Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

ROLLING-HORIZON SIMULATION OPTIMIZATION FOR A MULTI-OBJECTIVE BIOMANUFACTURING SCHEDULING PROBLEM

Kim van den Houten
Mathijs de Weerdt
David M.J. Tax

Delft University of Technology
Van Mourik Broekmanweg 6
Delft, 2628XE, Netherlands

Esteban Freydell

DSM
Alexander Fleminglaan 1
Delft, 2613AX, Netherlands

Eva Christoupoulou
Alessandro Nati

Systems Navigator
Elektronicaweg 25
Delft, 2628XG, Netherlands

ABSTRACT

We study a highly complex scheduling problem that requires the generation and optimization of production schedules for a multi-product biomanufacturing system with continuous and batch processes. There are two main objectives here; makespan and lateness, which are combined into a cost function that is a weighted sum. An additional complexity comes from long horizons considered (up to a full year), yielding problem instances with more than 200 jobs, each consisting of multiple tasks that must be executed in the factory. We investigate whether a rolling-horizon principle is more efficient than a global strategy. We evaluate how cost function weights for makespan and lateness should be set in a rolling-horizon approach where deadlines are used for subproblem definition. We show that the rolling-horizon strategy outperforms a global search, evaluated on problem instances of a real biomanufacturing system, and we show that this result generalizes to problem instances of a synthetic factory.

1 INTRODUCTION

Efficient bioprocess industries can play a crucial role in feeding the world population in a sustainable way. Production sites for the process industries are often multipurpose, highly flexible systems. The number of different products using (partially) the same machines in their production process has been increased due to market pressure. Given long-horizon demand up to a full production year, factory operators want to optimally use their resources, while ensuring that deadlines for customer orders are met. Using these deadlines as hard constraints makes the scheduling too restrictive; instead a lateness objective can be defined which sums the lateness of all customer orders. Efficiency of a schedule can be evaluated by measuring the makespan. Intuitively, makespan minimization also contributes to reducing lateness of customer orders. On the other hand, not considering deadlines can potentially lead to schedules with lower makespan values. To illustrate this; ignoring deadlines allows clustering products in such a way that shared resources are used in a more efficient manner. For example, the same products for different customers can sometimes be produced in a single batch, or when subsequent batches are for the same or similar products, machine set-up and cleaning times can be shorter. The trade-off between the lateness and makespan objective makes the optimization therefore challenging.

Schedulers in these industries are facing highly complex scheduling problems, which are in general NP-hard (Pinedo 2016; Georgiadis et al. 2019a). Therefore, the development of efficient algorithms to optimize scheduling decisions in large-scale bioprocessing industries is challenging. Several studies are dedicated to exact optimization methods, such as linear programming, for this type of scheduling problem, which are summarized by Georgiadis et al. (2019a). The main conclusion is that for the optimization of large-scale, long-horizon scheduling problems, the only successes can be achieved with help of decomposition, heuristics, and simplifications because linear programming models become otherwise intractable. Especially in biomanufacturing, such as performed by DSM (a global company in Nutrition, Health and Bioscience), details of the operation constraints are extremely important for the generation of feasible production schedules. Oversimplification should preferably be avoided in the solution strategy. Ideally, we have algorithms that can produce optimal solutions for very detailed models of such scheduling problem, but for problems of this complexity (NP-hard) and size, this may never be possible (Garey and Johnson 1990).

Simulation-based methods are more suitable than linear programming approaches for the consideration of interdependent processes, constraints, and interactions in manufacturing systems. We observe that detailed Discrete Event Simulators (DES) are already used in practice for scenario analysis and manual scheduling. Currently, DSM schedulers aim for a good production sequence, which is then later augmented by a highly-detailed DES that contains heuristic rules for machine selection, and timing, to evaluate different production sequences. Mimicking detailed processes of a factory can be done with help of sophisticated simulation software, while including all these details in a linear programming model results in intractable models. These DES are often very specific to the manufacturing system they represent. Algorithmic techniques that support scheduling decisions are more generic. One prime example of this are simheuristics (Juan et al. 2015), which are defined as metaheuristics combined with a simulation tool for cost function evaluation. Only a few studies have been dedicated to the design of such simheuristics integrated with industry-proven DES for long-horizon scheduling of large-scale biomanufacturing systems.

In this research, we focus on the development of simheuristics for sequencing of customer orders. The employed DES translates a production sequence into a feasible schedule, and evaluates the corresponding objectives. We investigate which optimization strategies are efficient for scheduling with long production horizons in the (bio)process industries. Inspired by the manually constructed schedules, for which deadlines are used to construct a reasonable production sequence, we wonder how much better results could be obtained by a global simheuristic, or by a rolling-horizon simheuristic that uses the deadlines to reduce the search space. We aim to get insight in how this search space reduction affects the solution quality and run time. Given a weighted sum of the makespan and lateness as cost function for optimization, the research examines whether the same weights should guide the subproblem optimizations within the rolling-horizon simheuristic.

We observe that employing product deadlines to construct a rolling-horizon simheuristic is beneficial for improving costs and computation time of the algorithm, compared to a global strategy. Surprisingly, the rolling-horizon strategy performs even better (by a significant margin) than a global simheuristic that begins with a production sequence sorted by deadline, when both simheuristics are given an equal budget. Although we expected that it would be better to increase the makespan weight in the rolling-horizon simheuristic, we observe that the differences in obtained objectives are minor compared to using the true weights in the subproblem search.

In Section 2, we give an overview of related work. The formal problem description is described in Section 3, which is inspired on an industrial use case, provided by DSM. The employed DES are discussed in Section 4, which comprises the industry-proven model of the DSM factory (Rockwell Arena), and a model for a synthetic factory implemented in SimPy, which we make publicly available. The latter is developed to show that our algorithms work independently of the DES, and can be generalized to other factories. Additionally, we discuss the optimization strategies in Section 5, which includes the rolling-horizon. For the experiments and results, we refer to Section 6 and Section 7.

2 RELATED WORK

Scheduling decisions in the process industries include batching, sequencing, routing, machine assignment, and timing. Over the past decades, generic optimization-driven methods for these problems have been studied (Kondili et al. 1993; Pantelides 1993), and a comprehensive overview is provided by Georgiadis et al. (2019a). In the process industries, it is often the case that the system cannot be modelled with jobs and operations, like is usually done in discrete manufacturing (Pinedo 2016). This is caused by the fact that operations such as mixing and splitting of batches can exist, which means that a product batch does not necessarily keep its identity throughout the process. Other examples of complicating factors are processing times that depend on the batch size, and the presence of continuous, and semi-continuous processes. Due to the large scale, long horizon, and industry-specific operation constraints, the potential of exact optimization methods is limited (Chica et al. 2020). Some examples where either Mixed Integer Linear Programming (MILP), or Constraint Programming (CP) models are employed for short-horizon industrial problems, can be found in (Awad et al. 2022; Georgiadis et al. 2019b). It was pointed out that even for short-horizon problems, simplifications are needed for linear programming (Georgiadis et al. 2019a), and there is a chance that manufacturers will not accept the proposed schedules because of infeasibility caused by an insufficient level of detail (Klanke and Engell 2022).

Simulation optimization (SO) is considered as a suitable alternative for exact optimization, since industry-strength simulation models could be integrated with optimization algorithms that interface with the simulation model as an external component. Simulation is powerful for the evaluation of stochastic components, but can also be used to model complex systems with interdependent processes and constraints. More specifically, the value of simheuristics for realistic combinatorial optimization problems has been widely recognized (Juan et al. 2015; Chica et al. 2020; Juan et al. 2022). A recent tutorial on how to connect Python with DES software for the development of simheuristics was given by Peyman and Dehghanimohammadabadi (2021). A simulation tool can be used to translate input scheduling decisions into an actual schedule (gannt chart), as well as to estimate objective values. Simheuristic applications in scheduling comprise various job shop problems combined with Monte Carlo Simulation (Juan et al. 2014; Gonzalez-Neira et al. 2017; Hatami et al. 2018). Research in the field of simheuristics for scheduling in process industries is limited, especially for multi-objective, long horizon problems. Piana and Engell (2010) developed a simheuristic method for a chemical engineering plant, where the makespan was considered as objective. Klanke et al. (2021) proposed an evolutionary algorithm (EA) combined with a DES tool to optimize an industrial formulation plant. They evaluated their method on problem instances for the medium-term.

For the design of simheuristics, it is advised that the choice for a simheuristic framework should fit the complexity of the simulation tool, meaning that complex simulation tools work often better with simple simheuristics, and the other way around (Chica et al. 2020). Iterated Greedy (IG) algorithms have proven to be successful for sequencing scheduling problems, such as the permutation flowshop scheduling problem (PFSP) (Ruiz and Stützle 2007), and the hybrid version (HFSP) (Öztop et al. 2018).

When specifically focusing on problems with long horizons, rolling-horizon algorithms could be of help. With this principle a sequence of subproblems is solved, for which the size can be controlled. Such strategies have been applied in several studies (Ovacik and Uzsoy 1994; Glomb et al. 2022). A rolling-horizon principle applied to a batch multi-product production system is provided by Wu et al. (2021), although the studied system is significantly smaller than the factory of interest. Earlier research has shown that such principles combined with exact methods can even yield near-optimal solutions (Glomb et al. 2022).

We recognize a clear gap in the literature regarding scheduling of bioprocess factories. On the one hand, we observe that linear programming approaches have shown to be successful on small, short horizon, and/or simplistic problems. On the other hand, simulation-based methods for large-scale industrial problems focusing on long horizons are rare. Rolling-horizon principles have been applied to batch manufacturing, but as far as we know it is not understood whether a rolling-horizon method would lead to better results in the same time as a global search in such a large-scale, complex system as studied in this research. In particular, we would like to understand whether the objectives in such a multi-objective problem should be weighted differently in a rolling-horizon setting compared to the long-horizon objective.

3 PROBLEM DESCRIPTION

The scheduling problem can be described using the standardized framework, provided by Georgiadis et al. (2019a), complemented with some additional specifications. We observe a multi-purpose, multi-product network facility, which is a system where different bio-based products are produced according to unique recipes, and the routings through the plant are product-specific, and flexible. Demand for a full year is given, which yields a scale of more than 200 jobs per year each consisting of multiple unit operations that are executed either subsequently or (partially) in parallel. An additional complexity is the mix of batch and semi-continuous processes.

Currently, DSM schedulers focus on the sequencing of the different products given the production plan for the full year. Decisions regarding batch sizing are made in advance. Soft deadlines for the different batches are given per month. A schedule includes assigning resources to tasks, and sequencing and timing the different tasks, and a feasible schedule satisfies all pre-defined operation constraints. The scheduling objectives are the total time of the schedule (makespan), and the total lateness of the customer orders which are combined in a cost function. Factory managers define the cost function as a sum of the two objectives weighted by weight $\lambda \in [0,1]$ for the makespan objective and $1-\lambda$ for the lateness objective. For the factory of interest, the decision-makers set $\lambda^* = 0.5$. Given that f_j is the finish time of job j, and d_j is the deadline of job j, the cost function is then defined as:

$$C_{\lambda^*} = \lambda^* \max_{j \in jobs} f_j + (1-\lambda^*) \sum_{j \in jobs} max(0, f_j - d_j) = 0.5 \max_{j \in jobs} f_j + 0.5 \sum_{j \in jobs} max(0, f_j - d_j) \tag{1}$$

The facility that we study consists of thirteen resource groups, with one up to eleven machines per group, from which capacities can differ within one group (for an overview, see Table 1). Processing times are often given in rates (dependent on batch sizes), and are typically long ($>$ 100 hours); this makes the use of time-indexed models problematic. The availability of raw material, and man hours are left out of scope. On this manufacturing site, a variety of 55 different end products can be produced. Additional complexities of the system are:

- compatibility constraints,
- sequence-dependent cleaning times and pre- and post-processes,
- scheduled maintenance,
- changing batch weights/sizes (e.g. due to filtering steps),
- cooling restrictions.

Given this large-scale, NP-hard scheduling problem with a comprehensive set of detailed operation constraints, we wonder which optimization strategy is suitable for the generation, and improvement of production schedules. We emphasize that oversimplification is undesirable, because it can lead to schedules that will be rejected by factory managers. We are particularly interested in how to handle long horizon problem instances up to a full production year because 1) this is highly relevant for our industrial partner, and 2) there is a clear gap in the literature, in which the majority studied short to medium term horizons.

Table 1: Resource groups.

Resource group	Number of machines
Fermenter 1 - 5	5
Harvesting tanks A	4
Harvesting tanks B	8
Filters A	3
Filters B	1
Buffer tanks	6
Filters C	5
Filters D	6
Stabilization tanks	11

4 DISCRETE-EVENT SIMULATION

Given our interest in scheduling a highly detailed process, simulation optimization (SO) emerges as a suitable approach for modelling all operational constraints and interdependent processes (Juan et al. 2015; Klanke and Engell 2022). As discussed earlier, finding optimal solutions in time for this problem is not to be expected because of its size and complexity. To address this challenge, we employ two deterministic Discrete Event Simulators (DES) that translate a production sequence, representing the ordered products based on demand, into a feasible schedule. The simulation model mimics the flow of batches through the factory while respecting all scheduling rules given by the product specific recipes. The simulation incorporates heuristic rules to determine machine assignments and precise timing of operations. Therefore, it should be noted that some sub-optimality cannot be avoided. Nonetheless, this approach facilitates the use of simheuristics that solely optimize the production sequence without having to consider all the complex interactions modeled in the DES.

Currently, DSM schedulers use a DES that is implemented in Rockwell Arena software (see Drevna and Kasales (1994)). This tool is validated, and is now integrated in a simheuristic framework. To test whether our proposed methods can be generalized, we create a synthetic factory, and developed a DES with help of the open-source Python package SimPy (see Matloff (2008)). Both models take as main input a production sequence, and output a feasible schedule, which can be visualized as Gannt chart, such as is shown in Figure 1, and Figure 2.

1. DSM Factory (Arena): The Arena model is a highly-detailed DES, which is developed by Systems Navigator consultants. The communication between Arena, and Python is realised with text files, and Arena replications. The model reflects the DSM factory including all product flows, mandatory cleaning times, scheduled maintenance, cooling constraints, and measures the required objectives. Given a production sequence, it keeps tracks of resource utilization at discrete time steps, including batch weights, and information about intermediate and semi-finished products. The simulation tool also deterministically translates the input sequence into the objectives.

2. Synthetic Factory (SimPy): We developed a synthetic factory with help of the open-source Python library SimPy (Matloff 2008). For the design of the SimPy simulator, we copied the combinatorial size of the real factory, i.e. we use a similar number of resource groups, resources, and unique products. Some of the characteristics of the real factory are left out, such as the exact flow of liquid volumes, maintenance, and change-over times. The simulation tool is accessible via https://github.com/kimvandenhouten/SimPyManufacturing. This tool is currently used in a deterministic mode.

5 OPTIMIZATION STRATEGY

This section presents two simheuristic strategies for sequence optimization: a global search approach and a rolling-horizon approach. We investigate which strategy is more effective for the considered long-horizon

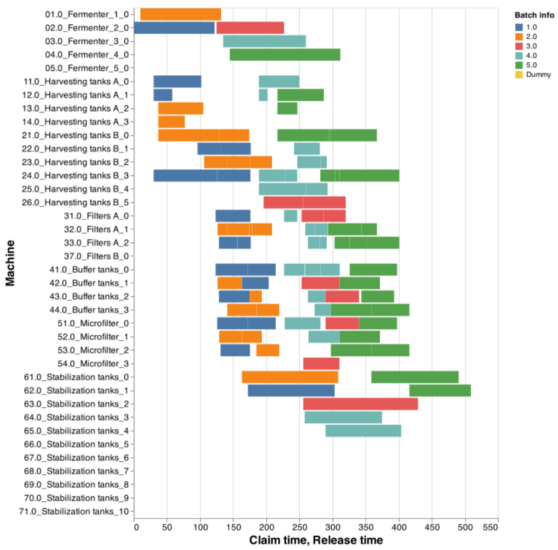

Figure 1: Example Gannt DSM Factory

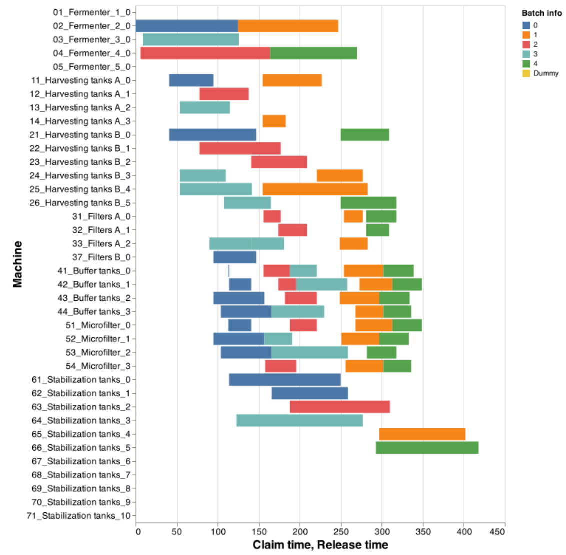

Figure 2: Example Gannt Synthetic Factory

problem instances. The search space is defined as the permutation space S_n, which represents all possible permutations of an unordered set of values $[n]$. The *budget* refers to the number of cost function evaluations or different sequences evaluated using the DES in a single algorithm run. To compare the different optimization strategies, we fix the number of cost function evaluations.

5.1 Global Search

The **global search** strategy utilizes the entire search space S_n and aims to minimize C_{λ^*} (see Equation (1)). We employ relatively simple search procedures, following key advice from the literature for simheuristics integrated with complex DES (Chica et al. 2020). We therefore use a local search with random swaps. The initialisation is either random, or sorted by deadline, and we hypothesize that the latter is helpful for the lateness objective. We furthermore test an iterated greedy algorithm, that has successfully been applied to sequencing scheduling problems (Ruiz and Stützle 2007). A fixed evaluation budget is given as hyperparameter, and tuned to the instance size. Due to the randomness in the search algorithms, we perform multiple restarts and report the minimum costs obtained. Furthermore, we compare all strategies against a random search baseline.

5.2 Rolling-Horizon

The **rolling-horizon** strategy aims for an efficient reduction of the search space S_n, and is summarized in Algorithm 1. Rolling-horizon algorithms have been successfully employed for large-scale problems (Ovacik and Uzsoy 1994). Our proposed strategy utilizes product deadlines to divide the global problem into subproblems. We aim to understand how the consequent reduction of the search space affects solution quality. We aim to investigate the performance difference between the rolling-horizon and global search strategies. The first step involves sorting the production sequence by deadline. Subsequently, parts of the sequence are iteratively optimized using a predefined search method. The hyperparameter k determines the size of each subproblem, and the hyperparameter m fixes the first m items of the just-optimized subsequence. Different (k,m) combinations control how much the search space is reduced. During subproblem evaluation, all previously solved and fixed items of the production sequence are included. This means that the length of the sequence evaluated with the DES increases throughout the algorithm. The optimization task remains to minimize C_{λ^*}, where the decision makers define $\lambda^* = 0.5$. However, the search algorithms within the rolling-horizon strategy use C_λ with $\lambda \in [0,1]$ as a hyperparameter. This investigation aims to determine whether the rolling-horizon strategy can achieve better sequences (i.e., lower C_{λ^*} values) while tuning λ. We again perform multiple random restarts and report the minimum costs obtained as well as the total runtime for the different restarts.

Algorithm 1 Rolling-horizon strategy

requires *search_algorithm*: method for sequence optimization, *budget*: total number of cost function evaluations, C_{λ^*}: true cost function, C_λ: cost function used for subproblems, *demand_list*: list of products including deadlines, n: length of demand list, k: number of products considered per search, m: number of products fixed after each search

initialize:

production_plan ← sort_by_deadline(demand_list)

nr_searches ← round(n/m)

budget_per_search ← round(budget / nr_searches)

fixed ← []

for i in range(0, nr_searches): **do**

    x ← production_plan[$i*m : i*m+k$]

    $x^{optimized}$ ← search_algorithm(budget_per_search, C_λ(combine_sequences(fixed, x)))

    production_plan[$i*m : i*m+k$] ← $x^{optimized}$

    fixed ← production_plan[$0 : (i+1)*m$]

end for

return production_plan, C_{λ^*}(production_plan)

6 EXPERIMENTS

This section provides details on our experimental setup, including the hyperparameter configurations for the algorithms. We begin by explaining the tuning process, which involves adjusting the budget and random restarts for the algorithms. Specifically, for the rolling-horizon strategy, we focus on tuning the weight parameter λ. Given that the rolling-horizon simheuristic implicitly considers the lateness objective, we hypothesize that it could be beneficial to increase the makespan weight λ during the rolling-horizon. We furthermore explain how we evaluate the different search algorithms, where we first test the methods on short horizon problems to identify the most promising search strategies. Subsequently, we conduct experiments that compare the effectiveness of global and rolling-horizon strategies for long horizon instances and optimize tuning for the rolling-horizon approach.

6.1 Evaluation

We evaluate our algorithms on the two different factories, which are introduced in Section 4. All experiments involving the Arena model are done on a Dell Latitude E7450 with Intel Core i7 5600U. The SimPy experiments are done on a virtual server that uses an Intel(R) Xeon(R) Gold 6148 CPU with two 2.39 GHz processors, and 16.0 GB RAM. We generated test instances for both problems that are representative for the problem faced by our industrial partner. In the DSM factory, on average 20 products per month are produced. We used historical data to make problem instances (one particular combination of parameters for the scheduling problem) of different sizes that have product mixes that are representative for the real factory.

First, we evaluate different simheuristics on small instances, with a horizon of one, or two months. We test the different algorithms for different initialisation, one that is initialized randomly, and one that starts from a heuristic rule: sorted by deadline. We analyze which search strategy minimizes the cost function for the short horizon. The outcome is used to determine which search strategies to use within the rolling-horizon algorithm. Then, we test the simheuristics for the long horizon problem instances with horizons of six months or a full year. Finally, we include an evaluation on one particular problem instance, for which we know the sequence that was selected by experts in the plant.

6.2 Hyperparameter Settings

The settings for budget, and random restarts are tuned on a subset of problem instances. For the budget, we use $budget = (size/20) \cdot 200$, and we decide to use a multi-start with 3 different random restarts for all algorithms, yielding per problem instance a total budget of $total\ budget = 3 * (size/20) \cdot 200$. For the rolling-horizon (k,m) and λ are tuned. The tuning for (k,m) resulted in the setting $(k,m) = (40,10)$. While tuning setting λ, we observed that in the calibration for different problem instances, different λ settings performed best. Since it was not always the case that the best performing setting was equal to λ^*, we decide to include two different rolling-horizon settings in our final experimentation. We both use $\lambda = \lambda^* = 0.5$ (the one that is similar to the weights in the costs function), and $\lambda = 0.9$ (which performed promising according to the tuning).

7 RESULTS

To read the results in Table 2, and Table 3, we introduce some abbreviations. We use *A*, and *S* to refer to the Arena, and the SymPy models. We refer to local search with *LS*, random search with *RS*, iterated greedy with *IG*, and rolling-horizon with *RH*. Furthermore, *i=r* refers to a random initialisation, and *i=s* is an initialisation sorted by deadline. The different instances are encoded with *size_id*. The presented costs are the best value obtained with the different restarts, and the runtime is the total runtime that was used to finish the algorithm.

7.1 Short Horizon Problem Instances

Our objective is to determine the most effective search strategy in terms of computation time and costs for the given problem instances. To accomplish this, we compare the performance of three search strategies: random search, local search, and iterated greedy approach. We specifically focus on short horizon problem instances spanning one to two production months, as presented in Table 2. These experimental results determine the search strategy that will be used in the rolling-horizon simheuristic.

We observe that for the short horizon problem instances, the local search strategy with random initialization outperforms both the random search and the iterated greedy algorithm when applied to instances of size 20. However, for instances of size 40, the local search strategy with sorted initialization outperforms all other methods. The differences in runtime are negligible.

7.2 Long Horizon Problem Instances

Based on the outcomes from the short horizon problem instances, summarized in Table 3, we observe that local search is the most effective approach. However, whether a random or sorted initialization is preferable remains unclear. We focus on comparing the effectiveness of employing a global simheuristic versus a rolling-horizon simheuristic for the long horizon problem instances. Additionally, we explore the potential benefits of adjusting the weights in the cost function used in the rolling-horizon approach.

We observe a clear pattern in the results. In the majority of instances, local search with sorted initialization outperforms local search with random initialization. However, applying a rolling-horizon strategy is better than the global search method for all test instances. The tuned setting $\lambda = 0.9$ only resulted in better performance for some of the instances. Overall, our results demonstrate that the rolling-horizon strategy is significantly faster than the global search strategy when evaluated on all test instances. This can be explained by the fact that the DES evaluates shorter sequences at the beginning of the algorithm, leading to improved efficiency.

Table 2: Results short horizon problem instances.

		Costs					Runtime (s)				
DES	**I**	**IG**$_{i=r}$	**IG**$_{i=s}$	**LS**$_{i=r}$	**LS**$_{i=s}$	**RS**	**IG**$_{i=r}$	**IG**$_{i=s}$	**LS**$_{i=r}$	**LS**$_{i=s}$	**RS**
A	20_1	1035	966	**614**	643	875	209	**206**	220	219	215
A	20_2	1605	1197	**1141**	**1141**	1513	199	204	202	**197**	220
A	20_3	1084	1360	891	**820**	1111	192	**190**	196	196	203
A	20_4	763	644	**578**	593	766	**185**	185	192	193	193
A	20_5	1228	1043	**945**	977	1210	**194**	**194**	201	201	201
A	20_6	1132	1128	**871**	904	1260	197	**196**	203	202	202
A	20_7	1180	1202	**878**	935	1138	**195**	**195**	203	204	203
A	20_8	1029	1316	972	**884**	1206	**188**	189	194	194	195
A	20_9	1274	1472	**830**	977	1357	**194**	195	202	203	201
A	40_1	6314	3410	3186	**2721**	6830	507	482	**459**	**459**	460
A	40_2	8014	3481	3012	**2876**	8083	481	480	**473**	476	**473**
A	40_3	7387	4816	3994	**3891**	8168	467	514	469	467	**466**
A	40_4	8834	6003	5056	**4813**	8903	509	490	**465**	469	467
A	40_5	6929	3474	3253	**2589**	7145	509	514	476	474	**473**
A	40_6	6655	3604	2479	**2447**	5852	490	484	467	473	**466**
A	40_7	8624	7400	5082	**4767**	8612	514	521	**475**	**475**	477
A	40_8	6814	5663	2897	**2811**	7004	497	501	**462**	481	464
A	40_9	6407	4806	**2151**	2608	5860	502	510	463	464	**462**
S	20_1	877	912	**817**	820	829	52	51	**39**	**39**	**39**
S	20_2	547	**467**	**467**	**467**	468	54	51	38	38	**37**
S	20_3	1491	1403	**1303**	1306	1321	51	53	**39**	41	**39**
S	20_4	735	724	**702**	**702**	708	51	54	40	**39**	**39**
S	20_5	972	**857**	**857**	**857**	863	36	40	29	**30**	**30**
S	20_6	624	628	589	587	593	54	54	**39**	**39**	**39**
S	20_7	976	**882**	**882**	**882**	890	54	59	**42**	44	**42**
S	20_8	689	**659**	**659**	**659**	660	48	47	36	**35**	**35**
S	20_9	1132	1130	**1086**	**1086**	1092	61	61	45	47	**45**
S	40_1	3991	2413	**2300**	2323	3068	235	237	181	**180**	**180**
S	40_2	3700	2210	1920	**1901**	2869	266	259	**192**	193	196
S	40_3	2851	1063	992	**984**	1689	254	256	**188**	191	189
S	40_4	2918	1759	1563	**1557**	2178	257	249	**188**	193	196
S	40_5	5519	3731	3420	**3405**	4184	240	231	178	**176**	181
S	40_6	3144	1547	**1402**	1404	2162	247	244	**183**	184	185
S	40_7	3781	2412	**2259**	2277	3108	250	245	191	187	**185**
S	40_8	4492	2386	2323	**2314**	3347	230	231	**171**	175	175
S	40_9	3090	2126	1860	**1848**	2824	250	256	190	**186**	189

7.3 Evaluation on Real Production Plan

As a final evaluation, we test how much we can improve a real schedule that was created by experts for the horizon July - December 2022, consisting of 122 products. We obtain the best solution with the rolling-horizon strategy. The total cost reduction is 45%. Looking at the two objectives seperately, this schedule improved the makespan with 4%, and total lateness with 58%.

Table 3: Results long horizon instances.

		Costs					Runtime (m)				
DES	**I**	**LS**	**LS**	**RS**	**RH**	**RH**	**LS**	**LS**	**RS**	**RH**	**RH**
		$i=r$	$i=s$		$\lambda_1=0.5$	$\lambda_1=0.9$	$i=r$	$i=s$		$\lambda_1=0.5$	$\lambda_1=0.9$
A	120_1	26157	20540	77153	**11903**	14069	43	43	42	38	**37**
A	120_2	30402	23217	74626	**14958**	15740	41	44	41	38	**36**
A	120_3	32640	23125	82278	**15765**	17604	40	40	40	36	**35**
A	120_4	25123	16379	77207	**11733**	11484	40	39	40	32	**30**
A	120_5	32416	23968	84219	**20328**	18215	40	41	40	32	**31**
A	120_6	30917	23509	84376	**17455**	18585	40	40	41	**32**	**32**
A	120_7	26769	21854	82850	14244	**11947**	40	40	40	**32**	33
A	120_8	17555	13239	62741	**5496**	6967	39	39	39	**31**	**31**
A	120_9	20671	12481	69812	**5932**	7650	40	39	39	32	**31**
A	240_1	158342	97156	380652	**43090**	49758	120	118	117	87	**86**
A	240_2	123525	72521	348745	**34820**	38030	110	120	126	**84**	**84**
A	240_3	120497	89206	343697	53132	**38557**	111	113	111	86	**83**
A	240_4	121902	77800	333146	**43307**	46398	121	118	119	**84**	85
A	240_5	113269	78149	350476	48115	**42695**	122	122	119	**84**	**84**
A	240_6	110896	71205	361004	**26128**	26901	124	137	129	84	**81**
A	240_7	124405	85067	369330	61705	**57794**	135	145	139	**81**	**81**
A	240_8	118431	86317	352642	**52791**	57862	151	126	159	**81**	**81**
A	240_9	111132	86317	345495	**43259**	44054	119	108	112	80	84
S	120_1	5272	3982	23986	**4029**	4074	57	56	57	**39**	40
S	120_2	4705	3640	24463	3675	**3670**	57	57	57	41	**40**
S	120_3	2590	2336	19617	2318	**2305**	48	49	50	36	**35**
S	120_4	8536	8105	25313	8208	**8135**	54	54	55	41	**39**
S	120_5	3900	3282	21076	3316	**3264**	44	45	45	34	**33**
S	120_6	4521	3308	24599	**3318**	3345	51	51	52	**37**	**37**
S	120_7	7113	5398	27043	**5343**	5395	52	51	52	**38**	**38**
S	120_8	3521	3004	21048	2984	**2979**	47	47	48	**34**	**34**
S	120_9	5799	4691	22149	4783	**4678**	54	53	54	**39**	**39**
S	240_1	31855	23246	124680	23019	**22873**	308	309	315	**173**	**173**
S	240_2	23204	16701	115122	**16127**	16337	298	294	300	**169**	176
S	240_3	11236	5596	99495	**5552**	5565	307	312	296	**170**	172
S	240_4	8373	5488	86714	**5289**	5300	274	294	299	**158**	**158**
S	240_5	10386	5412	96277	**5375**	5405	326	327	328	172	**171**
S	240_6	8987	6114	92082	**6093**	6098	313	315	317	**163**	166
S	240_7	16620	11549	105234	11346	**11334**	329	327	330	178	**174**
S	240_8	12416	4728	99579	4692	**4691**	337	343	341	**184**	**184**
S	240_9	18326	12600	109805	**12424**	12757	331	328	332	**174**	177

8 CONCLUSION AND DISCUSSION

In this work, we studied a real-world biomanufacturing scheduling problem, which is very computationally challenging due to high flexibility of the system, two objectives, and a long horizon considered. We investigated whether a rolling-horizon strategy is better for tackling such long horizon problem instances, compared with a global search approach. Additionally, we analyze whether adjusting objective weights in a rolling-horizon simheuristic yields lower-cost solutions. Our objective was to develop a solution strategy that avoids oversimplification of the system, a common requirement for exact methods that may result in infeasible solutions violating essential constraints, which are subsequently rejected by plant managers.

A simheuristic framework integrated with an industry-proven DES turned out to be effective for the generation of feasible schedules in a reasonable amount of time. We showed that a rolling-horizon principle results in better solutions than a global optimization strategy given a limited budget, and reduces the computation time significantly. Furthermore, we showed the generalizability of our methods with help of the synthetic factory. We discovered that adjusting the weights of the cost function within the rolling-horizon strategy, particularly by increasing the makespan weight, improved performance for certain problem instances.

As a valuable contribution, we achieved cost reduction in an actual schedule implemented at the DSM factory during the period of July-December 2022. For future research, we aim to better understand for which cases adjusting cost function weights is beneficial within the context of the rolling-horizon strategy. Future work will involve improving the simheuristics by using more informed metaheuristics, expanding simulations to incorporate uncertainty, and including additional scheduling decisions beyond sequencing to reduce the optimality gap. Despite these potential future improvements, this study lays the foundation for the development of intelligent simulation-based algorithms applicable to highly complex manufacturing systems in process industries.

To help further research in this domain, we have made the SimPy DES of the synthetic factory publicly available. We hope this will stimulate the research community to explore this area and contribute insights that can lead to the creation of a diverse set of benchmark factories for evaluation and algorithm development.

ACKNOWLEDGMENTS

This project is funded by DSM and is part of the AI4b.io lab (www.AI4b.io). The brainstorm sessions with the Systems Navigator team contributed important insights for the practical value of this work. We want to thank our anonymous reviewers for their feedback, which has greatly improved the quality of this paper.

REFERENCES

Awad, M., K. Mulrennan, J. Donovan, R. Macpherson, and D. Tormey. 2022. "A Constraint Programming Model for Makespan Minimisation in Batch Manufacturing Pharmaceutical Facilities". *Computers and Chemical Engineering* 156:107565.

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

Drevna, M., and C. Kasales. 1994. "Introduction to Arena". In *Proceedings of the 1994 Winter Simulation Conference*, edited by J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, 431–436. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Garey, M. R., and D. S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. 1st ed. USA: W. H. Freeman Co.

Georgiadis, G. P., A. P. Elekidis, and M. C. Georgiadis. 2019a. "Optimization-Based Scheduling for the Process Industries: From Theory to Real-Life Industrial Applications". *Processes* 7:438.

Georgiadis, G. P., C. Ziogou, G. Kopanos, B. M. Pampín, D. Cabo, M. Lopez, and M. C. Georgiadis. 2019b. "On the Optimization of Production Scheduling in Industrial Food Processing Facilities". *Computer Aided Chemical Engineering* 46:1297–1302.

Glomb, L., F. Liers, and F. Rösel. 2022. "A Rolling-Horizon Approach for Multi-Period Optimization". *European Journal of Operational Research* 300:189–206.

Gonzalez-Neira, E. M., D. Ferone, S. Hatami, and A. A. Juan. 2017. "A Biased-Randomized Simheuristic for the Distributed Assembly Permutation Flowshop Problem with Stochastic Processing Times". *Simulation Modelling Practice and Theory* 79:23–36.

Hatami, S., L. Calvet, V. Fernández-Viagas, J. M. Framiñán, and A. A. Juan. 2018, 8. "A Simheuristic Algorithm to Set Up Starting Times in the Stochastic Parallel Flowshop Problem". *Simulation Modelling Practice and Theory* 86:55–71.

Juan, A. A., B. B. Barrios, E. Vallada, D. Riera, and J. Jorba. 2014. "A Simheuristic Algorithm for Solving the Permutation Flow Shop Problem With Stochastic Processing Times". *Simulation Modelling Practice and Theory* 46:101–117.

Juan, A. A., J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira. 2015. "A Review of Simheuristics: Extending Metaheuristics to Deal With Stochastic Combinatorial Optimization Problems". *Operations Research Perspectives* 2:62–72.

Juan, A. A., Y. Li, M. Ammouriova, J. Panadero, and J. Faulin. 2022. "Simheuristics: An Introductory Tutorial". In *Proceedings of the 2022 Winter Simulation Conference*, edited by B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C. Corlu, E. C. L.H. Lee, T. Roeder, and P. Lendermann, 1325–1339. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Klanke, C., D. Bleidorn, C. Koslowski, C. Sonntag, and S. Engell. 2021. "Simulation-Based Scheduling of a Large-Scale Industrial Formulation Plant Using a Heuristics-Assisted Genetic Algorithm". In *Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion*, edited by F. Chicano, 1587–1595. New York, United States: Association for Computing Machinery.

Klanke, C., and S. Engell. 2022. "Scheduling and Batching With Evolutionary Algorithms in Simulation–Optimization of an Industrial Formulation Plant". *Computers and Industrial Engineering* 174:108760.

Kondili, E., C. Pantelides, and R. Sargent. 1993. "A General Algorithm For Short-Term Scheduling Of Batch Operations". *Computers Chemical Engineering* 17:212–227.

Matloff, N. 2008. *Introduction to Discrete-Event Simulation and the SimPy Language*. https://web.cs.ucdavis.edu/~matloff/matloff/public_html/156/PLN/DESimIntro.pdf.

Ovacik, I. M., and R. Uzsoy. 1994. "Rolling Horizon Algorithms for a Single-Machine Dynamic Scheduling Problem With Sequence-Dependent Setup Times". *International Journal of Production Research* 32:1243–1263.

Öztop, H., M. F. Tasgetiren, D. T. Eliiyi, and Q.-K. Pan. 2018. "Iterated Greedy Algorithms for the Hybrid Flowshop Scheduling with Total Flow Time Minimization". In *Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, 379–385. New York, United States: Association for Computing Machinery.

Pantelides, C. 1993. "Unified Frameworks for Optimal Process Planning and Scheduling". In *Proceedings of the Second International Conference on Foundations of Computer-Aided Process Operations*, 18-23 July, Crested Butte, United States, 253–273.

Peyman, M., and M. Dehghanimohammadabadi. 2021. "A Tutorial on How to Connect Python With Different Simulation Software to Develop Rich Simheuristics". In *Proceedings of the 2021 Winter Simulation Conference*, edited by S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, 1–12. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Piana, S., and S. Engell. 2010. "Hybrid Evolutionary Optimization of the Operation of Pipeless Plants". *Journal of Heuristics* 16:311–336.

Pinedo, M. 2016. *Scheduling: Theory, Algorithms, and Systems*. 6th ed. Basel, Switzerland: Springer Nature.

Ruiz, R., and T. Stützle. 2007. "A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem". *European Journal of Operational Research* 177:2033–2049.

Wu, O., G. D. Ave, I. Harjunkoski, and L. Imsland. 2021. "A Rolling Horizon Approach for Scheduling of Multiproduct Batch Production and Maintenance Using Generalized Disjunctive Programming Models". *Computers and Chemical Engineering* 148:107268.

AUTHOR BIOGRAPHIES

KIM VAN DEN HOUTEN is an PhD candidate at Delft University of Technology. She holds a Masters Degree in Operations Research. Her research interests are on the intersection of optimization, machine learning, and simulation. Her email address is K.C.vandenhouten@tudelft.nl.

DR. DAVID M. J. TAX is an associate professor in pattern recognition at Delft University of Technology. His main research interest is on detection algorithms and (one-class) classifiers for rare classes in structured data, like multivariate timeseries data. His email address is D.M.J.Tax@tudelft.nl.

PROF. DR. MATHIJS DE WEERDT is a full professor in planning, and scheduling algorithms at Delft University of Technology. His email address is M.M.deWeerdt@tudelft.nl.

DR. ESTEBAN FREYDELL is a Senior Scientist Bioprocess Development at DSM. He is one of the principal investigators of AI4b.io representing DSM. His email address is Esteban.Freydell@dsm.com.

EVA CHRISTOPOULOU is simulation consultant at Systems Navigator. She is one of the developers of the Arena simulation model. Her email address is eva.christopoulou@systemsnavigator.com.

ALESSANDRO NATI is a simulation consultant at Systems Navigator. He was actively involved in the development of the Arena simulation model. His email address is alessandro.nati@systemsnavigator.com.