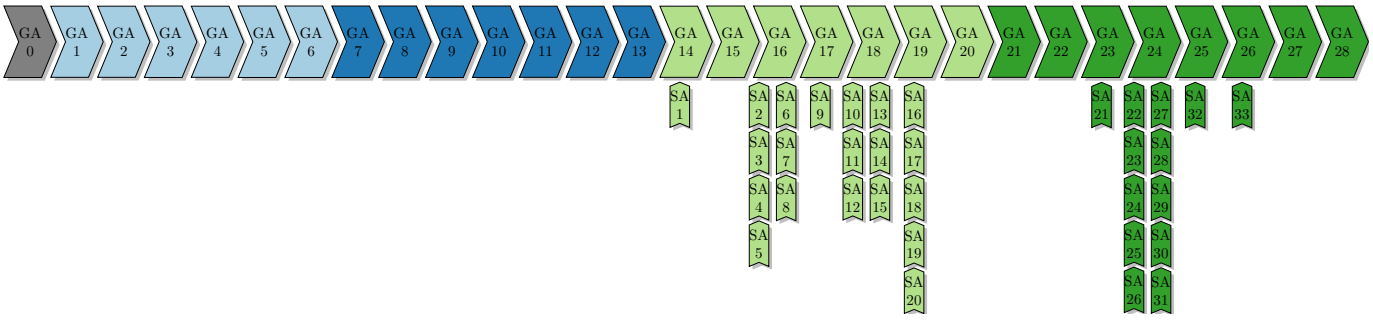


Fig. 2: Worker pool station responsibilities. The assembly lane is partitioned into four sections, and each section is operated by a pool of operators.

Increased Processing Time Variance - Slow Start 2: Like for `higher-variance-slow-start-log.csv`, but even larger mean processing times for $s \in \{\text{GA3}, \text{GA4}, \text{GA5}, \text{GA6}\}$:

$$\mu^s \in [8.83, 8.87], \qquad \sigma^s \in [0.11, 0.15], \tag{27}$$

$$\tau_l^s = 0.95 \cdot e^{\mu^s} > 108\,\text{min}, \quad \tau_h^s = 1.075 \cdot e^{\mu^s} < 127\,\text{min}. \tag{28}$$

Consequently, we expect the following mean processing times

$$\text{E}[\text{PTtr}^s] \in [115\,\text{min}, 119\,\text{min}], \quad \text{SD}[\text{PTtr}^s] \in [4.04\,\text{min}, 4.23\,\text{min}]. \tag{29}$$

The dataset is saved as `higher-variance-slow-start-2-log.csv`.

Increased Processing Time Variance - Slow End: In contrast to increased processing times at the beginning of the assembly lane, the dataset `higher-variance-slow-end-log.csv` represents the symmetric scenario. In this dataset, the processing times are increased for general assembly stations at the end of the assembly lane. More precisely, the same parameterization as for `higher-variance-log.csv` is used except for $s \in \{\text{GA24}, \text{GA25}, \text{GA26}, \text{GA27}\}$:

$$\mu^s \in [8.68, 8.87], \qquad \sigma^s \in [0.1, 0.15], \tag{30}$$

$$\tau_l^s = 0.95 \cdot e^{\mu^s} > 93\,\text{min}, \quad \tau_h^s = 1.075 \cdot e^{\mu^s} < 128\,\text{min}. \tag{31}$$

The resulting moments of the processing time distribution are:

$$\text{E}[\text{PTtr}^s] \in [99\,\text{min}, 120\,\text{min}], \quad \text{SD}[\text{PTtr}^s] \in [3.46\,\text{min}, 4.24\,\text{min}]. \tag{32}$$

B. Operator Availability Changes

For the next set of event logs, we first modeled station operators. Inspired by the underlying use case, there are "worker pools" for different sections of the production. The section partitioning is depicted in Figure 2. In principle, each worker can be assigned to each station in the respective production section, and each stations requires an operator for its execution. To model worker-dependent service times, each worker is assigned an additional factor that is multiplied with the sampled station service time. For this set of event logs, this factor is ignored (i.e., it is set to 1). In addition, each worker has preferences about the stations. These preferences are considered when assigning the worker. Initially, each pool contains as many operators as there are stations in the section.

Changing Pool Sizes I: The dataset `pool-3-27-to-7-step-2-log.csv` contains data from a scenario where the availability of operators changes over time. In this dataset, we reduced the number of workers in the third operator pool (light green station in Figure 2) every two weeks. Starting with 27 operators, which is more than sufficient to guarantee immediate processing, this number was reduced until there were only 7 operators left. This number still allows to simultaneously operate all general assembly stations in this section; however, in this case, there is no capacity left for the subassembly stations. The sampling of station processing times was parameterized as in `higher-variance-log.csv`.

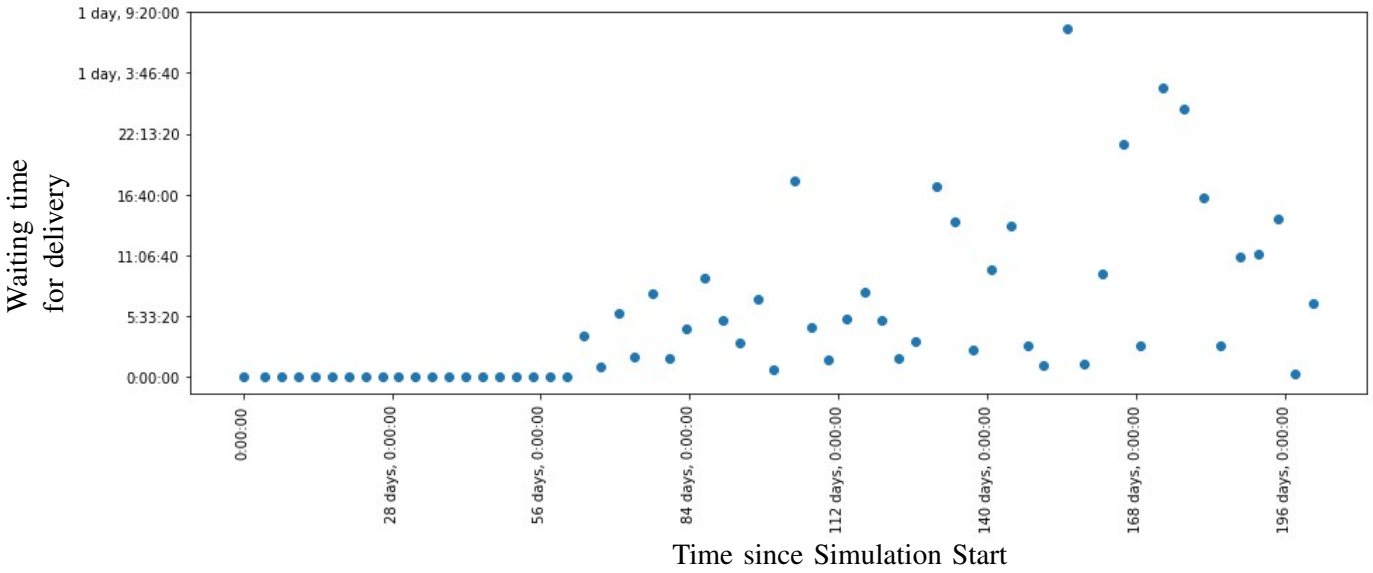


Fig. 3: Example waiting times due to delayed delivery of submaterial (`GA14-supply-0h-6h-12h-log.csv`).

Changing Pool Sizes II: The dataset `pool-3-27-to-3-step-2-log.csv` was generate like the dataset `pool-3-27-to-7-step-2-log.csv` with the only differences that the number of operators was reduced down to *three* workers.

C. Supply Shortages

For the next four scenarios, we modeled problems in the supply chain. To this end, in each scenario, there is one station that requires a part delivered by a subcontractor. The on-site storage capacity is furthermore limited to 50 parts for each external submaterial, and submaterial is delivered in batch sizes of 50 parts. Whenever the on-size storage runs empty, the station must pause its operation until the next delivery arrives. In particular, the expected time until the next delivery changes twice in each simulated log. While for the first 20 orders the required submaterial arrives promptly (i.e., as soon as the storage runs empty, the next 50 parts are delivered), the next 20 orders arrive delayed and the last 20 orders are even more delayed. We thereby sample the delay from an exponential distribution. The sampling of station processing times was parameterized as in `higher-variance-log.csv`.

The log collection comprises four datasets: `GA12-supply-0h-24h-36h-log.csv`, `GA12-supply-0h-6h-12h-log.csv`, `GA14-supply-0h-6h-12h-log.csv`, and `SA17-supply-0h-6h-12h-log.csv`. The folder names provide information about the station and the expected delays. Initially, the submaterial is delivered promptly (0 h), while for the remaining orders the delivery is delayed with the folder name showing the expected delay. An example showing the resulting waiting times for submaterial at station GA14 is depicted in Figure 3. After initial prompt delivery, the expected waiting time increases to 6 h and, later, to 12 h, respectively.

D. Operator Impact

The last two datasets are concerned with the impact of differences between operators. The setup is similar to the one described in Section II-B. However, in this scenario, we adapt the processing time factors associated with individual workers.

Static Slower Worker: In the dataset `static-slower-log.csv`, the processing time factor was increased to two for two out of the seven operators in the first operator pool. These operators require twice as long to complete an assembly step.

Worker Learning: Initially, the configuration for the `learning-log.csv` is the same as in `static-slower.csv`. However, the processing time factor for the two slower workers improves over time. After the first 1000 cases, the factor is reduced to 1.5, and, finally, it is set to one after another 1000 cases.

III. Limitations

The data was created using discrete event simulation. Thus, it exhibits typical characteristics such as strict working hours. Moreover, it does not contain real outliers neither in terms of control flow nor considering performance.

Another potential problem is the strict concurrency—and thus independence—between the general assembly lane and the subassembly lanes. As soon as a case is created in the simulation, all subassembly lanes can start. As a consequence, the subparts are often assembled long before they are actually required. Nevertheless, this also introduces an interesting problem when assessing the performance. There are long waiting times for submaterials that skew the general waiting time distribution (comp. Section IV). Yet, early assembly of submaterials is not necessarily a major concern (given that storage is not extremely expensive).

Considering the scenario where operators become a scarce resource, the scheduling of operators is random. Thus, if multiple assembly operations are pending, it is not guaranteed that an operator is assigned to the most critical task (e.g., the operator is scheduled for a subassembly task rather than moving forward the general assembly lane).

Finally, the introduced changes are rapid and often extremely strong.

IV. Challenges

In this section, we will briefly sketch problems that arise during an out-of-the-box application of standard process mining techniques. In doing so, we used the process mining workbench ProM 12[2].

First, despite an assembly lane can be considered highly structured (comp. Figure 1), the number of activities and degree of concurrency makes process discovery challenging. For example, Figure 4a shows the process model discovery by the Inductive Miner Infrequent Algorithm (IMf) [4] with lifecylce handling for the dataset `higher-variance-slow-start-log.csv`. While the model properly reflects the order of the general assembly stations, their true structural relation to the subassembly stations is not shown. The true dependencies in terms of material (e.g., GA22 requires parts assembled in SA21) were not discovered. Likewise for the model discovered by the Flexible Heuristics Miner (FHM) [6] depicted in Figure 4b.

The control flow shown in both model can be explained as follows: there is no direct relation between general assembly stations and their associated subassembly stations because, in the data, most subassembly steps are completed already shortly after the assembly started. Therefore, in the discovered models, most subassembly stations are positioned at the beginning of the process. Such findings can often be complicated to communicate to stakeholders who might think of their assembly lane in terms of models such as the one depicted in Figure 1. Finally, there is another potentially severe problem with this model that can make it difficult to proceed with the analysis. To proceed with conformance checking and process enhancement, one typically first aligns the model and the log [1]. Based on these alignment, one can devise conformance diagnostics and enhance the model by performance metrics. Yet, computing alignments on the discovered model can be hard. In fact, the alignment computation implementation that is internally used by the Inductive Visual Miner plugin did not terminated within 10 minutes on an Intel Core i7-8550U @ 1.8 GHz machine. However, using the Multi-perspective Process Explorer (MPE) plugin, it was possible to compute alignments.

Performance Analysis: A common step in process enhancement is to project performance metrics onto a discovered process model. However, for production processes such as the considered one, one faces three main challenges: 1) the size of the model, 2) concurrent subassembly processes, and 3) low granularity of the projected measurements. As illustrated in Figure 5, showing projected performance diagnostics using the MPE plugin in ProM, Petri net-based visualizations quickly become to big to be readable. Since the concurrency between different assembly lanes is not captured properly, we also obtain performance measurements that not very meaningful. Consider Figure 5a, rather than showing the time between the completion of GA3 and GA4, the reference point for both stations is GA5. Moreover, SA17 is depicted as a predecessor of GA5, despite it is not a subassembly station of the latter. Yet, the time between completing SA17 and GA5 is depicted. Even if we consider the snippet of the ground truth model (including life cycle transitions) shown in Figure 6, concurrency becomes a challenge. The figures shows that the median time to start with SA21 after the vehicle is scheduled for assembly is 1.2 days. Compared to the processing times, this is extremely long; yet, it is not necessarily a problem. As as consequence, such long waiting times easily skew the visualization. In fact, neither of the models properly shows the actual

[2]www.promtools.org

problem of the considered scenario—namely, that the increased average processing time at the start of the general assembly lane.

REFERENCES

[1] Adrian Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. "Towards Robust Conformance Checking". In: *Business Process Management Workshops*. Ed. by Michael zur Muehlen and Jianwen Su. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 122–133.

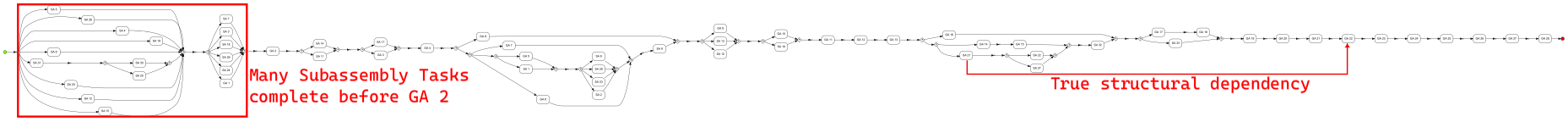
[2] Alessandro Berti, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. "Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science". In: *Proceedings of the ICPM Demo Track 2019, co-located with 1st International Conference on Process Mining (ICPM 2019)*. 2019.

[3] *CPN Tools*. https://cpntools.org/. Accessed: 2023-06-21.

[4] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. "Discovering Block-Structured Process Models from Event Logs - A Constructive Approach". In: *PETRI NETS*. Springer, 2013, pp. 311–329.

[5] Merih Seran Uysal et al. "Process Mining for Production Processes in the Automotive Industry". In: *Industry Forum of 18th International Conference on Business Process Management*. 2020.

[6] A. Weijters and Joel Ribeiro. "Flexible Heuristics Miner (FHM)". In: *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*. Apr. 2011, pp. 310–317. DOI: 10.1109/CIDM.2011.5949453.

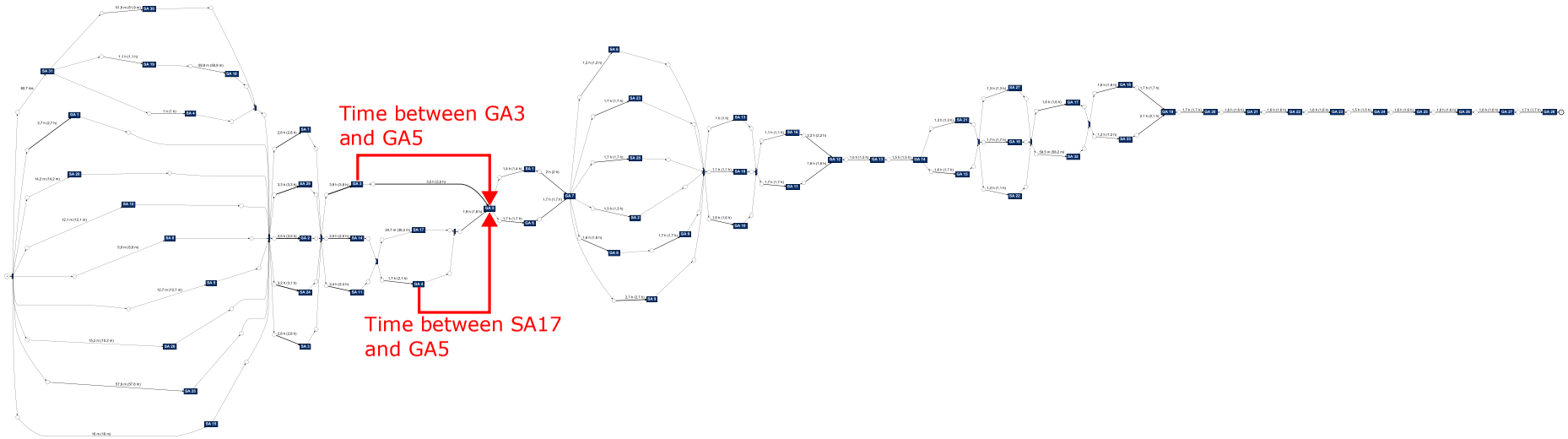


(a) Process model discovered using the Inductive Miner infrequent with lifecylce handling algorithm and noise threshold 0.8.

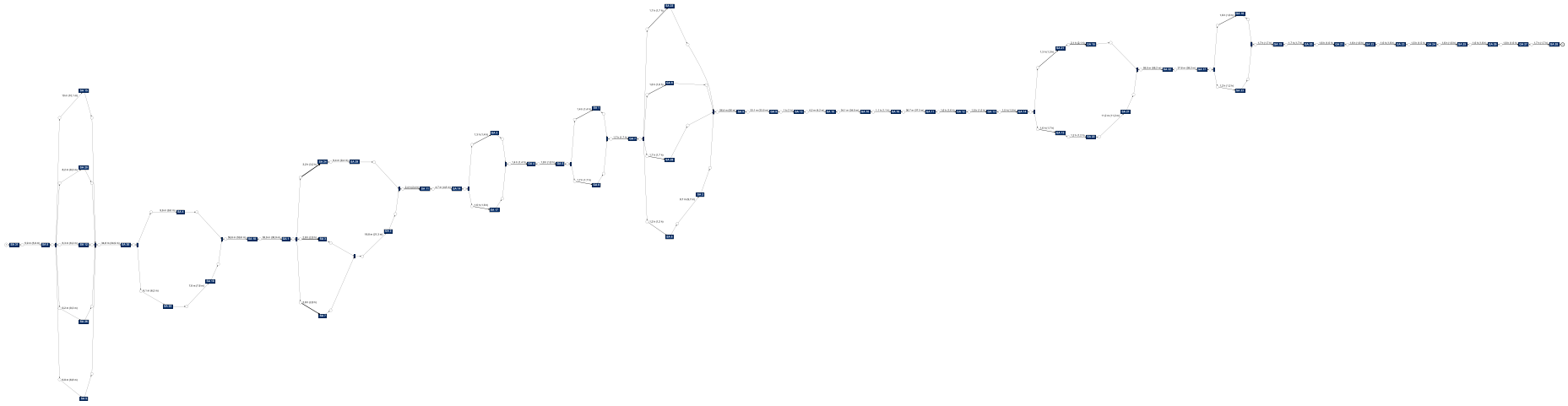


(b) Process model discovered using the flexible heuristic miner.

Fig. 4: Process Models discovered using common process discovery techniques



(a) Performance view for the IMf model



(b) Performance view for a Petri net obtained from the FHM C-net

Fig. 5: Model-based Performance Analysis using the MPE plugin for the discovered process models

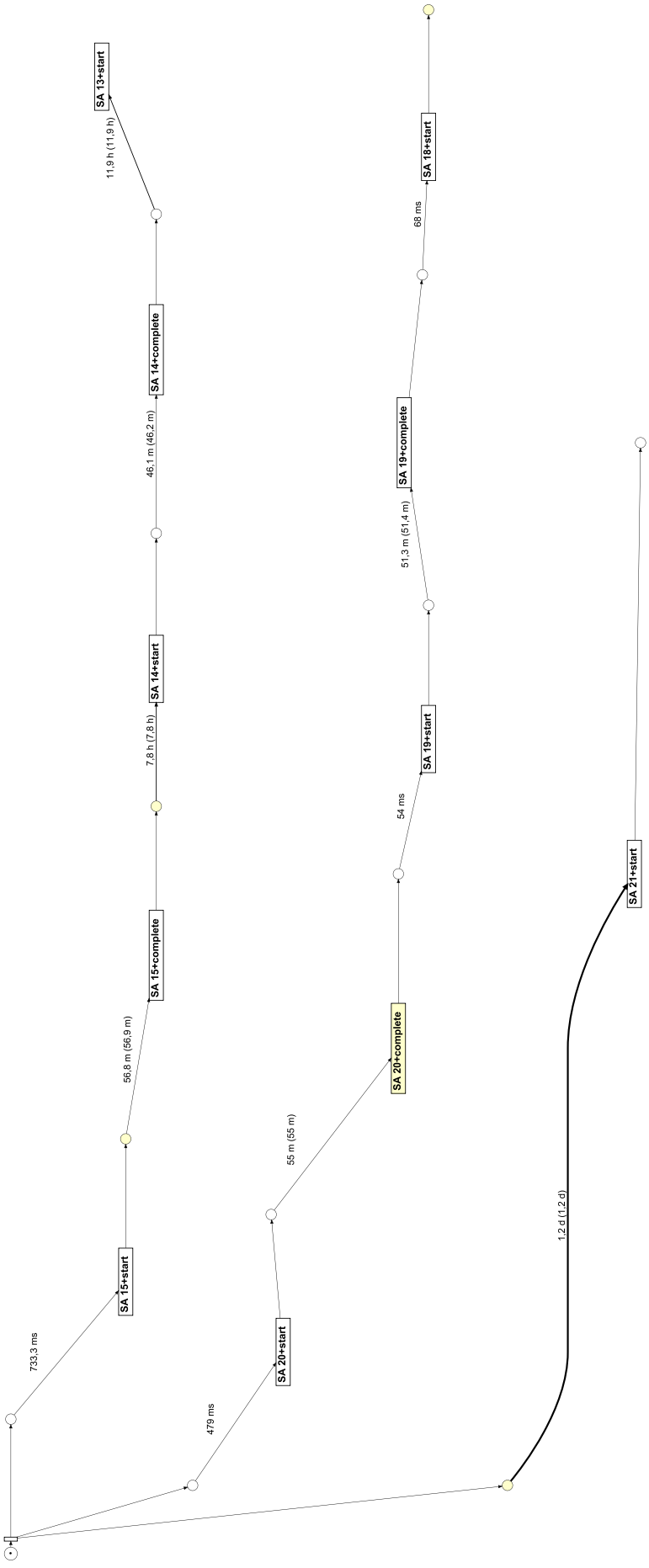


Fig. 6: Ground truth model with lifecycle information