Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

A Two-stage Simulated Annealing-based Scheduling Algorithm for a Make-and-Pack Production Plant ⋆

Vassilios Yfantis * **Sven Büscher** *,** **Christian Klanke** *
Francesc Corominas *** **Sebastian Engell** *

* *Process Dynamics and Operations Group, Department of Biochemical and Chemical Engineering, TU Dortmund University, Emil-Figge-Str. 70, 44227 Dortmund, Germany (e-mail: vassilios.yfantis@tu-dortmund.de).*
** *DEAMOS e.K., Marderweg 17, 48683 Ahaus, Germany*
*** *Procter & Gamble, Temselaan 100, 1853 Strombeek-Bever, Belgium*

Abstract: Industrial scheduling problems are characterized by their highly combinatorial nature due to the large number of alternative solutions. The presence of discrete and continuous decisions in scheduling models usually lead to discrete optimization problems. Although powerful solvers and algorithms exist for this class of problems, real-life applications often still require prohibitively long computation times, severely hindering their practical deployability. An alternative way to solve hard discrete optimization problems is the use of metaheuristic algorithms, which provide approximate solutions in short computation times. In this contribution, a scheduling algorithm based on simulated annealing is developed and applied to a make-and-pack process from the consumer goods industry. The algorithm consists of two stages, the first of which determines the allocation of the orders to the production lines, while the second refines the schedule by optimizing the sequence within each line. Two different layouts of the plant are examined, one where the make-and-pack stages are directly coupled and another where a finite intermediate buffer is used to decouple the two production stages. The generated production schedules are compared to nominal ones provided by the planners at the plant, underlining the potential of the proposed approach, both in terms of solution quality and necessary computation time.

Keywords: Scheduling algorithms, Optimization problems, Metaheuristics, Manufacturing processes, Flexible manufacturing systems

1. INTRODUCTION

The process industry has been increasing its efforts towards making its operations more efficient. In the case of the consumer goods industry this can be achieved by minimizing changeover and idle times through efficient production scheduling. The main challenge is the large amount of different products processed in the same plant which is supplying different markets, giving rise to large-scale optimization problems. The changeover from one product to another usually leads to downtimes due to adjustments that have to be made to various pieces of equipment. Minimizing these changeover times is a highly combinatorial problem and therefore difficult to perform manually due to the exponentially increasing number of possible production sequences with increased number of orders to be processed (Baumann and Trautmann, 2014). Furthermore, a large number of constraints like machine eligibility, inventory levels and demand satisfaction have to be respected, rendering the scheduling problems even more challenging. All of these issues can be addressed by modeling the problems as mixed-integer programs and solving these by rigorous optimization algorithms (Kopanos et al., 2011; Méndez and Cerdá, 2002). These approaches lead to solutions with guaranteed constraint satisfaction and an exact evaluation of their solution quality in terms of optimality. However, they may require significant computation times and become computationally intractable for industrial-sized problems (Harjunkoski et al., 2014). If the fast generation of good, but not necessarily optimal schedules is the main focus, as e.g. in the case of online scheduling, metaheuristic techniques can be used as an alternative (Wang et al., 2000). Metaheuristics are guided stochastic search techniques which try to find good solutions to hard optimization problems. The design of metaheuristics involves a trade-off between covering large regions of the search space (exploration) and focusing on regions near promising solutions (exploitation). Even though these methods provide no guarantee nor even a measure of closeness to optimality, they have been shown

⋆ The work leading to this publication has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723575 (CoPro, spire2030.eu/copro) in the framework of the SPIRE PPP.

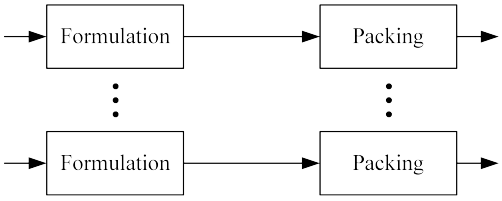

Fig. 1. Coupled Layout of the Make-and-Pack Plant.

to be suitable for the fast generation of good schedules (Sand et al., 2008). In this contribution a two-stage simulated annealing-based algorithm is presented and applied to a complex make-and-pack production plant that consists of several parallel lines. The first stage of the proposed algorithm takes the allocation decisions, i.e. decides on which production line each order will be processed. The second stage then serves as a refinement step in which the sequence within a production line is optimized.

2. MAKE-AND-PACK PROCESS

In this paper two different layouts of a consumer goods production plant will be examined. In the first one the two production stages of each line are coupled and in the second one they are decoupled by the use of a finite intermediate buffer. Both layouts are briefly presented in the following.

2.1 Coupled Layout

The coupled layout of the plant is shown in Fig. 1. It consists of several parallel production lines, each with a formulation and a packing unit directly connected in series. The production environment can therefore be described as a single stage process with multiple machines in parallel. All formulation units of the plant are identical and capable of processing each production order. However, each packing unit can only process a subset of orders, depending on the types of packing. Therefore each line can only process the subset of orders corresponding to the capability of its packing unit. Furthermore, the order dependent processing rates of the two units in each line have to be synchronized as the process is continuous without the possibility for intermediate storage. This results in a situation in which the unit with the lower processing rate becomes the bottleneck. Another drawback of this layout is that an entire line remains idle if either the formulation or the packing section requires a changeover. This may result in a significant loss of productivity. The issues of this layout show that efficient scheduling is necessary to ensure a profitable operation of the plant. A scheduling approach for this layout of the plant based on an extension of the precedence based mixed-integer programming (MILP) model from Kopanos et al. (2010) was proposed by Elekidis et al. (2019). It was shown that small problem instances could be solved directly in reasonable computation times while larger instances required a decomposition strategy to achieve satisfactory results. Nevertheless, improvements over manually generated schedules were clearly demonstrated.

2.2 Decoupled Layout

In order to address the limitations of the coupled layout, a finite intermediate buffer can be employed. This layout

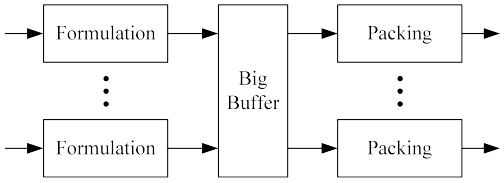

Fig. 2. Decoupled Layout of the Make-and-Pack Plant.

is schematically depicted in Fig. 2. While an order is being processed in a formulation unit the products are sent to a shared intermediate buffer before being packed. In this way both units can operate at their maximum processing rate since they are no longer directly coupled. On the one hand, if the formulation is faster, packing can operate at its maximum rate while the unpacked products are stored in the buffer. After the formulation finishes, the remaining products can be packed from the buffer. On the other hand, if packing is faster the formulation can start earlier and build up an inventory in the buffer. Once enough products are available, packing can start operating. Note that due to the different, unsynchronized processing rates the formulation stage can not supply the packing stage directly. The transfer time between the stages is negligible compared to the processing times. The only important restriction is that packing can not start or end before the formulation, as products that have not been produced yet can not be packed and packing cannot stop if there are still products that have to be packed. In addition to the decoupling of the processing rates, the changeovers are also decoupled. A formulation unit can be in operation supplying the buffer while a changeover is being performed in its corresponding packing unit and vice versa. Overall, this layout can be regarded as a hybrid flow shop with intermediate buffer. A similar decoupled layout of the plant in which each formulation unit could supply each packing unit was examined in Yfantis et al. (2019). However, in this case a formulation unit could not supply both the buffer and a packing unit, necessitating the use of the lower (synchronized) production rates if the buffer was not used. The scheduling problem was solved using an MILP model and a decomposition strategy, since a direct solution was not possible due to the large model size and complexity. In this paper, we assume that each formulation line is coupled to one packing line, i.e. not the full flexibility of an intermediate buffer is used. Nonetheless, the layout with the buffer already provides significant room for improvements compared to the layout without the buffer.

3. SIMULATED ANNEALING-BASED ALGORITHM

In this paper a simulated annealing (SA) algorithm is used to solve the scheduling problems. The SA algorithm is generally well suited for industrial applications, as it exhibits good performance with comparatively low implementation effort. In the following, the main elements of the algorithm are presented.

3.1 Principles of Simulated Annealing

SA is based on the analogy of physical annealing of materials, in which a solid is cooled starting from a high

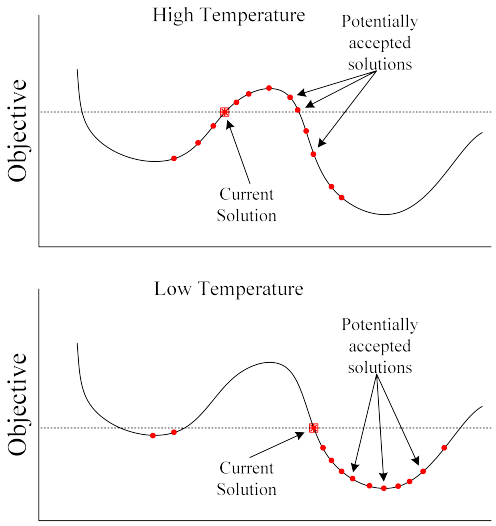

Fig. 3. Illustration of the simulated annealing algorithm for a minimization problem.

temperature, where its particles can move freely and randomly. The temperature is then steadily decreased, limiting the mobility of the particles until reaching a state of minimum energy (Delahaye et al., 2019). In the SA-algorithm, a temperature T is defined analogously, determining the probability that a worse solution with respect to the objective value of the optimization problem that was reached by some neighborhood step from the current solution, is accepted. While the temperature is high, the probability of accepting a worse solution is high and it decreases as the temperature decreases. The reasoning behind accepting worse solutions is the possibility to escape from local optima, thus approaching a globally optimal solution. The concept of the algorithm is illustrated in Fig. 3 for a minimization problem, where initially solutions with higher objective values can be accepted. In order to accept a worse solution a probability based acceptance criterion is used,

$$P(T,k) = \begin{cases} 1 \\ \text{if } f(x_k) < f(x_{k-1}), \\ \\ exp\left(\dfrac{f(x_{k-1}) - f(x_k)}{T}\right) \\ \text{if } f(x_k) \geq f(x_{k-1}). \end{cases} \tag{1}$$

In the case of the minimization of a function $f(x)$ the solution in the current iteration k, i.e. x_k, is accepted if the objective value decreases from the previous iteration $k-1$. If the current solution gives a worse objective value, it is only accepted if the value of the acceptance probability function is larger than a uniformly distributed random number between 0 and 1, i.e.

$$x_k \text{ accepted, if } P(T,k) \geq \text{rand}[0,1].$$

The temperature T decreases as the iteration progresses according to a specified cooling schedule, which can be based e.g. on the number of iterations or the evaluations of the acceptance probability function. In this paper the next temperature is computed after each iteration according to

$$T = T_0 \cdot A^h, \tag{2}$$

where T_0 is the initial temperature, $0 < A < 1$ is a constant parameter and the exponential factor h is increased by 1 after a specified number of iterations. The SA-algorithm terminates when a temperature threshold is reached, as it evolves into a random search, once the temperature is too low.

One of the main challenges for the use of metaheuristics in scheduling is the treatment of complex constraints in the problem formulation. In the case of exact approaches, like mathematical or constraint programming, the constraints are included explicitly in the optimization model and the used algorithms guarantee feasibility with respect to these constraints, if a solution is found. In the case of metaheuristics, like SA, constraints can also be included explicitly as parts of the objective function, penalizing their violation. This however may lead to unintuitive solutions, as the choice of the corresponding weights for the violations or the form of the objective function in general is not trivial. Another possible approach is to check the feasibility of a generated solution and discard it if it violates any of the constraints. Even though the final generated solution would be feasible, it might be very challenging to even find such a feasible solution. Simply discarding an infeasible solution does not provide any feedback with regard to the direction in which the search should focus. Therefore instead of completely discarding the schedule, various repair heuristics have been proposed that attempt to modify the infeasible schedule in a way that it satisfies the constraints (Coelle Coello, 2002). All of the approaches mentioned above can be computationally demanding. In this contribution, an approach similar to the one in Bousonville (2002) is employed, in which the actual algorithm has only a limited number of degrees of freedom which are then provided to a "scheduler". This scheduler generates a feasible schedule based on a process model. To be more precise, the SA-algorithm determines the allocation of a set of orders $\mathcal{I}$ to a set of units $\mathcal{J}$ and the sequence of orders $\mathcal{I}_j$ therein. [1] In the following both elements of the algorithm, the schedule generation and the SA-algorithm are introduced.

3.2 Sequence-based Schedule Generation

The goal of the schedule generation is to provide a feasible schedule, given a totally ordered set of orders $\mathcal{I}_j$ allocated to each unit $j \in \mathcal{J}$. Each order is characterized by an ordered amount D_i that has to be produced to satisfy the customer demand and a processing rate v_i. From these input parameters the processing time for each order can be calculated by

$$p_i = D_i / v_i. \tag{3}$$

Based on the processing time and the changeover time between two consecutive orders $co_{i,i+1}$ the start and end times of processing of each order S_i and E_i can be calculated recursively in each line j as

$$S_i = 0, \text{ for } i = \mathcal{I}_j[1], \tag{4}$$
$$E_i = S_i + p_i, \ \forall i \in \mathcal{I}_j, \tag{5}$$
$$S_{i+1} = E_i + co_{i,i+1}, \ \forall i \in \mathcal{I}_j, \tag{6}$$

where $\mathcal{I}_j[1]$ denotes the first order of the sequence $\mathcal{I}_j$. The objective is to minimize the makespan, i.e. the largest end time,

$$\min \max_{\forall i \in \mathcal{I}} E_i. \tag{7}$$

[1] The order of the elements in the sets $\mathcal{I}_j$ determines the sequence in the schedule.

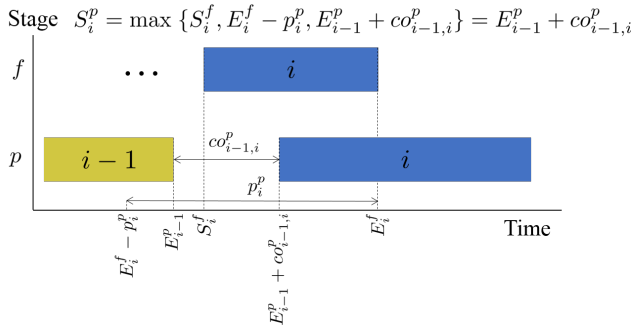

Fig. 4. Example of the computation of a start time in the packing stage.

It should be noted that other objective functions like the total completion time can easily be incorporated as well, e.g.

$$C_j = \max_{\forall i \in \mathcal{I}_j} E_i, \tag{8}$$

$$\min \sum_{\forall j \in \mathcal{J}} C_j. \tag{9}$$

3.3 Modifications for the Decoupled Layout

In the decoupled layout, the start and end times of the processing tasks have to be calculated for both, the formulation and the packing stages. The variables in equations (3)-(6) are defined for each stage $l \in \mathcal{L} = \{f, p\}$, i.e. S_i^l, E_i^l, p_i^l, v_i^l and $co_{i,i+1}^l$. The variables corresponding to the formulation stage ($l = f$) are computed by equations (3)-(6). The timing of the packing stage is restricted by the formulation stage, as packing can not start or end before the formulation. Furthermore, packing can not start before the changeover from the previous packing task has elapsed. The start time of packing of order i is therefore computed by

$$S_i^p = \max\{S_i^f, E_i^f - p_i^p, E_{i-1}^p + co_{i-1,i}^p\}. \tag{10}$$

An example, in which the changeover time determines the start time, is shown in Fig. 4. Once the start time is known, the end time can be computed by Eq. (5). In addition to the interaction of the timing decisions, the amount stored in the intermediate buffer has to be computed. This is achieved by discretizing the time horizon into finitely many time intervals $t \in \mathcal{T}_H$ and employing a simple material balance to compute the stored amount Inv_t,

$$Inv_{t+1} = Inv_t + \sum_{\forall i \in \mathcal{I}} (Y_{it}^f \cdot v_i^f - Y_{it}^p \cdot v_i^p), \tag{11}$$

where the binary variables Y_{it}^l are set to one if order i is being processed in stage l at time t.

3.4 Two-stage Simulated Annealing Algorithm

The SA-algorithm is divided into two stages. The first stage determines the allocation of orders to the units while the second stage performs a refinement of the schedule by determining the sequencing within each line. They can also be regarded as an initial exploration and a subsequent exploitation stage. In each iteration in both stages of the algorithm, a set of sequences $\mathcal{I}_j$ is generated which is used to generate a schedule using Eq. (4)-(6) for the coupled layout, while additionally Eq. (10)-(11) are used for the decoupled layout. The general scheme of the algorithm, which is conceptually identical in both stages, is illustrated in Fig. 5. In each iteration a permutation of the sequence is performed resulting in a new schedule, the objective value of which is evaluated. If the schedule is better than the current solution, the current one is overwritten. If additionally the schedule is better than the best solution found so far, the best solution is overwritten as well. If however the schedule is worse than the current solution, the acceptance probability function (1) is evaluated. If the condition is satisfied, the current solution is overwritten. After each iteration the temperature T is recomputed by (2), where h is increased by one after a fixed number of iterations. The main difference between the two stages is the permutation which generates a new solution. In the first stage, each order i is randomly allocated to a feasible line $j \in \mathcal{J}_i$, i.e. a unit that can process order i, according to the current sequence. This allocation results in a new sequence for each line. The sequence within each line is not explicitly optimized in the first stage, since the allocation of orders to lines has a greater impact on the objective than the exact sequencing of the orders. The new sequences are implicitly determined by the previous sequences. Once the temperature threshold is reached for the first stage, the allocation is fixed to the best found solution and the sequence within each line is optimized. In the second stage the permutation of the sequence is performed by changing the position of a random order within the sequence. In the case of the decoupled layout a solution is discarded if the amount stored in the buffer exceeds its maximum capacity, i.e. if

$$\max_{\forall t \in \mathcal{T}_H} Inv_t > Cap. \tag{12}$$

4. RESULTS

The SA-based scheduling algorithm was tested on two different real-life case studies of increasing complexities using actual plant data. Both cases included 3 lines and differ in the number of orders (Case 1: 36, Case 2: 62), The algorithm was implemented in Julia 1.0.3.1 (Bezanson et al., 2017) on a MS Windows 10 desktop PC (intel® Core™ i7-8700CPU @ 4.30 GHz, 6 Cores, 12 logical processors, 16 GB RAM). The cooling schedule was given by $T_0 = 15000$, $A = 0.9$ and h was increased every 1000 iterations for both layouts. Due to the stochastic nature of the algorithm, 10 optimization runs were performed for each case and for each plant layout. The results of the makespan minimization are summarized in Table 1, where they are compared to nominal schedules provided by the planners at the plant. The results show that the makespan was decreased for all examined cases. The decoupled layout provides better results due to the increased flexibility and the higher production rates compared to the coupled layout. The fact that more degrees of freedom are available increases the complexity of the scheduling problem but also enables improvements of the makespan. The mean CPU times for the examined cases are summarized in Table 1. The nominal schedules as well as exemplary Gantt charts for each case are shown in Fig. 6-9. The Gantt charts show that the algorithm provides schedules in which the different lines finish processing approximately at the same time, indicating a balanced utilization of the available

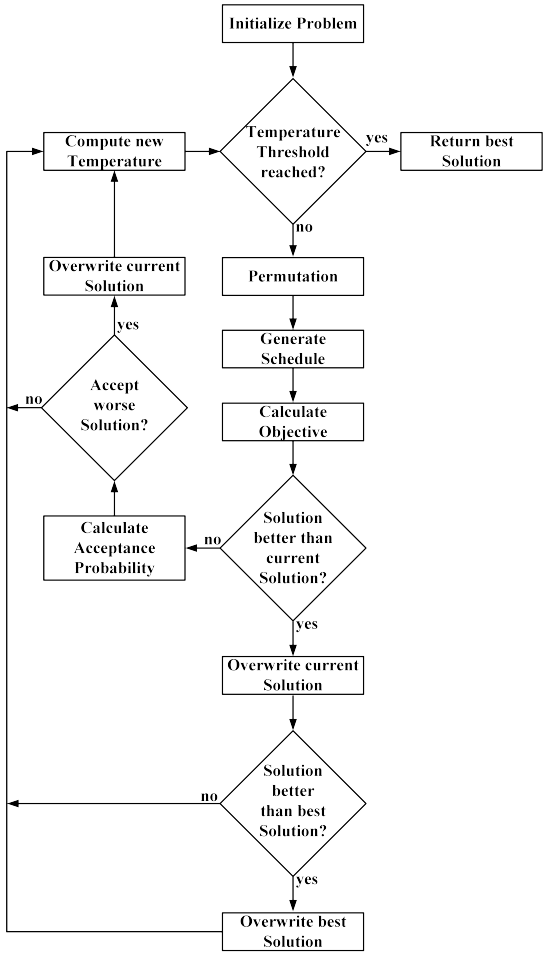

Fig. 5. Simulated annealing-based optimization algorithm.

resources. This is very significant, as the examined process is continuous, i.e. production continues immediately after the execution of the current schedule. For the decoupled layout the evolution of the amount stored in the buffer is also shown.

Table 1. Results of the SA-based optimization.

Case		MS $[min]$	Impr. [%]	CPU Time $[s]$
1	Nom.	3202	–	–
	Coup.	3127 ± 13	2.34	1742
	Dec.	2796 ± 38	12.68	1828
2	Nom.	5371	–	–
	Coup.	4548 ± 46	15.32	1940
	Dec.	3876 ± 57	27.93	2802

As mentioned above, the SA-based algorithm is stochastic in nature, therefore each run results in a different solution. The evolution of the objective value of the best found solution for the decoupled layout of Case 2 is shown in Fig. 10 for ten different runs in order to evaluate the reproducibility and the performance of the algorithm. The results show that a drastic improvement of the solution occurs in the first iterations. After around 50000 iterations, or approximately 5 min of CPU time, a solution that is very near to the final solution is already found. This indicates that the algorithm can be used to generate good schedules even within short computation times, if necessary. This figure also illustrates the effect of the two

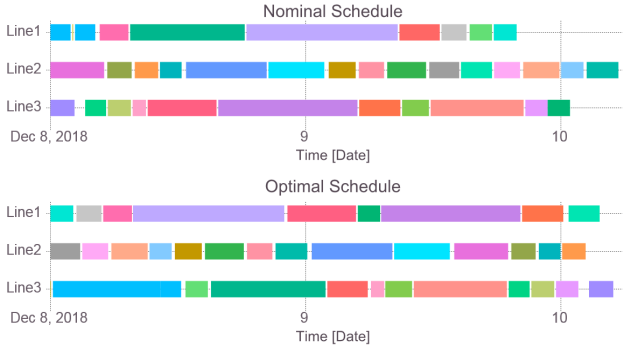

Fig. 6. Case 1: Nominal and optimal schedules for the coupled layout.

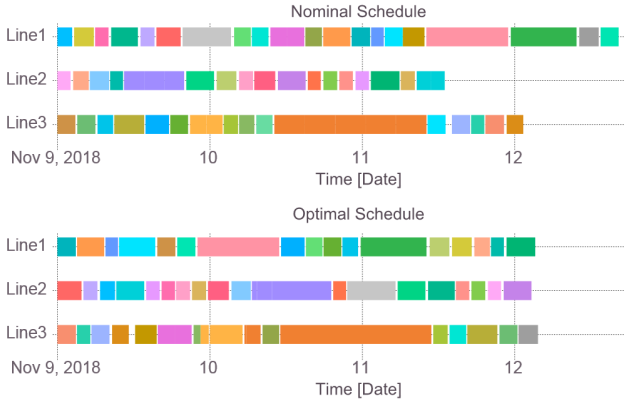

Fig. 7. Case 2: Nominal and optimal schedules for the coupled layout.

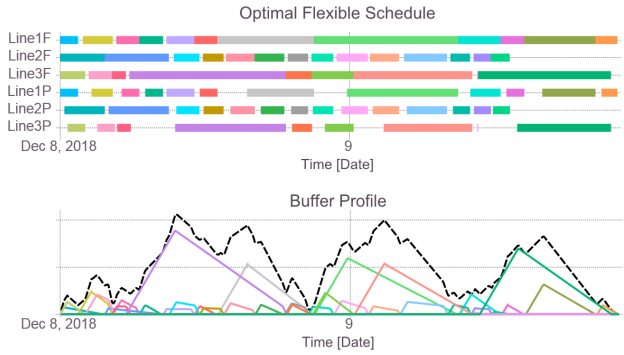

Fig. 8. Case 1: Optimal schedule and buffer profile for the flexible layout.

stages (allocation and sequencing). The main improvement of the soultion occurs in the allocation stage. After around 150000 iterations, the sequencing stage starts and refines the schedule further. All optimization runs converge to similar objective values (see Table 1), indicating the reproducibility of the results.

5. CONCLUSION

This contribution presented a two-stage SA-based algorithm for two different layouts of a make-and-pack consumer goods production plant. The solution algorithm was split into an allocation and a sequencing stage in order to reduce the computational complexity. The results for two different case studies show clear benefits for both layouts

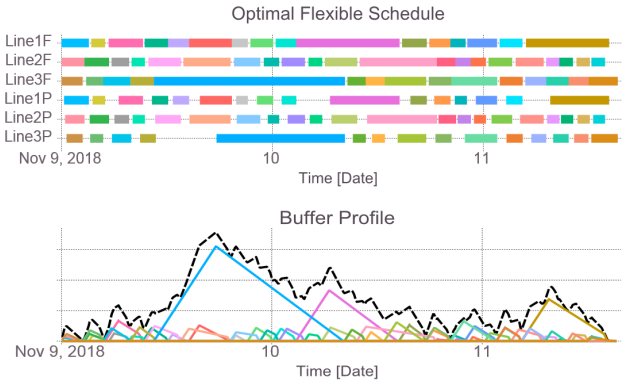

Fig. 9. Case 2: Optimal schedule and buffer profile for the flexible layout.

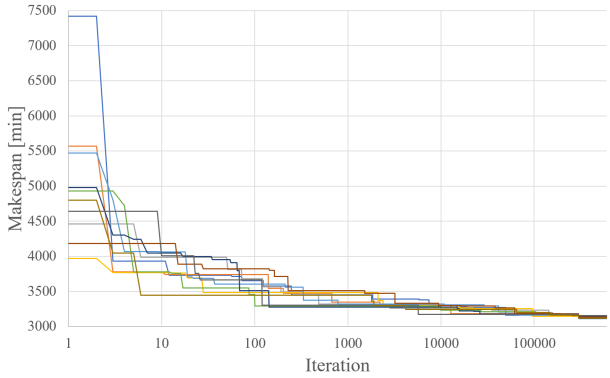

Fig. 10. Reproducibility of the optimization results for the coupled layout of Case 1.

compared to the nominal schedules. Furthermore, the algorithm exhibits a good reproducibility and can provide good solutions even in very short computation times, making it suitable for online scheduling applications.

A main drawback of the algorithm when applied to the decoupled layout is the fixed assignment of formulation lines to packing lines, as this significantly reduces the flexibility of the plant. Since all units of the formulation stage are identical and can process any order, additional flexibility can be added by allowing different allocations of the orders in the two production stages. This would necessitate modifications to the sequence-based schedule generation. Furthermore, the splitting of the scheduling problem into two separate optimization stages also decreases the improvement potential, as it limits the degrees of freedom of the optimization. In order to address this drawback while still keeping the problem computationally tractable the two stages could be solved in an integrated fashion. In each iteration the allocation could be performed by a metaheuristic algorithm and a simplified MILP could be used to compute the optimal sequence for the given allocation.

REFERENCES

Baumann, P. and Trautmann, N. (2014). A hybrid method for large-scale short-term scheduling of make-and-pack production processes. *European Journal of Operational Research*, 236(2), 718–735. doi:10.1016/j.ejor.2013.12.040.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V.B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1), 65–98. doi:10.1137/141000671.

Bousonville, T. (2002). The two stage continuous parallel flow shop problem with limited storage: Modeling and algorithms. In P. Collet, C. Fonlupt, J.K. Hao, E. Lutton, and M. Schoenauer (eds.), *Artificial Evolution*, volume 2310 of *Lecture Notes in Computer Science*, 180–191. Springer, Berlin and Heidelberg.

Coelle Coello, C.A. (2002). Theoretical and numerical constraints handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, (191 (11-12)), 1245–1287.

Delahaye, D., Chaimatanan, S., and Mongeau, M. (2019). Simulated annealing: From basics to applications. In M. Gendreau and J.Y. Potvin (eds.), *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, 1–35. Springer International Publishing, Cham.

Elekidis, A.P., Corominas, F., and Georgiadis, M.C. (2019). Optimal short-term scheduling of industrial packing facilities. In *29th European Symposium on Computer Aided Process Engineering*, volume 46 of *Computer Aided Chemical Engineering*, 1183–1188. Elsevier. doi:10.1016/B978-0-12-818634-3.50198-3.

Harjunkoski, I., Maravelias, C.T., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J., Méndez, C., Sand, G., and Wassick, J. (2014). Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering*, 62, 161–193. doi:10.1016/j.compchemeng.2013.12.001.

Kopanos, G.M., Méndez, C.A., and Puigjaner, L. (2010). Mip-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. *European Journal of Operational Research*, 207(2), 644–655. doi:10.1016/j.ejor.2010.06.002.

Kopanos, G.M., Puigjaner, L., and Maravelias, C.T. (2011). Production planning and scheduling of parallel continuous processes with product families. *Industrial & Engineering Chemistry Research*, 50(3), 1369–1378. doi:10.1021/ie100790t.

Méndez, C.A. and Cerdá, J. (2002). An milp-based approach to the short-term scheduling of make-and-pack continuous production plants. *OR Spectrum*, 24(4), 403–429. doi:10.1007/s00291-002-0103-5.

Sand, G., Till, J., Tometzki, T., Urselmann, M., Engell, S., and Emmerich, M. (2008). Engineered versus standard evolutionary algorithms: A case study in batch scheduling with recourse. *Computers & Chemical Engineering*, 32(11), 2706–2722. doi:10.1016/j.compchemeng.2007.09.006.

Wang, K., Löhl, T., Stobbe, M., and Engell, S. (2000). A genetic algorithm for online-scheduling of a multiproduct polymer batch plant. *Computers & Chemical Engineering*, 24(2-7), 393–400. doi:10.1016/S0098-1354(00)00427-0.

Yfantis, V., Corominas, F., and Engell, S. (2019). Scheduling of a consumer goods production plant with intermediate buffer by decomposition and mixed-integer linear programming. *IFAC-PapersOnLine*, 52(13), 1837–1842. doi:10.1016/j.ifacol.2019.11.469.