Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 200 (2022) 1836–1845

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Industry 4.0 and Smart Manufacturing

Combining metaheuristics and process mining: Improving cobot placement in a combined cobot assignment and job shop scheduling problem

Alexander Kinast[a,*], Karl F. Doerner[b], Stefanie Rinderle-Ma[c]

[a] *University of Vienna, Forschungsplattform Data Science, Kolingasse 14-16, 1090 Wien, Austria, alexander.kinast@univie.ac.at*
[b] *University of Vienna, Department of Business Decisions and Analytics, Oskar-Morgenstern-Platz 1,1090 Wien, Austria, karl.doerner@univie.ac.at*
[c] *Technical University of Munich, Department of Informatics, Chair for Information Systems and Business Process Management, Boltzmannstrasse 3, 85748 Garching, Germany, stefanie.rinderle-ma@tum.de*

Abstract

Human workers can share a workspace with modern collaborative robots (cobots). The main differences to traditional robots are, that workers do not need a safety distance when interacting with cobots, as they move slower than typical industrial robots. Cobots also have fast setup times compared to traditional resources. The hybridization is based on a previous work of the authors, where a job shop scheduling problem that is extended with a robot to workstation assignment with a hybrid genetic algorithm is solved. In this paper, the potential of hybridizing an optimization algorithm with process mining techniques to improve the solution quality by gaining information on the solution structure is analyzed. This additional information should help to guide the search process. Process mining techniques are presented to analyze the solutions and learn from them. The idea of this work is to understand the solutions generated by the genetic algorithms as process executions, e.g., the production of a part as a process instance executed across the selected work stations. Then, by generating process event log data out of selected solutions, state-of-the-art process mining techniques can be used as visualization and scanning tools for the underlying processes. This way, for example, bottleneck workstations in the production process can be highlighted. Based on created scenarios, this paper demonstrates how genetic algorithms and process mining techniques can be combined. In the future, it is planned that information that is mined from generated logs of an evaluation framework is used to improve the performance of hybrid genetic algorithms by using this information in a feedback loop. Generated insight in cobot placement can also be used for prescriptive analytics in real-world manufacturing companies that want to utilize cobots. The focus of this paper lies in the discussion of the usage of potential information extracted from process mining.

Keywords: Process mining; Hybrid genetic algorithm; Collaborative robots; Job shop scheduling; Prescriptive analytics

* Corresponding author.
E-mail address: alexander.kinast@univie.ac.at

1877-0509
10.1016/j.procs.2022.01.384

1. Introduction

Automation has gained significant importance in the past years, see e.g. [5]. Process automation is used to improve product quality, efficiency, and availability of the production process. For many processes in production companies, it is important to use state-of-the-art technology in order to stay competitive in the future. In [16], it is shown, that collaborative robots (cobots) offer new opportunities for different fields of manufacturing companies. These cobots are designed to work with human workers and no safety distance is necessary. Through this interaction with humans, cobots allow automatizing workstations, where it was previously not possible. Cobots can be used for activities such as picking, welding, assembling, or inspecting products.
The different cooperation modes between a cobot and a human actor are described in [3]. The mode that is the furthest away from cooperation is a traditional robot in a cell that acts alone. When a cobot and a human worker share a workspace, the following modes are differentiated:

- Synchronization:
  Even if the human and the cobot share a workspace in the synchronized mode, only one of both actors works on the task at the same time.
- Cooperation:
  In the cooperation mode, both actors can work simultaneously at the same shared workspace on different tasks.
- Collaboration:
  The most interactive mode is the collaboration mode, where both actors can work at the same time on the same task.

An example for that is interactive process automation based on a picking and placing station [9].
Cobots can be used if traditional robots are not flexible enough or if the workstation does not allow a degree of automation that is typically used for industrial robots [20]. Through a combination of the humans' strengths such as flexibility, adaptability, or decision making and the cobots' strengths such as speed, endurance, and accuracy the performance of many workstations can be increased. In comparison to traditional resources, cobots have convenient setup times (programmable within half a day) which allows companies to adapt to fast changes in production.
When deploying cobots to a real-world environment it is possible to use prescriptive analytics. In [8], it is described that prescriptive analytics uses historical and real-time data in combination with optimization algorithms and expert systems to predict possible outcomes of a system. This is done to predict delays or bottlenecks and present options for how these can be avoided. However, this means that it is necessary to understand why bottlenecks or delays happened in the past and in the case of cobots, why cobots are placed at specific workstations.

1.1. Problem description and Solution technique

Due to a budget constraint, most companies will only invest in a limited amount of cobots. These cobots should assist workstations, where a bottleneck is expected in the next planning period. Depending on the objective function, this can be workstations that will delay the reset of the production or workstations where the production cost can be significantly reduced by deploying a cobot.
When knowing the orders for the next planning period, it is important to determine on which workstation tasks of these orders should be produced and where to place cobots to have maximum impact. This impact can be measured in various ways, examples include production cost, makespan, lead time, tardiness, or a combination of these and other factors.
Genetic algorithms are commonly used for the optimization of complex problems. These genetic algorithms are population-based algorithms that start with an initial random generation. With genetic operators such as selection, crossover, and mutation, new generations get created until a stopping criterion gets reached. A big advantage of genetic algorithms is that they can be applied to nearly all optimization problems and deliver high-quality solutions. The basic concepts and properties of genetic algorithms are explained in [2].
Since traditional resources are not as flexible as cobots, most existing algorithms are not designed to reallocate resources. In our previous work [6], a new encoding for a job shop scheduling problem for a genetic algorithm is developed. This encoding is able to assign tasks to workstations, give them a priority, and assign cobots to worksta-

tions. It is shown, that even a small number of cobots can lead to a high improvement of the solution quality. This is due to the fact, that the cobots are deployed to bottleneck workstations that slow down the whole production or workstations that create high production costs.
A genetic algorithm, as described in [6] can be used. This algorithm considers the production speed and production cost from the different workstations in addition to the orders that should be produced on each workstation. Using this algorithm results in a high-quality solution, however, it might not be clear what factors or problem characteristics lead to the decisions of the algorithm.
Whenever the genetic algorithm generates an encoded solution, the evaluation framework is used to evaluate this solution. During this evaluation, the production of all orders is simulated and tasks are assigned to workstations. This task to workstation assignment with start/end timestamps and additional properties is logged to the event log file.
Process mining is a technique that can be used to mine implicit knowledge from log files. When the genetic algorithm is executed, log files from selected solutions can be created which then can be analyzed with process mining techniques and the results can be visualized.
Based on these considerations the following questions arise:

- What factors influence the cobot to workstation assignment?
- Can process mining visualize relevant factors that lead to the decision of the genetic algorithm?
- Can information gained from process mining improve the quality of results found by the genetic algorithm?

Since this is a novel optimization problem, it is not clear what factors besides the objective function are relevant factors that influence the placement of a cobot. Examples of such factors could be the number of workstations that can handle the same tasks, the duration of the individual tasks, or the number of tasks.
State-of-the-art process mining techniques should be used to visualize the attributes such as production cost and production time that lead to the decisions of the genetic algorithm. If process mining techniques can be used to visualize the decisions of the genetic algorithm, it might be possible to use this implicit knowledge to improve the performance of the genetic algorithm.

2. Solution method

2.1. Process mining

"Process automation and process mining are regarded as key technologies for digital transformation" [11]. Process mining has created high expectations due to the increased level of transparency [10], particularly in manufacturing [14]. Process execution information – explicitly managed by a process engine or implicitly executed by one or several information systems – is stored in process event logs. The main part of a log is a large number of traces. A trace is a set of events related to one case. In a production company, this trace could be one order and the events could then define the production steps that have been executed to produce this order. Various extensions can be used to define attributes of traces and events. These attributes that are defined by extensions are essential for many process mining algorithms. The ".xes" format is commonly used for representing process event logs [1] and is also used for generated log files in this paper.
Process mining [15] comprises techniques to a) discover the underlying process model from the process event logs (process discovery), b) check if the process event log conforms to an expected process model (conformance checking), and c) enhance process models. To discover process models out of these event logs, well-known process mining techniques, such as the inductive miner can be used. In [13] it is shown, that the inductive miner is able to generate process models out of car manufacturing processes and identify bottlenecks in the manufacturing process. This inductive miner is used to generate BPMN models out of log data in this paper.

2.2. Design Choices

In order to use process mining to find factors that influence the placement of cobots, process mining has to be able to analyze the results generated by a metaheuristic. Process mining is based on implicit knowledge that is mined from event log files, however, this is not the default output of a metaheuristic. The genetic algorithm with a biased random-key encoding developed in [6], has been used to solve the extended cobot placement and job shop scheduling problem. The objective function is the sum of production cost and makespan. The genetic algorithm is extended in such a way, that a ".xes" event log file is created for the best solution that has been found during the execution of the algorithm. This can be seen in Figure 1.

The created event log file of the best solution of the genetic algorithm is used as a basis for further process mining

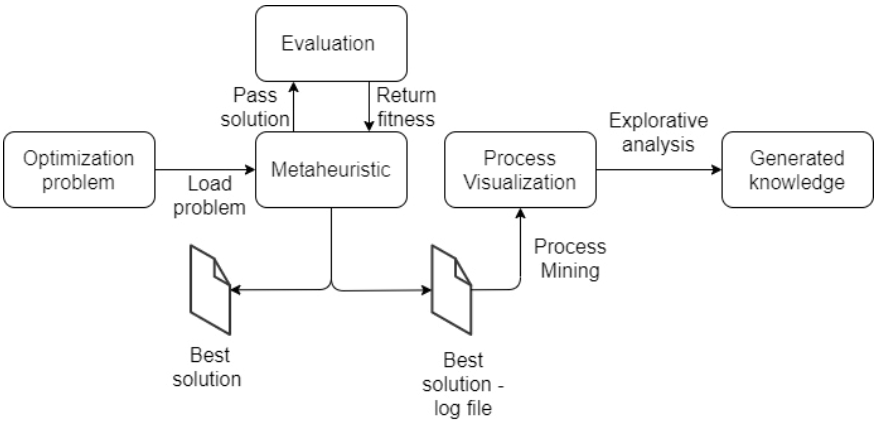

Fig. 1. Overall approach

steps. Potential results or typical patterns of this process mining step are visualized in this paper and explorative analysis steps are used to generated knowledge. This can be seen in Figure 1. The idea is, that the logs of the best solutions contain the implicit knowledge of why and based on what factors cobots are placed on workstations. In

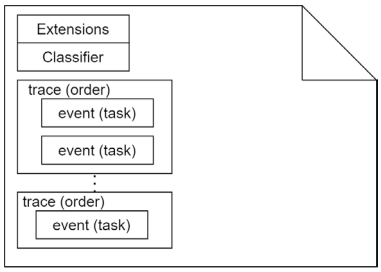

Fig. 2. Event log file structure

Figure 2, the structure of such an event log file can be seen. The file starts with an extensions section, that defines attributes that can be used for different process mining algorithms. Examples of such commonly used attributes are process start and end dates. The next section is a classifier section. These classifiers are predefined sets of attributes that are used for the nodes in the mined models. To reach the goal of displaying interesting attributes of workstations, the classifier for the scenarios is the workstation where a process is executed. The main part of the file contains traces and events. In this case, traces relate to orders that should be produced and events in these traces contain information about tasks that have to be performed. An example order could be the production of a mechanical part. To produce this mechanical part, different tasks such as cutting, drilling, welding, and assembling have to be done.

The standard representation for business processes is the Business Process Modeling and Notation (BPMN) format (www.bpmn.org). This format has the advantage of being easily understandable for end-users. Hence, we opt for mining process models in BPMN format from the process event logs. To mine a BPMN model out of the generated

event log files a process discovery algorithm, an inductive miner, as described in [7], is used. This process mining method generates a Petri net, which is converted to a BPMN model.
For analyzing the solution of the genetic algorithm mined BPMN model should be enriched with attributes that highlight why a cobot is placed on a specific workstation. Following [4], the attribute values are displayed by visualizations styles applied to the process tasks, i.e., the color and size of the nodes are varied in order to represent attribute values. The attributes that are visualized are attributes that have been logged to the event log file. The goal of this visualization is, that decision-relevant attributes for cobot placement are discovered, that can be used alongside mined information to improve cobot placement.

2.3. Implementation

The genetic algorithm with a biased random-key encoding as described in [6] is implemented in HeuristicLab [17]. For the evaluation of one solution, the simulation framework Easy4Sim provided by the RISC Software GmbH has been used. This framework can be easily extended and allows an efficient evaluation of individuals that have been generated by the genetic algorithm [19].
To perform process mining tasks on the event log files that have been created during the evaluation in the evaluation framework, the process mining framework pm4py is used [18].

3. Preliminary results

In [12], fundamental patterns that can occur when describing a business process are described. To find out which attributes influence the placements of cobots in the extended cobot assignment and job shop scheduling problem, scenarios are created with the following control-flow patterns:

- Sequence
  An activity can be done after a preceding activity finishes.
- Exclusive choice
  A branch is split into two or more branches. However, only one of these branches is active after the split.
- Simple merge
  Two or more branches are joined together. If one of the incoming branches is active, the outgoing branch will be active.
- Loop
  A loop allows the representation of cycles in a process model.

In the extended cobot assignment and job shop scheduling problem described in [6], tasks are limited to one predecessor and one successor task. This is also assumed in the generated scenarios and therefore no parallel split and parallel merge scenarios are created. Based on these patterns, the goal was to find cobot placement relevant attributes. In the created scenarios, each task that should be produced has a base duration that is modified by a speed factor of a workstation. When a task with a base duration of 100 seconds is produced on a workstation with a speed factor of 0.8, the task will be finished in 80 seconds. The cost to produce a task is the total time multiplied by the cost factor. This means if the task with a total duration of 80 is produced on a workstation with a cost factor of 1.5, a cost of 120 will be generated. The objective function for the optimization for these examples is simply the sum of costs and makespan (total time until all tasks have been finished). It is assumed that a cobot speeds up the production by 30% and, therefore, also reduces the costs by 30%.

3.1. Attribute visualization

In [4] it is described, how scaling, positioning, and labeling can be used to visualize multiple process attributes. For the following scenarios we use scaling, labeling, and additionally coloring of the nodes for the visualization. The

production cost will be visualized with the use of color in the BPMN. The color ranges from green (cheap workstation compared to the other workstations) to red (expensive workstations). The width of the nodes in the BPMN represents the production time of the workstations (a higher production time means a wider node). The production cost and production time are also displayed in the label of the node. At the left top of each scenario, the base duration of the tasks is stated. The color of the start and end node has no semantics.

3.2. Scenario 1 - exclusive choice

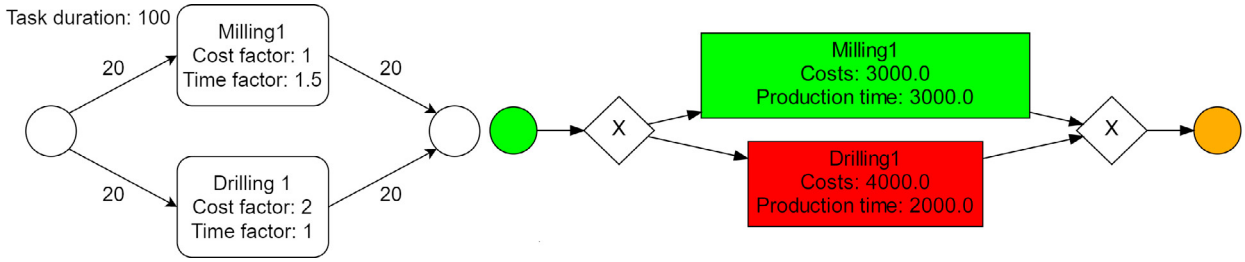

Fig. 3. Scenario 1 (Cost: color, Production time: width of the nodes)

In Figure 3 on the left side of the first scenario, two workstations work in parallel and both should produce 20 tasks that have a base duration of 100. For this parallel part, the tasks are split with an exclusive choice and merged with a simple merge after the production. On the right side, it can be seen that the mined BPMN model has been enriched with information from the workstations. It is shown that the workstation ”Milling1” created costs of 3000 and had a production time of 3000. The drilling workstation finished significantly faster in 2000 and had costs of 4000. When taking a closer look at how much the objective function gets influenced due to the deployment of a cobot, the following changes can be seen:

- Milling1
  - Cost: -900
  - Production time: -900
- Drilling1
  - Cost: -1200
  - Production time: -600 (not relevant for the makespan)

The total objective function in this example is 10000 (7000 costs + 3000 makespan). Since the production time of the drilling workstations is not relevant for the objective function, the algorithm will place a cobot to the milling workstation if one cobot is available, reducing the objective value to 8200.

3.3. Scenario 2 - exclusive choice and sequence

In Figure 4, a more complex version of the first scenario is shown. In comparison to Figure 3, the additional workflow pattern sequence is used. In the scenario depicted in Fig. 4 on the left side, there is only 1 task for milling/drilling and 1 task for assembling/turning. This task has a base duration of 2000. When running the algorithm with no cobots, the result that is shown at the bottom is generated. When allowing the algorithm to place a cobot, this cobot gets placed on one of the top workstations. This is due to the fact that these workstations have the highest costs and are relevant for the makespan. Placing the cobot on one of these workstations will reduce the objective function from 22000 (14000 costs + 8000 makespan in the top path) to 19800 (1200 cost and 1000 makespan reduction).
When considering the version of this scenario shown at the top right in Figure 4, the only change is that instead of one task with a duration of 2000, there are now 20 tasks with a duration of 100. Since tasks can be started on the next machine once they are finished, this will lead to a new makespan of 6100 (the top path will finish at 4200).

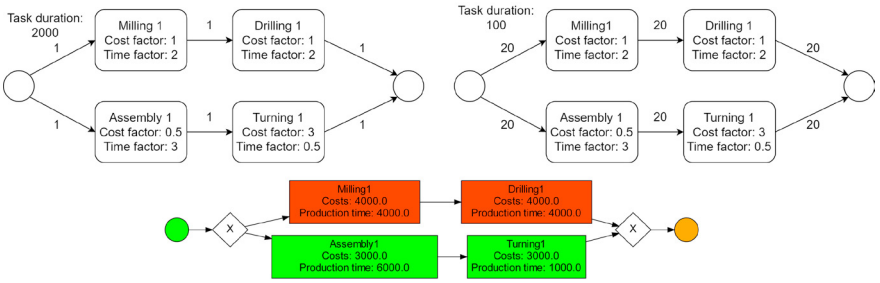

Fig. 4. Scenario 2 (Cost: color, Production time: width of the nodes)

When allowing the algorithm to place a cobot, the cobot now gets placed at the assembly workstation (high costs and the only makespan relevant workstation). Reducing the objective function from 20100 (14000 costs and a makespan of 6100) to 17370 (13100 costs and a makespan of 4370). This scenario demonstrates how the task duration can influence the cobot placement.

3.4. Scenario 3 - workstation groups

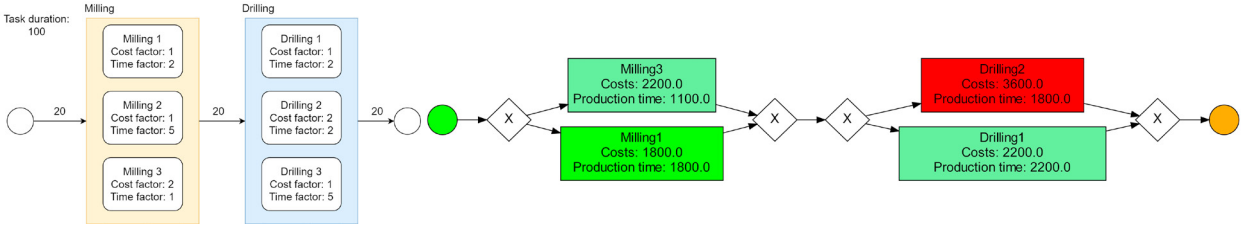

Fig. 5. Scenario 3 (Cost: color, Production time: width of the nodes)

In Figure 5 on the left hand side, a scenario with workstation groups can be seen. For each workstation group, an exclusive choice and a simple merge pattern is used. This means that for all 20 tasks a milling task has to be done before a drilling task. A milling task can be produced on all milling workstations and a drilling task can be done on all drilling workstations.
After running the genetic algorithm with 0 cobots, the BPMN model on the right side of Figure 5 can be mined from the log files. What can be seen is, that the two workstations with the highest time factor are not utilized at all. The drilling workstation with cost factor two gets only nine tasks assigned and still has the highest cost.
In Figure 6, the computed solution when the algorithm is allowed to place one cobot can be seen. Interestingly, the

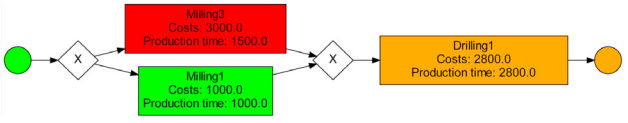

Fig. 6. Scenario 3 - 1 cobot (Cost = color, Production time = width of the nodes)

best solution is to place the cobot on the cheapest drilling workstation and then produce all tasks on this workstation, while the more expensive workstations are idle.

3.5. Scenario 4 - loop

In Figure 7, a rather simple scenario with three workstations is shown. Ten orders need a preparation, drilling, and packaging task. When allowing the placement of one cobot, this cobot gets placed on the preparation workstation,

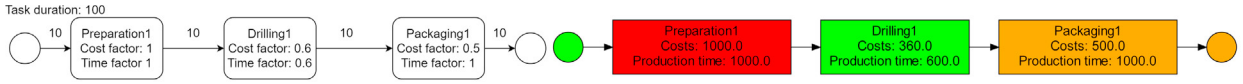

Fig. 7. Scenario 4.1 (Cost: color, Production time: width of the nodes)

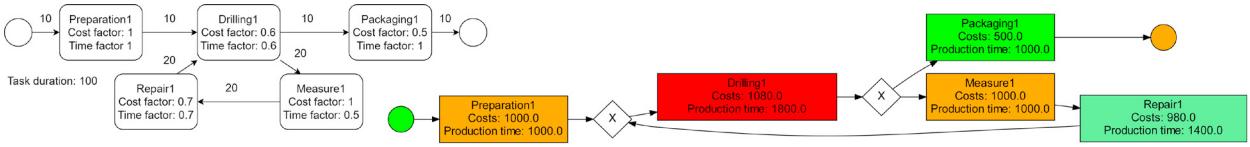

Fig. 8. Scenario 4.2 (Cost: color, Production time: width of the nodes)

because this workstation has the highest costs and the longest production time. When looking at Figure 8, it can be seen that this is an extended version of scenario 4.1. In this scenario, the drilling workstation has an error and an additional measure, repair, and drilling step is necessary two times, for each drilling task in scenario 4.1. This will result in the workflow pattern, a loop.
In this example, the drilling workstation has the lowest sum of production speed and cost factor. However, in the mined model on the right side of Figure 8, it is apparent that this workstation is part of the loop and is therefore highly utilized. Through this utilization, the workstation still has high costs and a long production time. When allowing the algorithm to place one cobot, this cobot gets deployed to the drilling in comparison to the preparation workstation of scenario 4.1.

3.6. Findings

Based on the discussed scenarios, we group properties that influence the placement of a cobot in two categories. The first category is properties that have a direct impact on the objective function:

- Time factor
- Cost factor
- Task amount
- Base duration of the tasks

These four properties result in production cost and production time and all four have a high impact on the placement of cobots.
The second category is properties of the scenario that can influence the placements of cobots, even if they do not directly influence the objective function:

- Workstation groups
- Scenario layout

Workstation groups allow the algorithm to shift tasks between workstations resulting in interesting choices as it is demonstrated in Figure 5. The layout of the scenario can also influence the cobot deployment in a meaningful way. An example would be the loop in Figure 8 that will favor assigning the cobot to a fast and cheap workstation.
The scenarios have been constructed to demonstrate what factors influence the placement of a cobot and show that process mining has huge potential to visualize and analyze the results of genetic algorithms. The default output for these algorithm runs would be a text file that shows the cobot-to-workstation assignment and the task to workstation assignment. For small scenarios, this could manually be transformed to visualizations as shown in this paper. However, when the scenarios increase in complexity, this is not possible any longer.

4. Conclusion and outlook

This paper demonstrates that process mining can interact with genetic algorithms in a way that visualizes the results of the genetic algorithm. Based on these visualizations, important properties for the placement of a cobot in a job shop scheduling problem are identified.
It is shown that implicit knowledge that is generated by utilizing a genetic algorithm or another heuristic solution approach on an extended cobot assignment and job shop scheduling problem can be analyzed with modern process mining approaches. The findings of this process mining, especially the cobot placement relevant properties, should help to enable prescriptive analytics for cobot placement in real-world companies.
State-of-the-art process mining might be applicable to analyze the optimization results from other optimization techniques and problems. This can allow companies to analyze and visualize optimization results where it was previously not possible. These visualizations can be used to gather implicit process knowledge as demonstrated in this paper and to improve the processes based on the generated knowledge. Process mining might be an interesting tool to better understand the results of optimization algorithms for any kind of real-world company or researcher.
In our upcoming work, the found factors that influence the placement of a cobot should be used in addition to implicit knowledge that is mined from event logs. This combination should be used to improve the quality of the results that are generated by a genetic algorithm or another metaheuristic on a large real-world data set.
Additionally, scenarios with conditional decisions and parallel workflow patterns should be reviewed. Such a conditional decision would be that, depending on a predecessor task, only a subgroup of successor tasks is allowed. For instance, that only successor workstations within a specific range to a workstation are allowed. This will mean, that orders can choose between different paths through the shop floor. In such scenarios, the physical position of a workstation will have an impact on the placement of cobots.

References

[1] Acampora, G., Vitiello, A., Di Stefano, B., van der Aalst, W., Günther, C., Verbeek, E., 2017. Ieee 1849tm: The xes standard. IEEE Computational Intelligence Magazine , 4–8.

[2] Affenzeller, M., Wagner, S., Winkler, S., Beham, A., 2009. Simulating evolution: Basics about genetic algorithms, in: Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. CRC Press, pp. 0–10. https://doi.org/10.1201/9781420011326.

[3] Bauer, W., Bender, M., Braun, M., Rally, P., Scholtz, O., 2016. Lightweight robots in manual assembly – best to start simply! Examining companies' initial experiences with lightweight robots. Technical Report. Fraunhofer Institute for Industrial Engineering IAO.

[4] Gall, M., Rinderle-Ma, S., 2020. Assessing process attribute visualization and interaction approaches based on a controlled experiment. Int. J. Cooperative Inf. Syst. 29, 2050007:1–2050007:33. https://doi.org/10.1142/S0218843020500070.

[5] Jämsä-Jounela, S.L., 2007. Future trends in process automation. Annual Reviews in Control 31, 211–220. URL: https://www.sciencedirect.com/science/article/pii/S1367578807000387, doi:https://doi.org/10.1016/j.arcontrol.2007.08.003. https://doi.org/10.1016/j.arcontrol.2007.08.003.

[6] Kinast, A., Doerner, K.F., Rinderle-Ma, S., 2021. Biased random-key genetic algorithm for cobot assignment in an assembly/disassembly job shop scheduling problem. Procedia Computer Science 180, 328–337. https://doi.org/10.1016/j.procs.2021.01.170.

[7] Leemans, S.J., Fahland, D., van der Aalst, W.M., 2013. Discovering block-structured process models from event logs-a constructive approach, in: International conference on applications and theory of Petri nets and concurrency, Springer. pp. 311–329.

[8] Lepenioti, K., Bousdekis, A., Apostolou, D., Mentzas, G., 2020. Prescriptive analytics: Literature review and research challenges. International Journal of Information Management 50, 57–70. https://doi.org/10.1016/j.ijinfomgt.2019.04.003.

[9] Mangat, A.S., Mangler, J., Rinderle-Ma, S., 2021. Interactive process automation based on lightweight object detection in manufacturing processes. Comput. Ind. 130, 103482. https://doi.org/10.1016/j.compind.2021.103482.

[10] Reinkemeyer, L., 2020. Process Mining in Action – Principles, Use Cases and Outlook. Springer International Publishing. https://doi.org/10.1007/978-3-030-40172-6.

[11] Rinderle-Ma, S., Mangler, J., 2021. Process automation and process miningin manufacturing, in: Int'l Conf. Business Process Management. (to appear).

[12] Russell, N., Ter Hofstede, A.H., Van Der Aalst, W.M., Mulyar, N., 2006. Workflow control-flow patterns: A revised view. BPM Center Report BPM-06-22, BPMcenter. org 2006.

[13] Siek, M., Mukti, R.M.G., 2020. Process mining with applications to automotive industry. IOP Conference Series: Materials Science and Engineering 924, 012033. URL: https://doi.org/10.1088/1757-899x/924/1/012033. https://doi.org/10.1088/1757-899x/924/1/012033.

[14] Stertz, F., Mangler, J., Scheibel, B., Rinderle-Ma, S., 2021. Expectations vs. experiences – process mining in small and medium sized manufacturing companies, in: Business Process Management (Forum). (accepted for publication).

[15] Van Der Aalst, W., 2016. Process Mining - Data Science in Action, Second Edition. Springer. https://doi.org/10.1007/978-3-662-49851-4.
[16] Vojić, S., 2020. Applications of collaborative industrial robots. Machines. Technologies. Materials. 14, 96–99.
[17] Wagner, S., Kronberger, G., Beham, A., Kommenda, M., Scheibenpflug, A., Pitzer, E., Vonolfen, S., Kofler, M., Winkler, S., Dorfer, V., Affenzeller, M., 2014. Architecture and design of the heuristiclab optimization environment, in: Advanced Methods and Applications in Computational Intelligence. Springer, pp. 197–261. https://doi.org/10.1007/978-3-319-01436-4_10.
[18] Website of Fraunhofer FIT - pm4py, . Fraunhofer fit - pm4py. https://pm4py.fit.fraunhofer.de/. Accessed: 2021-07-23.
[19] Website of the RISC Software GmbH, . Risc software gmbh. https://risc-software.at/. Accessed: 2021-07-23.
[20] Weckenborg, C., Kieckhäfer, K., Müller, C., Grunewald, M., Spengler, T.S., 2020. Balancing of assembly lines with collaborative robots. Business Research 13, 93–132. URL: https://link.springer.com/article/10.1007/s40685-019-0101-y. https://doi.org/10.1007/s40685-019-0101-y.