SOBA: A Self-Organizing Bucket Architecture to Reduce Setup Times in an Event-Driven Production

Martin Krockert, Marvin Matthes, Torsten Munkelt
Faculty of Computer Science
Dresden University of Applied Sciences
Dresden, Germany
Email: *firstname.lastname*@htw-dresden.de

***Abstract*—Modern industry prefers self-organization in production over central production planning for the sake of greater flexibility, faster response to disruptions and to deviations, and less effort. The strategy also propagates highly customizable products. These products mostly require different group technologies and therefore cannot be grouped into lots. This leads to a huge number of operations that have to be scheduled and processed, which in turn leads to high computation times for scheduling algorithms and high setup costs for production due to frequent setup changes. To ensure high flexibility and robust planing, we present a self-organizing bucket architecture (SOBA) to group equal operations which require the same group technology to reduce setup times and even maintain high flexibility and robust planning for any scheduling algorithm. In this paper, we explain our approach and show an implementation of the approach in our self-organizing production. Furthermore, we show a set of empirical studies that compares our approach to simple and exhaustive queuing rules. The tests show the superiority of our approach and indicate further development opportunities.**

***Keywords*—Self-Organization; Self-Adaptation; Production; Group Technology; Job Shop; Setup Time Reduction**

I. Introduction

Industry 4.0 propagates a lot size of one for modern production of piece goods, because customers more often expect individual products and there is no reason to combine individual products into conventional lots, because machines process individual products differently and have to be setup for every individual product [1]. If a machine only processes lots the size of one in no specific sequence and the products differ from each other, more frequent setups are necessary which lead to longer overall setup times. Irrespective of all measures to shorten setup times technologically, setup times of considerable length still occur [2]. It is well known that long setup times extend throughput times and reduce effective capacity utilization [3]. In order to reduce the number of setups, thereby shortening the overall setup time and ultimately reducing their negative effects, we propose our Self-Organizing Bucket Architecture (SOBA) that combines separate operations into so-called buckets. All operations in a bucket require the same setup, but can belong to different products. SOBA only combines operations to be processed on the same machine successively in a certain period of time. The machine processes the operations from the bucket successively according to urgency. Buckets exist only temporarily: Before SOBA creates a bucket and fills it with operations, the operations are independent of each other; and after the machine processed all operations from the bucket, the bucket dissolves and its operations are then independent of each other again. In this way, SOBA differs from conventional lot sizing, which usually keeps lots together during the processing of their production orders (although it sometimes splits and overlaps lots). In this paper, we present the implementation of SOBA and some results of an empirical study. SOBA is part of our event-driven, self-organizing production, and SOBA itself is driven by events and organizes itself.

The paper is organized as follows: The next section classifies the problem and provides a review of related work. Section III declares the concept including architecture and algorithms of SOBA. Subsequently, Sections IV and V describe the implementation of SOBA into our self-organizing production system and our empirical study. Finally, Section VI concludes and gives an outlook on further work.

II. Problem classification

In job shop production, an efficient and reactive scheduling for a given set of machines and jobs is essential to meet the economical objectives. In the context of manufacturing each product is represented by a job, which contains a sequence of operations. In job shop scheduling every operation has to be processed without interruption on any capable machine, while a machine can only process operations one by one. The job shop problem considering more than two machines is known to be NP-complete [4]–[6].

Operations with similar characteristics, which require the same machine setups are called group technology (GT), as they share the same technology requirements but are assigned to different products [7], [8].

Consequently, the operations combined to one group can be seen as a horizontal cross-linked aggregation of different products over the same GT. The optimization problem for scheduling job shop systems reaches another dimension of complexity by integrating setup times [9]. By grouping operations requiring the same technology, the schedule becomes more efficient, because operations of the same group can be processed without intermediate machine setups, which eliminates additional setup times compared to pure priority heuristics. In real-world manufacturing, especially simple dispatching rules become the most applied solution to solve the

Copyright (c) IARIA, 2020. ISBN: 978-1-61208-781-8

scheduling problem in a highly unpredictable and dynamical environment [10]. Dispatching rules are applied to queues to prioritize the operation to be processed next.

Moreover, *Dispatching rules* can be divided into two groups, exhaustive and non-exhaustive rules. While exhaustive rules perform all operations of a GT existing in one queue, non-exhaustive rules allow splitting of grouped operations and therefore switching of setups even though there are still operations remaining requiring the same setup. In previous research from Frazier, exhaustive rules prove to be superior to non-exhaustive rules in flow shop production [11]. As our research focuses on high flexibility and robustness in a job shop scenario, we developed a non-exhaustive approach and compare it with an exhaustive rule. Therefore, the problem is classified as a job shop scheduling problem including setup times and it differs from the flow shop problem analyzed by Frazier. But like Frazier, we cover uncertainty: Customer orders/Sales orders/Jobs arrive after exponentially distributed inter-arrival times, and processing times of operations are log-normally distributed. Because of this uncertainty, it is impossible to generate a optimal schedule. In order to still gain a robust schedule, we need a dynamic grouping and scheduling of operations. Therefore, we create dynamic time scopes based on forward and backward scheduling, then apply the dynamic time scopes to group operations within one GT, and delegate the group to a scheduling mechanism afterwards.

Under the term of group planning heuristics, dispatching rules resulted in approaches that solve the planning problem of operations that require the same technology. Those heuristics have received increasing attention. They reduce setup times and increase processing efficiency in production [11]–[14]. Several studies with focus on flow cell manufacturing and group heuristics already exists [10], [11], [15] and were an inspiration to this paper. Aside from dispatching rules, mixed integer linear programming (MILP) can be used to solve the GT-based scheduling. For our research, we do not consider MILP as it requires complete knowledge of all operation instances to find a solution; and using a suitable algorithm already leads to high computational costs, even for small instances of scheduling problems as [16] showed.

To sum up, our approach to reduce setup times combines non-exhaustive dispatching rules with a dynamic time scope to solve the job shop scheduling problem under uncertainty. Our approach differs from previous research, as the focus lies on a job shop scheduling problem. Most previous concepts focus on cell manufacturing, where the queuing of operations happens in front of a cell with identical machines [10]. Forward and backward scheduling is a well known technique, but to the best of our knowledge, we are not aware of other research applying forward and backward scheduling to group operations in a job shop production.

III. The concept of SOBA

A. Structure of the self-organizing bucket architecture

Figure 1 shows the structure of SOBA. It consists of three elements: a set of jobs, one or more ”SOBA - Bucket Managers” and a set of machines. Each *job* contains a sequence of operations. Thus, alternative and parallel operations are not considered yet. Moreover, all operations contain the following information: estimated processing time, due time of the job, average transition time, required setup time, and earliest start as well as latest start obtained by forward and backward scheduling.

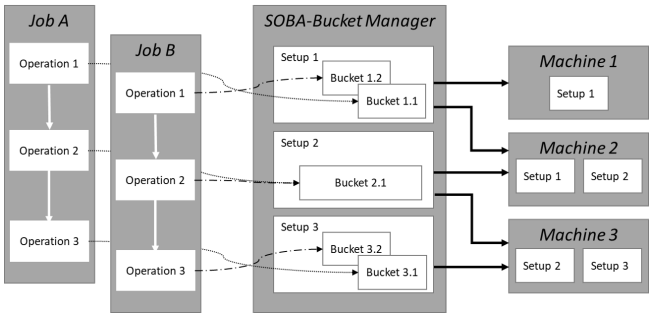

Figure 1. Self-organizing bucket architecture

The resources are represented by machines in the context of our production. Each resource can be assigned to one or more setups, but only be equipped with one setup at a time. Each setup represents a certain group technology to process an operation, i.e., drill a hole with a diameter of 4 mm. The same group technology can be provided by different resources. As shown in Figure 1, *Setup 1* is assigned to *Machine 1* and *Machine 2*. Thus, setups can be assigned to more then one machine and each machine can be equipped with one of the assigned setups. This way, it is possible to have multiple machines equipped with the same setup at the same time.

The bucket manager is a persistent instance. It assigns open operations requiring the same group technology to buckets linked to the same setup. If there is no suitable bucket for an operation, the bucket manager creates a bucket and assigns the operation to this bucket. After creating a bucket, the allocation of the bucket to the resource can start. How the allocation is carried out is irrelevant for the creation of the buckets. The bucket manager organizes the varying number of *buckets*, which are created upon or filled with incoming operations. As the number of potential operations in a bucket can be equal to the number of all operations of the same group technology, multiple buckets can be created for each setup to maintain flexibility by splitting or merging operations into new buckets. But each bucket must contain at least one operation. This is the operation the bucket manager originally created the bucket for. After the initial creation of the bucket, further operations can be added depending on the time scope. To determine the time scope we use forward and backward scheduling when the job and its operations enter the production initially. After processing the last operation of the bucket and removing it from the bucket, the bucket dissolves.

B. Creating the scope for buckets

Considering industry 4.0, materials become smart by giving them not only essential information about their current status.

TABLE I
SYMBOL DEFINITION

Symbol	Definition
gt	group technology with $gt \in GT$
d	duration with $d > 0\,,\, d \in \mathbb{N}$
dt	due time with $dt > 0\,, dt \in \mathbb{N}$
sbt	start time from backward scheduling
sft	start time from forward scheduling with $(sbt > sft)$
o	is a column vector with $(gt, d, dt, sbt, sft)^T$
O	is a set $\{o_1, \cdots, o_n\}$ of operations sorted by $sft(o)$
b	is a vector with $\begin{pmatrix} O \\ gt \leftarrow gt(o_1) \\ d \leftarrow \sum_{o \in O} d(o) \\ dt \leftarrow dt(o_1) \\ sbt \leftarrow sbt(o_1) \\ sft \leftarrow sft(o_1) \\ scp \leftarrow sbt(o_1) - sft(o_1) \end{pmatrix}$

Moreover, the materials get digital twins providing them with further details, i.e., their locations, their bills of materials and their operations. We aim to calculate a time scope from the processing and transition times for each operation.

The bucket manager should assign the operation to the bucket which is the most suitable bucket regarding processing and completion time. The procedure to find a suitable bucket is shown in Figure 2.

While exploding the bill of material, each operation is scheduled individually. Crucial for the scheduling, the operation's duration includes the processing and transition time. After calculating the schedule both, the start and end times for the backward schedule and the forward schedule for each operation, are set. As example, Operation 1 in Figure 3 has an earliest start time of 0 from forward scheduling and a latest start time of 9 from backward scheduling. The resulting time frame is the scope, in which the operation shall be processed in order to finish the product in time.

Let us assume an empty production without any bucket. By scheduling the first operation, the Bucket Manager responsible for the group technology required by the operation would create a new bucket with the earliest start time 0 and the latest start time 9. The scope of the bucket would be a total of 9 time units. Usually scopes between earliest start time and latest start time can be very large, if the orders enter the production immediately. Therefore, we apply a maximum bucket scope, to avoid blocking of resources. We suggest a setup-specific maximum bucket size depending on working shifts, such as 8 hours, 4 hours, 2 hours or less. To ensure high flexibility, we do not set a minimum bucket size, as we want single urgent operations to be processed as fast as possible. In our experiment we make sure, that the start time from backward scheduling is always higher then the start time from forward scheduling. Thus, a time scope can be determined. Otherwise buckets would only contain single operations.

Data: S is a set of all existing buckets
A is a priority queue of operations ordered by $\rightarrow sft(o)$
T is a set of buckets requiring equal gt

Input: An operation o

1 **Procedure:** ManageBuckets(o)
2 $A \leftarrow \{o\}$
3 **while** $A \neq \emptyset$ **do**
4 // take the most important
5 // operation and remove it from A
6 $o \leftarrow o_1 \in A$
7 $A \leftarrow A \setminus \{o\}$
8 // create a set of possible
9 // matching buckets
10 $T \leftarrow \{b \in S | gt(o) = gt(b)\}$
11 // find all buckets with later
12 // start to dissolve them and
13 // remove them from T
14 $tmp \leftarrow \{b \in T | sft(b) > sft(o)\}$
15 $A \leftarrow A \cup \{o | (o \in b) \wedge (b \in tmp)\}$
16 $T \leftarrow T \setminus tmp$
17 // find fitting buckets in T
18 // and take one bucket with the
19 // least start time from forward
20 // scheduling
21 $tmp \leftarrow \left\{ b \in T \,\middle|\, \left(\sum_{p \in b} d(p) \right) + d(o) \leq scp(b) \right\}$
22 **if** $tmp \neq \emptyset$ **then**
23 $b \leftarrow$ an arbitrary $b \in \arg\min_{t \in tmp} sft(t)$
24 $O(b) \leftarrow O(b) \cup \{o\}$
25 **else**
26 new b with $O(b) \leftarrow \{o\}$
27 $S \leftarrow S \cup \{b\}$
28 **end**
29 **end**

Figure 2. Procedure assign operation to bucket

C. Procedure to create, find and modify buckets

As mentioned before, buckets are virtual elements organized by the bucket manager. The buckets are created, modified or deleted if any event occurs concerning the involved group technology. All properties of a bucket shown in Table I are calculated based on the operations assigned to the bucket. As shown in Algorithm 2, by receiving a new operation, the bucket manager determines the matching buckets depending upon their group technology (see line 10).

All buckets with an earliest start higher than the earliest start

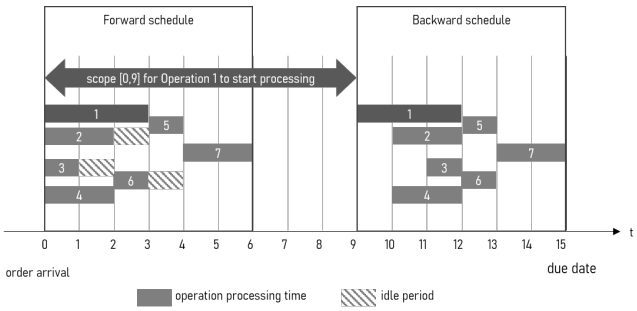

Figure 3. Scope definition by forward and backward scheduling

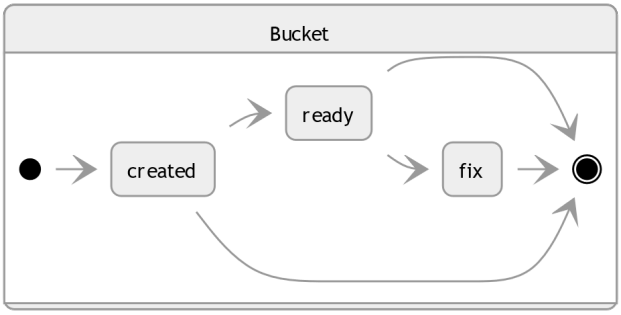

Figure 4. The main status flow for one bucket.

of the newly arrived operation dissolve into their operations (see lines 14 to 16). These operations are rescheduled during the next loop. Then, the bucket manager calculates for all remaining buckets whether the new operation still fits in the bucket scope or not and returns a set of fitting buckets (see line 21). If at least one fitting bucket can be found, the bucket manager chooses the bucket with the smallest earliest start time to assign the operation to the bucket (see lines 22 to 24). If no fitting bucket can be found, a new bucket will be created for the operation (line 26).

After assigning an operation to a bucket, this newly created or modified bucket can be scheduled at any of the capable resources. Scheduled buckets can still be rescheduled in case the bucket is modified afterwards.

D. Releasing buckets to production

Crucial for the event-driven production, we use a releasing mechanism to process buckets on shop floor. After assigning the bucket to a machine, the bucket remains in a planning queue in front of the machine. The machine picks the next bucket from the planning queue using the Least Slack Time (LST) rule. LST is known to be the best due date criteria based rule for timeliness and throughput [17]. Of course this rule can be replaced with any other priority rule, but that is not part of this contribution. Our approach includes the setup time into the calculation of the LST.

All buckets are queued regardless of their precondition. It is likely that a bucket contains operations that are not ready to be processed as their materials are not yet in stock or preceding operations of the same job are not yet completed. We solve the issue by giving buckets a state. When operations of a bucket receive material and their predecessors are completed, we set the state of the bucket to *ready*. Once setting the bucket *ready*, we enable the machine to select the bucket from the planning queue in front of the machine. At this time, new operations can still be inserted into the bucket. The bucket reaches the status *fix* when the machine selects the bucket as the next bucket to be processed. At that moment, all operations of the bucket without satisfied preconditions, are sent back to the bucket manager and trigger the algorithm in 2 to find or create a new bucket. Moreover, it is not possible to insert new operations into the bucket in status *fix*. The transitions between the states are shown in Figure 4.

After the resource chooses the bucket and sets its status to *fix*, the resource organizes the handling of the operations from the bucket. In general the operations in the bucket are unsorted. However, the operations of the bucket are processed applying again the LST-rule.

IV. Adaptation of SOBA to our event-driven production

ACATECH promotes self-organizing production systems with ”intelligent” resources and ”intelligent” materials in their Industry 4.0 concepts to meet all upcoming challenges of future production [1], [18]. In our self-organizing production, resources and materials are agents and can be physically attached to the product or have a virtual representation known as digital twin [19]. Our agents communicate asynchronously and cooperate based on an event-driven system [20]. Thus, we created a robust and reactive multi-agent production that we named SOPA - ’Self Organizing Production Architecture’ shown in Figure 5. Each symbol represents one type of agent. Each arrow is a communication path between two types of agents. In the text below, the numbers (#number) refer to the communication paths. The letters in brackets describe the type of the agents according to their lifetime: (P) for persistent and (T) for transient. [21] introduced the self organizing production in detail. Therefore, it is only described in short here.

The ”intelligent” resources and materials of Industry 4.0 suggest multi-agent systems (MAS) to realize self-organization in production. Figure 5 shows the agents and their communication. At the beginning there are the supervisor agent (SA), the warehouse agents (WA), the imparting agents (IA), the resource agents (RA), and the hub agents (HA). The SA is responsible for communication with the system’s environment. The HA manages a resource group. The IA imparts knowledge about resource capabilities between HA and production agents (PA). When a customer order arrives, the SA creates a new contract agent (CA) (1). For each sales order item, the CA creates a disposition agent (DA) (2) that represents the product of the sales order item. Each DA asks the corresponding WA if the material is available (3). If the material is not available and is manufactured externally, the WA transmits a purchase order for the required material via the SA (4). Then the WA informs the DA about the availability (5) of the requested material. If

Copyright (c) IARIA, 2020. ISBN: 978-1-61208-781-8

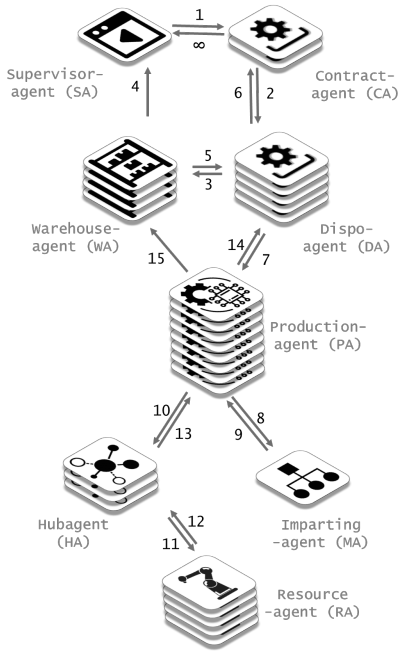

Figure 5. Architecture of our multi-agent-based self-organizing production

the material is available, the DA send a completion notification (6) to the CA. Otherwise, the DA creates a PA for the material (7). The PA creates a new DA for each of its BOM items, and the process described starts again until all BOM elements have been completely exploded. During BOM explosion, a backward scheduling is carried out. Each PA requests an HA from the IA for each of the PA's containing operations (8), and the IA returns the responsible HA (9). Now, the PA sends a request to the HA (10) for the next time slot in which the resources can execute the operation. The HA forwards the request to each of its RAs (11). Each RA calculates a start time based on his current workload and returns it as an offer to the HA (12), which chooses the best offer, sends a confirmation to the corresponding RA, and informs the requesting PA of the start time of the operation (13). The RA queues the operation according to its start time, removes those operations from the queue that are delayed by the newly queued operation, and asks the HA to reschedule these operations (12), as this enables other RAs to submit better offers. If the processing time of an operation deviates from its expected duration, the same mechanism reorganizes subsequent operations so that the self-organizing production always reacts to deviations immediately, which is one of its superior advantages. After a resource carried out an operation, its RA sends a completion message to the HA (12), which forwards the message to the PA of the operation (13), whereupon the PA sets the status of the following operation to *"ready"*, so that the assigned resources can start processing the operation. If no subsequent operation exists, the PA sends its completion message to its DA (14) and informs the WA about incoming material (15). The WA supplies that DA with the material (5) which needs it most urgently. After all DAs of the PA have been received their materials, the production of the PA starts and the described process starts again until the end product is finished and the sales order can be fulfilled.

V. Empirical Study

A. Structure of the simulation model

In order to test SOBA, we use simulation model of a production with three machine types as shown in Figure 6. The main production flow involving two cutting machines, one drilling machine, and two assembly units. The production allows different routings for every material. The drilling machine is the bottleneck. Each machine is able to yield at least two different kind of tools. The production is able to produce two types of wooden toy trucks. The bill of material of each product is three levels deep and contains approximately 30 materials, were each material is either produced or purchased. Each material to be produced is manufactured in up to three operations. The operations of one material can seize a machine more than once.

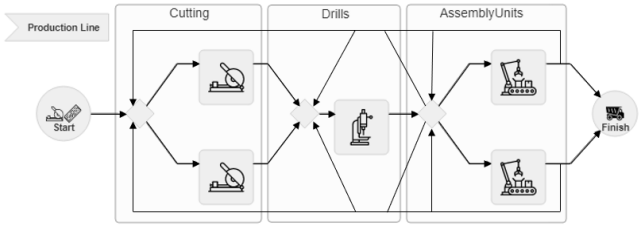

Figure 6. Production line of our simulation model

The operations sum up to 20 per product. Transportation times are not considered yet, but can be represented by an additional operations assigned to the material. In our empirical study, we compare SOBA with two different approaches and thus compare three approaches in total:

1) **Default:** No rules for setup time improvement are applied. Therefore, the resources do not group any operations and only process operations from the queue applying shortest slack time rule.
2) **Stacked:** All operations in the resource queue which are matching the current setup and having all preconditions satisfied, will be processed before the resource decides which group technology to setup next. The selection of the next technology and the selection of the next operation both apply the shortest slack time rule.
3) **Bucket (SOBA):** In order to determine an appropriate value for a maximum bucket size we increase the value step by step. To take the proportions of usage of group technologies into account and to force a more frequent change of group technology on the resources, we use different proportions based on the average required usage in our production. The proportions per category are shown in Table III. For instance, for all resource groups, we define the same overall bucket size of 1440 time units. The maximum bucket size for each group technology is

determined by proportion of its usage shown in Table III. Therefore we calculate: $1440 * 50\% = 720$ and gain the maximum bucket size for the screwdriver in resource group assembly. To ensure a reasonable bucket creation, the maximum bucket size cannot drop below 60 minutes, so at least 4 operations with the maximum processing duration of 15 minutes can join a bucket. Creating and processing buckets with only one operation should be possible to ensure fast processing of urgent operations. Therefore we do not set a minimum bucket size.

TABLE II
SIMULATION PARAMETERS

Value	Unit	Description
28	days	simulation end time
2	days	settling time
36	hours	average time from order placement to delivery
20	%	deviation of estimated operation processing time
80	%	target machine utilization

TABLE III
PROPORTIONS OF USAGE OF GROUP TECHNOLOGIES IN BUCKET MANAGER

Category	Group technology	Proportion Group Usage
Cutting	Blade Big	50%
	Blade Small	50%
Drill	Drill Head M4	25%
	Drill Head M6	75%
Assembly	Screwdriver	50%
	Holding	33%
	Hammer	17%

During the simulation of the production, new customer orders arrive continuously at the production. The inter-arrival time of customer orders is exponentially distributed as suggested in [22] and [23]. We choose an inter-arrival time, which leads to a well-utilized production but does not cause an overload. The production runs for four weeks, 24 hours a day. The production will produce approximately 33 products per day if the processing times do not deviate. To examine the flexibility of the production, we vary the processing times of the operations. According to [22], processing times are distributed log-normally. The inter-arrival times, the processing times, the capacity of the machines as well as the duration of the simulation can be configured separately for each simulation run. All simulation parameters of our empirical study are listed in Table II. However, the production have to dynamically adapt upon fluctuating effects. Those effects are caused by varying arrival rates and delivery dates of orders as well as deviations in processing time of operations. The production reaches its steady state after approximately 24 hours. We add another 48 hours before we start the measurement of the KPIs. All our simulations run with a deviation of processing times of ± 20 % and an average time to delivery of 36 hours.

B. Simulation results

Table IV, *Bucket* shows overall good results. In comparison, *Bucket* is superior to *Stacked* in work in progress and average setup time, which indicates less capital commitment and fewer setups. However, using *Bucket* with unlimited bucket size creates significant bigger buckets which lead to the expected phenomenon of potentially blocking bottlenecks in the production. Thus, our results indicate that finding a suitable maximum bucket size is crucial for SOBA. By reducing the maximum bucket size, we obtain smaller buckets which force the machine to setup different group technologies more frequently. Consequently the setup time increases significantly. Moreover, using a suitable bucket size such as *Bucket 1200* leads to better results in terms of throughput time. Generally for our simulation model, bucket sizes from *Bucket 720* to *Bucket 960* are superior to *Stacked* in terms of work in progress and setup times, while delivering almost the same results at timeliness and throughput time. *Default* as our reference behaviour is inferior in all KPIs. Although *Default* results in the smallest amount of work in progress and therefore less capital commitment, the simulation run for four weeks only completes 652 orders in total, which is by far the worst result for completed orders regarding all experiments. However, using *Stacked* already leads to significant better results. These results are consistent with [11], which demonstrates that exhaustive rules are superior to non-exhaustive rules in terms of timeliness.

VI. CONCLUSION

The target of our research was to develop a concept to group operations requiring the same group technology and prove SOBA empirically. We developed the concept and implemented a prototype of SOBA in our self-organizing production. Then, we simulated a four week production cycle and were able to prove the viability of SOBA. The results show that the production system is still as robust as before in terms of disruptions and deviations. We were also able to reduce the work in progress, throughput times and average setup times by completing the same amount of orders. The trade off for these savings is a slightly lower timeliness. For now, our approach is limited to a single scenario with focus on discrete manufacturing. In general, the concept can be adapted to any production size in terms of machines, setups and production structure. To cope with this, we will investigate further simulation parameters like different priority rules, load dependent delayed production starts and different bucket limitations. Furthermore, we will experiment with larger production models to gain more general results and afterwards transfer the solution to a real world scenario with production data from different cooperating companies.

ACKNOWLEDGEMENT

The authors acknowledge the financial support by the German Federal Ministry of Education and Research within the

TABLE IV
TEST SCENARIO FOR BUCKET

Behaviour	Timeliness	Work in Progress	Throughput time	Average machine utilization time	Average machine setup time	Completed orders	Comparison to Stacked		
							△ Timeliness	△ Work in Progress	△ Average setup time
Default	21.00%	25.81	6882.27	52.00%	41.60%	652	-79.29%	-65.49%	118.95%
Stacked	99.28%	92.43	476.72	78.82%	18.74%	931	0.00%	0.00%	0.00%
Bucket 600	92.64%	83.08	483.84	78.94%	15.26%	931	-6.69%	-10.12%	-18.57%
Bucket 720	96.09%	85.37	503.93	78.82%	15.24%	931	-3.21%	-7.64%	-18.68%
Bucket 840	95.35%	82.87	509.79	78.94%	15.02%	931	-3.96%	-10.34%	-19.85%
Bucket 960	96.63%	79.41	522.64	78.94%	17.60%	931	-2.67%	-14.09%	-6.08%
Bucket 1080	94.35%	80.58	520.21	78.96%	14.80%	931	-4.97%	-12.82%	-21.02%
Bucket 1200	94.65%	80.07	506.82	78.96%	15.04%	931	-4.66%	-13.37%	-19.74%
Bucket 1320	94.49%	81.73	509.93	79.08%	14.64%	931	-4.83%	-11.57%	-21.88%
Bucket 1440	94.38%	83.48	493.03	78.98%	14.84%	931	-4.93%	-9.68%	-20.81%
Bucket 1920	93.63%	83.29	448.69	79.06%	14.74%	931	-5.69%	-9.88%	-21.34%
Bucket 2400	91.42%	83.59	443.22	79.34%	14.52%	931	-7.92%	-9.57%	-22.52%
Bucket ∞	88.68%	77.02	448.27	79.94%	14.14%	931	-10.68%	-16.68%	-24.55%

funding program "Forschung an Fachhochschulen" (contract number: 13FH133PX8).

REFERENCES

[1] acatech, *Recommendations for implementing the strategic initiative INDUSTRIE 4.0 (Abschlussbericht Industrie 4.0)*. Munich: acatech, 2013.

[2] S. C. Kim and P. M. Bobrowski, "Impact of sequence-dependent setup time on job shop scheduling performance," *International Journal of Production Research*, vol. 32, no. 7, pp. 1503–1520, 1994.

[3] A. M. Spence and E. L. Porteus, "Setup reduction and increased effective capacity," *Management Science*, vol. 33, no. 10, pp. 1291–1301, 1987.

[4] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[5] W. Domschke, A. Scholl, and S. Voß, *Production planning: aspects of industrial engineering (Produktionsplanung: Ablauforganisatorische Aspekte)*, 2nd ed., ser. Springer-Lehrbuch. Berlin: Springer, 1997.

[6] F. Herrmann, *Operational planning in IT systems for production planning and control(Operative Planung in IT-Systemen für die Produktionsplanung und -steuerung)*. Wiesbaden: Vieweg+Teubner, 2011.

[7] I. Ham, K. Hitomi, and T. Yoshida, *Group technology: Applications to production management*, ser. International series in management science/operations research. Boston: Kluwer-Nijhoff, 1985.

[8] R. W. Brennan, "Modeling and analysis of manufacturing systems. isbn 0-417-51418-7 [pp.461]," *International Journal of Computer Integrated Manufacturing*, vol. 8, no. 2, pp. 155–156, 1995.

[9] C. T. Ng, T. C. E. Cheng, A. Janiak, and M. Y. Kovalyov, "Group scheduling with controllable setup and processing times: Minimizing total weighted completion time," *Annals of Operations Research*, vol. 133, no. 1-4, pp. 163–174, 2005.

[10] A. Klausnitzer, J. S. Neufeld, and U. Buscher, *Scheduling dynamic job shop manufacturing cells with family setup times: a simulation study*. U. Logist. Res., 2017.

[11] G. V. Frazier, "An evaluation of group scheduling heuristics in a flow-line manufacturing cell," *International Journal of Production Research*, vol. 34, no. 4, pp. 959–976, 1996. accessed: November 2019. [Online]. Available: https://www.tandfonline.com/doi/pdf/10.1080/00207549608904945

[12] R. A. Ruben, C. T. Mosier, and F. Mahmoodi, "A comprehensive analysis of group scheduling heuristics in a job shop cell," *International Journal of Production Research*, vol. 31, no. 6, pp. 1343–1369, 1993.

[13] B. Grabot and L. Geneste, "Dispatching rules in scheduling dispatching rules in scheduling: a fuzzy approach," *International Journal of Production Research*, vol. 32, no. 4, pp. 903–915, 1994.

[14] D.-J. van der Zee, G. J. Gaalman, and G. Nomden, "Family based dispatching in manufacturing networks," *International Journal of Production Research*, vol. 49, no. 23, pp. 7059–7084, 2011. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00669040/document

[15] G. Egilmez, E. M. Mese, B. Erenay, and G. A. Süer, "Group scheduling in a cellular manufacturing shop to minimise total tardiness and nt: a comparative genetic algorithm and mathematical modelling approach," *International Journal of Services and Operations Management*, vol. 24, no. 1, p. 125, 2016.

[16] D. Giglio, "A milp model for single machine family scheduling with sequence-dependent batch setup and controllable processing times." [Online]. Available: http://arxiv.org/pdf/1501.07396v2

[17] H. Corsten and R. Gössinger, *Production management (Produktionswirtschaft): Introduction to industrial production management(Einführung in das industrielle Produktionsmanagement)*, 13th ed., ser. Lehr- und Handbücher der Betriebswirtschaftslehre. München: Oldenbourg, 2012.

[18] C.-C. Kuo, J. Z. Shyu, and K. Ding, "Industrial revitalization via industry 4.0 – a comparative policy analysis among china, germany and the usa," *Global Transitions*, vol. 1, pp. 3–14, 2019.

[19] R. Stark and T. Damerau, "Digital twin," in *CIRP Encyclopedia of Production Engineering*, S. Chatti and T. Tolio, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, vol. 66, pp. 1–8.

[20] G. Di Marzo Serugendo et al., "Self-organisation: Paradigms and applications," in *Engineering Self-Organising Systems*, ser. Lecture Notes in Computer Science, G. Goos et al., Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, vol. 2977, pp. 1–19.

[21] T. Munkelt and M. Krockert, "Agent-based self-organization versus central production planning," in *2018 Winter Simulation Conference (WSC)*. [Piscataway, NJ]: IEEE, 2018?, pp. 3241–3251.

[22] G. Zäpfel and R. Braune, *Moderne Heuristiken der Produktionsplanung: Am Beispiel der Maschinenbelegung*, ser. WiSo-Kurzlehrbücher Reihe Betriebswirtschaft. München: Vahlen, 2005.

[23] J. Košturiak and M. Gregor, Eds., *Simulation of production systems (Simulation von Produktionssystemen)*. Vienna: Springer Vienna, 1995.

Copyright (c) IARIA, 2020. ISBN: 978-1-61208-781-8