Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

A BIASED-RANDOMIZED DISCRETE-EVENT HEURISTIC FOR THE HYBRID FLOW SHOP PROBLEM WITH BATCHING AND MULTIPLE PATHS

Christoph Laroque
Madlene Leißau

Institute for Management and Information
University of Applied Sciences Zwickau
Kornmarkt 1
Zwickau, 08056, GERMANY

Pedro Copado
Javier Panadero
Angel A. Juan

IN3 – Computer Science Dept.
Universitat Oberta de Catalunya
Euncet Business School
Barcelona, 08018, SPAIN

Christin Schumacher

Informatik 4 – Modeling and Simulation
TU Dortmund University
Department of Computer Science
D-44221 Dortmund, GERMANY

ABSTRACT

Based on a real-life use-case, this paper discusses a manufacturing scenario where different jobs be processed by a series of machines. Depending on its type, each job must follow a pre-defined route in the hybrid flow shop, where the aggregation of jobs in batches might be required at several points of a route. This process can be modeled as a hybrid flow shop problem with several additional but realistic restrictions. The objective is to find a good permutation of jobs (solution) that minimizes the makespan. Discrete-event simulation can be used to obtain the makespan value associated with any given permutation. However, to obtain high-quality solutions to the problem, simulation needs to be combined with an optimization component, e.g., a discrete-event heuristic. The proposed approach can find solutions that significantly outperform those provided by employing simulation only and can easily be extended to a simheuristic to account for random processing times.

1 INTRODUCTION

An intensification of global business, an ongoing digitization, and a growing demand for customized products are shaping the competitive situation for manufacturing companies today. An economically optimal utilization of the production chains was often considered the essential goal in the past. During the last years, however, this goal has increasingly shifted towards a more customer-oriented production. Therefore, the main focus is now on the timely fulfillment of promised delivery dates by the respective production company. Accordingly, operational production planning deals with questions around the latest possible release of a certain batch, so that it can still be manufactured and delivered on time. A high and constantly increasing complexity of production systems, in conjunction with a high degree of automation, repeatedly pose challenges for the respective production company. This is specially the case in the field of semiconductor manufacturing. The use of simulation-optimization approaches (Amaran et al. 2016)

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

could provide a more differentiated answer. In planning, design, and ramp-up, these methods are well established, but are not frequently used yet for operational support in decision-making.

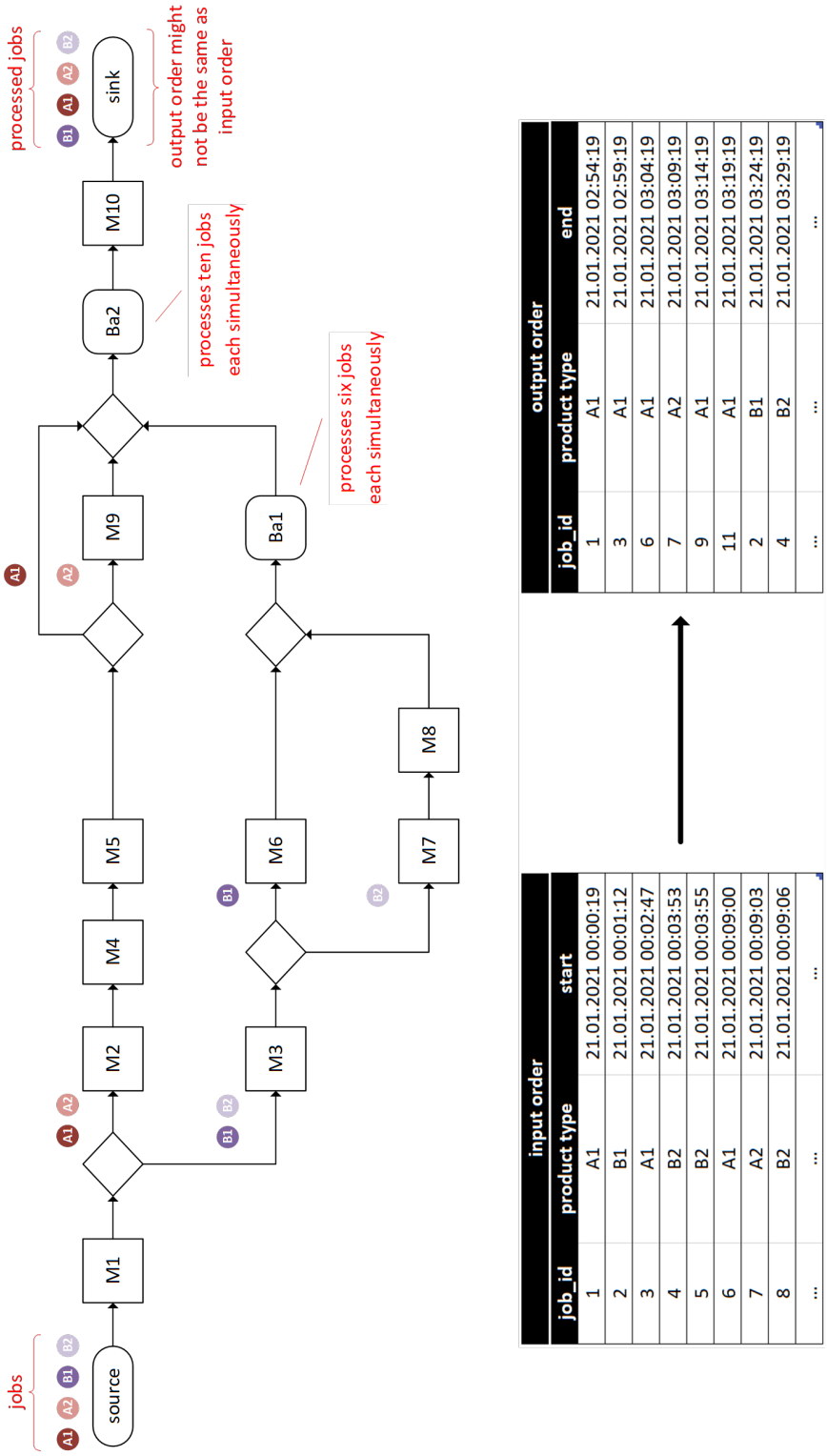
In real-life applications of scheduling, it is possible for jobs to take different paths through the production system. The specific path might depend upon a particular specification or parameter, which defines the machines to be visited by the job. The existence of different paths through the production system might cause the order in which the jobs leave the system to be different from the order in which the jobs entered it. Consequently, this can lead to problems in the context of the planning of batch-related insertion dates, especially if batch processes are also found within the production system. In this paper, a multi-path version of the permutation flow shop scheduling problem (Tosun et al. 2020) is analyzed. This version also considers two batch processes at the same time, and it is based on a real case from a German manufacturing industrial partner.

Accordingly, the main contributions of this work can be stated as follows: *(i)* the modeling of a permutation flow-shop scheduling problem with different pre-determined paths, which depend upon a particular specification or parameter; *(ii)* an original discrete-event-driven heuristic, which allows us to deal with the complexity of the system in a natural way; *(iii)* its extension into a biased-randomized algorithm; *(iv)* a review of related work in the existing scientific literature; and *(v)* a series of computational experiments that illustrate the application of the proposed methodology for solving such a challenging flow-shop scenario. Discrete-event driven heuristics combine concepts of discrete-event simulation to guide a constructive heuristic. They have been employed in problems where synchronization issues where relevant (Fikar et al. 2016). Biased-randomized algorithms allow us to obtain alternative solutions based on a constructive heuristic (Juan et al. 2009). In a nutshell, they make use of Monte Carlo simulation and skewed probability distributions to introduce a non-uniform random behavior into a constructive heuristic. Thus, the heuristic is transformed into a more powerful probabilistic algorithm, which can be run in virtually the same wall-clock time as the original heuristic if parallelization techniques are employed. Some applications of these algorithms include flow shop problems (Ferrer et al. 2016), integrated routing and facility-location problems (Quintero-Araujo et al. 2017), and vehicle routing problems (Dominguez et al. 2016; Belloso et al. 2019). To the best of our knowledge, this is the first time that both techniques are hybridized to solve a complex flow shop problem as the one analyzed here.

The remainder of the paper is organized as follows. Section 2 provides a more detailed description of the scheduling problem studied in this paper. Section 3 reviews related work on similar flow shop problems. Section 4 proposes a novel optimization heuristic that incorporates concepts from discrete-event simulation. Section 5 carries out a series of numerical experiments to test our methodology. Section 6 analyzes the obtained results. Section 7 discusses some preliminary managerial insights derived from our work. Finally, Section 8 summarizes the main findings of this paper and points out some future research lines.

2 A DETAILED DESCRIPTION OF THE PROBLEM

Figure 1 shows an example of a production scenario in which different jobs have to be processed by a number of machines. Following this, jobs arrive in the system at the source on the left side, are processed in machine *M1* and are subsequently forwarded to various predefined paths depending on their product type. Accordingly, jobs of product types *A1* and *A2* pass through machines *M2*, *M4*, and *M5* before jobs of product type *A2* have to pass through machine *M9* as well. In contrast, jobs of product types *B1* and *B2* are both processed in machine *M3* before the processing paths differ again depending on the product type. While the jobs of product type *B1* are processed in machine *M6*, the jobs of product type *B2* pass through machines *M7* and *M8*. The jobs of both product types *B1* and *B2* then enter a batch process *Ba1*, which processes six jobs each – regardless of their product type – simultaneously. The same applies subsequently in a batch process *Ba2*, which processes ten jobs simultaneously and includes the jobs of all four product types before the jobs are finally processed in machine *M10* and then leave the production system.



input order		
job_id	product type	start
1	A1	21.01.2021 00:00:19
2	B1	21.01.2021 00:01:12
3	A1	21.01.2021 00:02:47
4	B2	21.01.2021 00:03:53
5	B2	21.01.2021 00:03:55
6	A1	21.01.2021 00:09:00
7	A2	21.01.2021 00:09:03
8	B2	21.01.2021 00:09:06
...	...	...

output order		
job_id	product type	end
1	A1	21.01.2021 02:54:19
3	A1	21.01.2021 02:59:19
6	A1	21.01.2021 03:04:19
7	A2	21.01.2021 03:09:19
9	A1	21.01.2021 03:14:19
11	A1	21.01.2021 03:19:19
2	B1	21.01.2021 03:24:19
4	B2	21.01.2021 03:29:19
...	...	...

Figure 1: An example of a production scenario with multiple paths and different product types.

The fact that jobs are processed on different predefined paths, depending on their product type, makes it possible for jobs to overtake others during processing, i.e.: the order in which the jobs leave the system might differ from the order in which the jobs were placed in the system. The lower part of the figure contains a first comparison of the order in which the jobs enter the system and the order in which the jobs leave it. For example, *job 1* enters the system at *position 1* and leaves at *position 1*, while *job 2* enters the system at *position 2* and leaves at *position 7*. The aim of the methodology described below is to find a permutation of jobs (solution) that reduces the makespan, which depends on the order in which the jobs are placed in the system.

3 RELATED WORK

Since Johnson (1954) published the first approach solving flow shops, a large number of flow shops with and without parallel machines per stage have been analyzed in the literature. Flow shops with multiple machines on at least one of the processing stages are defined as flexible or hybrid flow shops (Ruiz and Vázquez-Rodríguez 2010; Pinedo 2016). Ribas et al. (2010) and Lee and Loong (2019) show that hybrid flow shop problems of different application areas can be modeled as mixed integer programming problems and solved using exact methods, heuristics, or even simulation. Gupta (1988) proves that even a two-stage hybrid flow shop with only two identical parallel machines on one of both stages, and one machine on the other stage, is *NP-hard*. So it can be assumed that the presented problem is also *NP-hard*. In contrast to exact optimization methods, heuristics cannot verify the optimality of a solution. Rather, they can provide approximated solutions. However, solving real-world problems using heuristics is much less time-consuming than solving exact mathematical models for these complex real-world problems. Heuristics can also be used as an iterative way of exploring the solution space and attempt to improve initial solutions with respect to the objective function (Naderi et al. 2010).

Configurations of (hybrid) flow shops that include machines needing different types of jobs to start their production are often categorized as assembly flow shops (Komaki et al. 2019; Nikzad et al. 2015). On the contrary, (hybrid) flow shops that include machines that process more than one job of the same type simultaneously (see machine Ba1 and Ba2 in Figure 1) are mostly categorized as hybrid flow shops with batching machines (Morais et al. 2013). Literature overviews of assembly flow shops are provided by Komaki et al. (2019) and Nikzad et al. (2015). Morais et al. (2013) has specialized his literature review on the integration of batching components in hybrid flow shops. As we focus on the makespan objective in our paper, Komaki et al. (2019) and Lee and Loong (2019) show that makespan and time-based objectives are dominating in their analyzed papers of hybrid flow shops. Hence, it can be assumed that the makespan is one of the most relevant objectives in both applications and theory.

In the following, some examples of studies with hybrid flow shops including batching machines and the makespan objective are presented. Wilson et al. (2004) use a genetic algorithm for a hybrid flow shop with batching machines. In their problem, jobs can also skip stages – like in our problem formulation (see Figure 1). To generate an initial solution for their genetic algorithm, they partly use a combination of a longest processing time algorithm for the permutation. The following stages are scheduled using the earliest completion time concept. Logendran et al. (2006) also consider a hybrid flow shop in which not every stage has to be visited by every job. They develop a tabu search for their problem. In each iteration, two jobs are swapped. The tabu list consists of the pairs of jobs exchanged in the previous steps of the algorithm, so that they are not reused in the following steps. These authors compare three constructive heuristics for generating an initial solution for their tabu search algorithm. The first heuristic orders the jobs numerically or alphabetically, and schedules them in order on the machines that become available. The other two heuristics use variations of longest processing times, with the sum of the processing times over all stages per order. However, in comparison, none of the constructive algorithms yields significantly better results than the others. He et al. (2007) analyze a shortest processing time heuristic with polynomial runtime for a two-stage hybrid flow shop problem. A specificity of the problem is that jobs can only be processed by one dedicated machine during the first stage. In the second stage, all jobs are processed

on the same machine in batches. In their heuristics, they work in the first processing stage according to the shortest processing time. In the second processing stage, they employ a combination of the earliest completion time and a minimization of the processing times for all batches.

Due to the high complexity of the problem considered in our paper, a large number of possible permutations needs to be explored. Therefore, although simulation can be useful for modeling the framework, it needs to be combined with optimization components in order to generate high-quality solutions.

4 A DISCRETE-EVENT HEURISTIC

Figure 2 shows a conceptual schema of the solving approach. It relies on a multi-start framework that iteratively calls a biased-randomized heuristic inspired by the well-known NEH heuristic for the permutation flow shop scheduling problem (Nawaz et al. 1983).

The NEH heuristic works as follows: *(i)* consider each job as if it were the only one in the system, and compute the total time it requires to process the job by all the machines in its path; *(ii)* create a list of jobs and sort it from higher to lower total times; *(iii)* iteratively select the next job from the sorted list, and allocate it in each of the possible locations of the emerging solution (permutation of jobs) to find out which is the best position for the new job. Particularly, we have used the idea of processing the jobs from higher to lower total times, due to its providing good results in terms of makespan. Additionally, we have extended this technique by introducing a balancing mechanism, which modifies the job-selection order based on the type of job and on the current status of each path – this avoids collapsing any of the paths while the other is idle. In our case, the balancing mechanism consists in simply selecting, from the sorted list, the next job that can be routed to the path containing the lowest number of jobs at the time of the selection. Moreover, in order to introduce a non-uniform randomization into the constructive heuristic, we apply biased-randomization (BR) techniques during the job-selection process. These techniques make use of skewed probability distributions to transform a deterministic heuristic into a probabilistic algorithm able to generate many 'good-quality' solutions to the problem in short computational times. In our case, a geometric probability distribution, driven by a single parameter $\alpha(0 < \alpha < 1)$, is used to induce this skewed behavior. Among other applications, these techniques have been recently applied in solving different flow-shop scheduling problems (Ferrer et al. 2016), vehicle routing problems (Belloso et al. 2019; Dominguez et al. 2016), capacitated location routing problems (Quintero-Araujo et al. 2017), and even to extend and enhance traditional metaheuristic frameworks (Ferone et al. 2019).

One of the main challenges of this problem is to compute the makespan in a multi-path flow shop scenario with batch processing at some machines. Notice that this computation is not straightforward due to the fact that some jobs must be oriented towards the right path at several cross-paths, and also that some machines might need to wait until all jobs in a batch become available. Hence, in order to compute the makespan associated with a given solution (permutation of jobs) generated by the constructive biased-randomized algorithm, we employ a discrete-event deterministic simulation. Three types of events are considered: *(i)* the starting of a job i in a machine j at time t, denoted as $(t,i,0,j)$: *(ii)* the ending of a job i in a machine j at time t, denoted as $(t,i,1,j)$; and *(iii)* the arrival of a job i to a cross-path j at time 5, denoted by $(t,i,2,j)$. The discrete-event list is initialized by considering the ending events generated by the list of jobs $(1,2,\ldots,n)$ that reach the first machine or cross-path node. Then, the simulation clock is started and the next event in the list is extracted from it and processed. On the one hand, ending events might free the corresponding machine and also schedule new starting events for the associated jobs in case they have not completed the whole sequence of machines yet. On the other hand, starting events will block the required machine and will also generate new ending events associated with the starting ones. Finally, whenever a job reaches a cross-path node, an event is triggered and, depending of the type of the job, a new starting time in a machine is scheduled. This generation of new events continues until the last job is processed by the last machine, which defines the solution makespan.

The entire process describe above is repeated in a multi-start framework (Listing 1) insofar as the elapsed time does not exceed the maximum computational time allowed by the manager (or any other

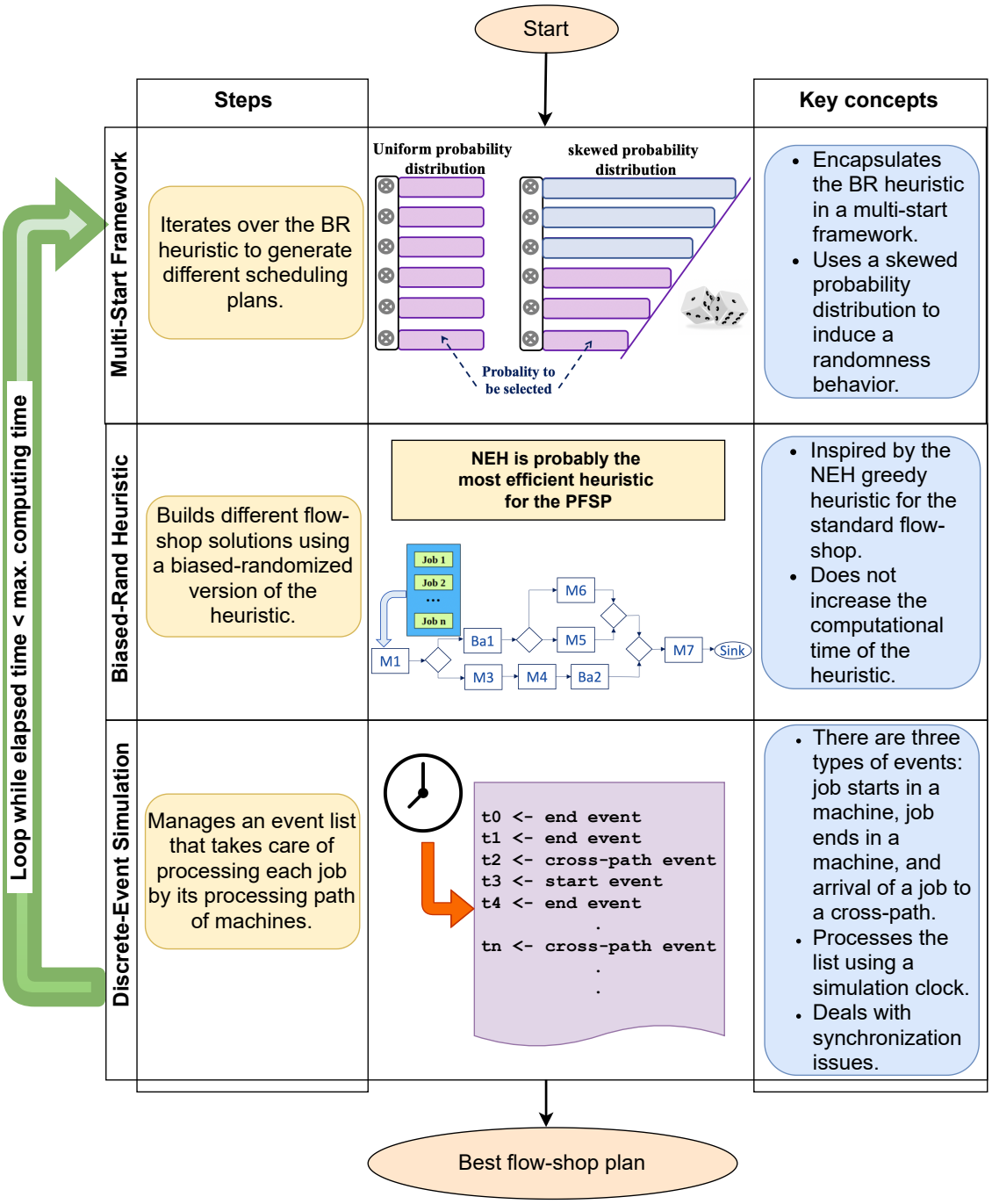


Figure 2: Conceptual schema of the solving approach.

```
def MultiStart(inputs,strategy,maxTime,beta1,beta2):
    #inputs: instance
    #beta1, beta2: range of beta's for BR
    #maxTime: maximum execution time in seconds
    #strategy: 0 --> FIFO, 1 --> TT

    start = ElapsedTime.systemTime()
    elapsed = 0.0

    bestSol = br_heuristic(inputs, strategy, beta1, beta2, False) # greedy heuristic (
    BR is switched-OFF)
    print("Initial Solution Makespan:",bestSol.getCosts())

    while elapsed < maxTime:
        newSol = br_heuristic(inputs, strategy, beta1, beta2, True) # BR is swiched ON
        if newSol.getCosts() < bestSol.getCosts():
            bestSol = copy.deepcopy(newSol)
            elapsed = ElapsedTime.calcElapsed(start, ElapsedTime.systemTime())
            print("New Best Solution Makespan:",bestSol.getCosts())
        elapsed = ElapsedTime.calcElapsed(start,ElapsedTime.systemTime())
    return bestSol
```

Listing 1: MultiStart Framework

finishing criterion). Because of the biased-randomization effect, each iteration is likely to provide a different solution. Once this maximum time is reached, the best solution found (the one with the lowest makespan) is returned by the algorithm.

5 COMPUTATIONAL EXPERIMENTS

The proposed algorithm has been implemented using Python 3.7 and tested on a workstation with a multi-core processor Intel Xeon E5-2650 v4 with 32GB of RAM. To the best of our knowledge, there are no public instances for the proposed problem. Hence, we have generated a set of instances, which are publicly available at http://dx.doi.org/10.13140/RG.2.2.12173.05601. This set is based on the production scenario defined in Section 2, which is composed of 9 processing machines and 2 additional batch machines, which are distributed in 4 different paths. A total of 4 types of jobs have been considered, each type being processed by a specific path. The total number of jobs varies from 200 to 5040 – depending on the instance –, and each job j has associated a processing time of p_{ij} units at each machine i. Note that, each instance has been performed 2 hours and the determination of the best production sequence for all machines needs the exploration of $n!$ sequences – the number of possible job permutations at each machine. Thus, the higher the number of jobs that compose the instances, the more challenging to find optimal or near-optimal solutions to the problem.

Table 1 displays the results for some instances with different number of jobs. The first column of the table identifies the instances. Each instance is characterized by the following nomenclature *ixx_j.m.t.b*, where: *ixx* is a sequential identifier used to identify the instance in an easy and comprehensive way; j is the total number of jobs to be processed; j is the total number of machines; t is the different types of jobs, which match the number of total number of different paths of the machine; and b is the total number of batch machines. Notice that the number of batch machines is included in the total number of machines (parameter m). Subsequently, the next four columns display the obtained results, considering the different sorted strategies used in our algorithm to compute the makespan. Thus, the second column of the table presents the computed makespan when the jobs are processed in the system using a first-in-first-out (FIFO) strategy. In this case, the jobs enter in the system without any logical ordering, i.e., following the order as they are in the processing queue. The third column reports the makespan when the jobs are selected using

the previous FIFO strategy combined with biased-randomization techniques. In this sense, this allows us to enter jobs in the system in a different order at each iteration of the algorithm. In our case, a geometric probability distribution, driven by a single parameter α ($0 < \alpha < 1$), is used to induce this skewed behavior. The value for this parameter was set after a quick tuning process over a random sample of instances, establishing a good performance whenever α falls between 0.3 to 0.4 (i.e., any random value inside this interval will generate similar results). Similarly, the next two columns show the computed makespan when a new sorting strategy is employed. This strategy consists in sorting the jobs according to their total processing times (from highest to lowest). This way, the longest jobs are prioritized to be processed at the very beginning of the process. The fifth column shows the computed makespan using this strategy, while the sixth column displays the makespan when this strategy is combined with biased-randomization techniques. Finally, the last three columns of the table report the gaps for the different strategies with respect to the initial FIFO strategy.

Table 1: Results across different scenarios.

	Sorting Strategies						
	Original (FIFO)		**Total Time (TT)**		**Gap (%)**		
#Instance	**Heuristic [1]**	**BR [2]**	**Heuristic [3]**	**BR [4]**	**[1]-[2]**	**[1]-[3]**	**[1]-[4]**
i01_200.4.11.2	2778.5	2760.0	2711.0	2705.0	0.67	2.43	2.65
i02_200.4.11.2	2714.0	2668.0	2605.0	2599.0	1.69	4.02	4.24
i03_200.4.11.2	2798.5	2730.5	2654.0	2641.0	2.43	5.16	5.63
i04_200.4.11.2	2800.0	2744.0	2680.0	2669.5	2.00	4.29	4.66
i05_200.4.11.2	2876.5	2822.0	2747.5	2737.0	1.89	4.48	4.85
i06_400.4.11.2	5112.0	5112.0	5015.0	5005.0	0.00	1.90	2.09
i07_400.4.11.2	5192.0	5146.5	5075.0	5069.0	0.88	2.25	2.37
i08_400.4.11.2	5534.0	5523.0	5429.0	5417.0	0.20	1.90	2.11
i09_400.4.11.2	5307.0	5194.5	5137.0	5131.0	2.12	3.20	3.32
i10_800.4.11.2	11047.0	10976.0	10893.0	10891.0	0.64	1.39	1.41
i11_800.4.11.2	10918.5	10856.0	10789.0	10785.0	0.57	1.19	1.22
i12_800.4.11.2	10870.0	10810.5	10701.0	10697.0	0.55	1.55	1.59
i13_800.4.11.2	10521.0	10436.5	10359.0	10357.0	0.80	1.54	1.56
i14_5040.4.11.2	65900.5	65858.5	65759.0	65755.0	0.06	0.21	0.22
i15_5040.4.11.2	65256.5	65186.5	65115	65104	0.11	0.22	0.23
				Average	0.97	2.38	2.54

6 ANALYSIS OF RESULTS

Figure 3 summarizes the results provided in Table 1. One can notice that the average gap obtained after applying a BR technique on the original FIFO strategy is able to enhance the former in about 1% on the average, with a maximum gap of 2.43% (columns [1]-[2]). Also, the new heuristic, based in sorting by total time, is able to enhance the original heuristic by 2.38% on the average, with a maximum gap of 5.16% (column [1]-[3]). Finally, an average improvement of 2.54% is obtained when adding the biased-randomization technique, with a maximum gap of 5.63% (column [1]-[4]). Notice also that the batch processing machines can reduce the benefits of the BR techniques. This is due to both delays and serialization issues that occur during the batching process. Such a behavior can be observed in the largest instances, i.e., those composed of 5040 jobs, where the use of batch machines are more intensive. When using BR techniques in these instances, the gain with respect to the constructive heuristic is low.

7 MANAGERIAL INSIGHTS

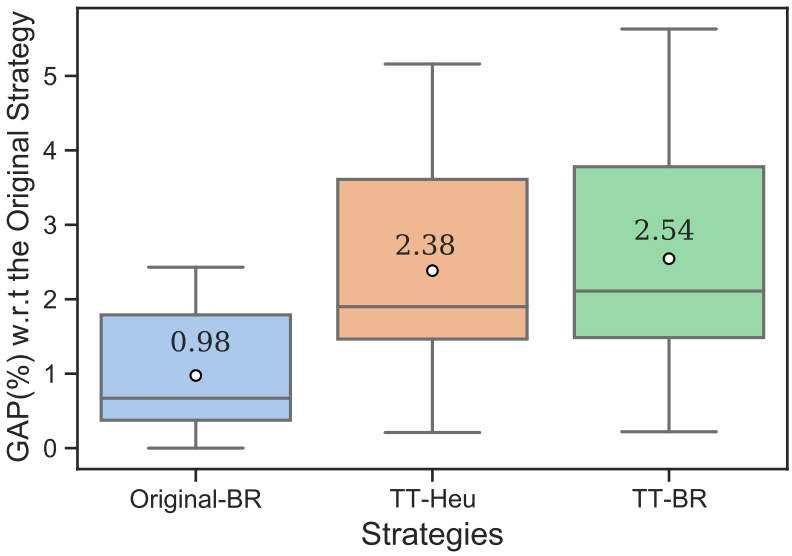


Figure 3: Gaps(%) using the different strategies with respect to the original strategy.

While the flow-shop literature is vast, real-life systems are diverse and they generate rich scheduling problems with special characteristics that, when considered altogether, impose new challenges. The combination of specific assumptions and limitations have seldom been analyzed in previous works, but originate in real-world use-cases with a possible high impact. That is the case, for instance, of flow-shops with multi-server machines and re-entering points. In these complex scenarios, even traditional metaheuristic algorithms might find difficulties in generating reasonably good solutions due to the unpredictable time dependencies generated by the combination of loops and parallel servers. However, discrete-event heuristics – like the one proposed here – can deal with these synchronization issues in a quite natural way and, as shown in the section above, deliver relevant and improved results. Moreover, by incorporating simulation concepts inside a heuristic optimization framework, these simulation-optimization algorithms go beyond classical simulation-alone methods, which lack the capacity to efficiently explore the solution space.

All in all, these simulation-based heuristics may constitute new managerial tools, that allow even operational, data-driven decision making in the manufacturing and process industries. Due to the (in comparison to mathematical optimization) lower computation times for the heuristic approach, an real-life application for operational decision support can be regarded as realistic. Additionally, with the use of stochastic input factors, uncertainty known from the real-world setting can be considered in operational decision making and should lead to better decisions. Here, the simulation-based approach might especially prove beneficial.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we analyze a challenging multi-path permutation flow shop problem which also includes batching. The problem discussed here is inspired on a real-life case, and in order to solve it efficiently we propose a simulation-optimization approach. In particular, our method makes use of a discrete-event driven heuristic, which is also enhanced by employing biased-randomization techniques. The discrete-event driven heuristic makes use of a discrete-event simulation list to schedule and trigger new events as the process goes on. This way, we can compute the makespan associated with any given sorting strategy. When extended into a biased-randomized algorithm, our methodology is capable of providing high-quality solutions in relative computing times. In dynamic and connected production systems, such as the ones we

can find in industry 4.0 environments, this ability to quickly generate a high-quality sorting of jobs might have a noticeable impact on the system performance.

Some future actions that might be considered in order to extend our work are described next: *(i)* to test the algorithm in a real-life manufacturing environment and quantify how it can improve the results obtained by current industrial practices; *(ii)* to consider optimization objectives other than makespan, e.g., to maximize rewards for partially completed jobs, etc.; and *(iii)* to extend the algorithm into a full simheuristic one (Chica et al. 2020), so it can also consider stochastic processing times.

ACKNOWLEDGMENTS

This work has been partially funded by the Spanish Ministry of Science (PID2019-111100RB-C21/AEI/10.13039/501100011033, RED2018-102642-T), and the Erasmus+ Program (2019-I-ES01-KA103-062602). Likewise, this work has been partially funded by the European project iDEV40. The iDev40 project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 783163. The JU receives support from the European Union's Horizon 2020 research and innovation program. It is co-funded by the consortium members, grants from Austria, Germany, Belgium, Italy, Spain, and Romania.

REFERENCES

Amaran, S., N. V. Sahinidis, B. Sharda, and S. J. Bury. 2016. "Simulation Optimization: a Review of Algorithms and Applications". *Annals of Operations Research* 240(1):351–380.

Belloso, J., A. A. Juan, and J. Faulin. 2019. "An Iterative Biased-Randomized Heuristic for the Fleet Size and Mix Vehicle Routing Problem with Backhauls". *International Transactions in Operational Research* 26(1):289–301.

Chica, M., A. A. Juan, C. Bayliss, O. Cordón, and W. D. Kelton. 2020. "Why Simheuristics? Benefits, Limitations, and Best Practices when Combining Metaheuristics with Simulation". *SORT-Statistics and Operations Research Transactions* 44:311–334.

Dominguez, O., A. A. Juan, I. A. De La Nuez, and D. Ouelhadj. 2016. "An ILS-Biased Randomization Algorithm for the Two-Dimensional Loading HFVRP with Sequential Loading and Items Rotation". *Journal of the Operational Research Society* 67(1):37–53.

Ferone, D., A. Gruler, P. Festa, and A. A. Juan. 2019. "Enhancing and Extending the Classical GRASP Framework with Biased Randomisation and Simulation". *Journal of the Operational Research Society* 70(8):1362–1375.

Ferrer, A., D. Guimarans, H. Ramalhinho, and A. A. Juan. 2016. "A BRILS Metaheuristic for Non-Smooth Flow-Shop Problems with Failure-Risk Costs". *Expert Systems with Applications* 44:177–186.

Fikar, C., A. A. Juan, E. Martinez, and P. Hirsch. 2016. "A Discrete-Event Driven Metaheuristic for Dynamic Home Service Routing with Synchronised Trip Sharing". *European Journal of Industrial Engineering* 10(3):323–340.

Gupta, J. N. D. 1988. "Two-Stage, Hybrid Flowshop Scheduling Problem". *The Journal of the Operational Research Society* 39(4):359.

He, L., S. Sun, and R. Luo. 2007. "A Hybrid Two-stage Flowshop Scheduling Problem". *Asia-Pacific Journal of Operational Research* 24(01):45–56.

Johnson, S. M. 1954. "Optimal Two- and Three-Stage Production Schedules with Setup Times Included". *Naval Research Logistics Quarterly* 1(1):61–68.

Juan, A. A., J. Faulin, R. Ruiz, B. Barrios, M. Gilibert, and X. Vilajosana. 2009. "Using Oriented Random Search to Provide a Set of Alternative Solutions to the Capacitated Vehicle Routing Problem". In *Operations Research and Cyber-Infrastructure*, 331–345. Springer.

Komaki, G. M., S. Sheikh, and B. Malakooti. 2019. "Flow Shop Scheduling Problems with Assembly Operations: a Review and new Trends". *International Journal of Production Research* 57(10):2926–2955.

Lee, T.-S., and Y.-T. Loong. 2019. "A Review of Scheduling Problem and Resolution Methods in Flexible Flow Shop". *International Journal of Industrial Engineering Computations*:67–88.

Logendran, R., P. deSzoeke, and F. Barnard. 2006. "Sequence-Dependent Group Scheduling Problems in Flexible Flow Shops". *International Journal of Production Economics* 102(1):66–86.

Morais, M. d. F., M. G. Filho, and T. J. Perassoli Boiko. 2013. "Hybrid Flow Shop Scheduling Problems Involving Setup Considerations: a Literature Review and Analysis". 20(11/12):614–630.

Naderi, B., R. Ruiz, and M. Zandieh. 2010. "Algorithms for a Realistic Variant of Flowshop Scheduling". *Computers & Operations Research* 37(2):236–246.

Nawaz, M., E. E. Enscore Jr, and I. Ham. 1983. "A Heuristic Algorithm for the m-Machine, n-Job Flow-Shop Sequencing Problem". *Omega* 11(1):91–95.

Nikzad, F., J. Rezaeian, I. Mahdavi, and I. Rastgar. 2015. "Scheduling of Multi-Component Products in a Two-Stage Flexible Flow Shop". *Applied Soft Computing* 32:132–143.

Pinedo, M. 2016. *Scheduling: Theory, Algorithms, and Systems*. 5 ed. Cham and Heidelberg and New York and Dordrecht and London: Springer.

Quintero-Araujo, C. L., J. P. Caballero-Villalobos, A. A. Juan, and J. R. Montoya-Torres. 2017. "A Biased-Randomized Metaheuristic for the Capacitated Location Routing Problem". *International Transactions in Operational Research* 24(5):1079–1098.

Ribas, I., R. Leisten, and J. M. Framiñan. 2010. "Review and Classification of Hybrid Flow Shop Scheduling Problems from a Production System and a Solutions Procedure Perspective". *Computers & Operations Research* 37(8):1439–1454.

Ruiz, R., and J. A. Vázquez-Rodríguez. 2010. "The Hybrid Flow Shop Scheduling Problem". *European Journal of Operational Research* 205(1):1–18.

Tosun, Ö., M. Marichelvam, and N. Tosun. 2020. "A Literature Review on Hybrid Flow Shop Scheduling". *International Journal of Advanced Operations Management* 12(2):156–194.

Wilson, A. D., R. E. King, and T. J. Hodgson. 2004. "Scheduling Non-Similar Groups on a Flow Line: Multiple Group Setups". *Robotics and Computer-Integrated Manufacturing* 20(6):505–515.

AUTHOR BIOGRAPHIES

CHRISTOPH LAROQUE studied business computing at the University of Paderborn, Germany. Since 2013 he is Professor of Business Computing at the University of Applied Sciences Zwickau, Germany. He is mainly interested in the application of simulation-based decision support techniques for operational production and project management. His email address is Christoph.Laroque@fh-zwickau.de.

PEDRO J. COPADO-MENDEZ is a Post-doctoral researcher in the ICSO group at the IN3 – Universitat Oberta de Catalunya. He completed his PhD at the University Rovira i Virgili (Spain). His main research interests include metaheuristics, hybrid heuristics and their applications. His email address is pcopadom@uoc.edu.

JAVIER PANADERO is an Assistant Professor of Simulation and High Performance Computing in the Computer Science Dept. at the Universitat Oberta de Catalunya. He is also a Lecturer at the Euncet Business School. He holds a Ph.D. and a M.S. in Computer Science. His major research areas are high performance computing and simheuristics. He has co-authored more than 40 scientific articles. His website address is http://www.javierpanadero.com and his email address is jpanaderom@uoc.edu.

MADLENE LEIßAU studies management (M.Sc.) at the University of Applied Sciences Zwickau, Germany. She currently works as a research assistant in the Team Industry Analytics (www.industry-analytics.de) in the EU-project iDEV40. Her email address is Madlene.Leissau.Gel@fh-zwickau.de.

CHRISTIN SCHUMACHER is a researcher in the department of Computer Science at TU Dortmund University. She holds a M.Sc. in business mathematics. Her research interests include mathematical optimization, heuristics and discrete event simulation for machine scheduling and real world optimization problems. Her e-mail address is christin.schumacher@tu-dortmund.de. Her website is https://ls4-www.cs.tu-dortmund.de/cms/de/home/schumacher/.

ANGEL A. JUAN is a Full Professor of Operations Research & Industrial Engineering in the Computer Science Dept. at the Universitat Oberta de Catalunya (Barcelona, Spain). He is also the Director of the ICSO research group at the Internet Interdisciplinary Institute and Lecturer at the Euncet Business School. Dr. Juan holds a Ph.D. in Industrial Engineering and an M.Sc. in Mathematics. He completed a predoctoral internship at Harvard University and postdoctoral internships at Massachusetts Institute of Technology and Georgia Institute of Technology. His main research interests include applications of simheuristics and learnheuristics in computational logistics and transportation, as well as computational finance. He has published about 110 articles in JCR-indexed journals and more than 260 papers indexed in Scopus. His website address is http://ajuanp.wordpress.com and his email address is ajuanp@uoc.edu.