Reactive Real-time Scheduling Using Simulation-Optimization and Evolutionary Algorithms

Engelbert Pasieka*, Sebastian Engell*

***Abstract*— Industrial scheduling is an important task that involves the allocation of the orders to resources and the sequencing and timing of the operations to meet delivery dates and to improve the performance of the production system. Most of the research in this area addresses static or offline scheduling problems, i.e. solving scheduling problems for a given set of resources, orders, and due dates, and focuses on the efficiency of the solution process. However, in industrial operations each schedule is outdated shortly after it has been computed due to unforeseen events, e.g. new orders or due dates, lack of materials, maintenance, etc. Often the precomputed schedule does not meet the constraints any more or cannot be executed at all because resources are (temporarily) not available. Then in practice manual re-scheduling takes place. In this contribution we discuss how a static scheduling system can be redesigned for use in dynamic situations, i.e. for reactive scheduling.**

Our approach builds on previous work on production planning using simulation-optimization. It combines a discrete-event simulator with a tailored evolutionary algorithm to implement a dynamic adaptation of the schedules to the available information. The simulation model is continuously updated with the latest information about the state of the plant and of the orders. At the beginning of a cycle the evolutionary algorithm evaluates all schedules of the current population with this model and then improves them. We demonstrate our solution for a pharmaceutical batch plant example with a high combinatorial complexity. Our study demonstrates that the simulation-optimization framework can effectively handle unforeseen events and provides good solutions fast.

***Index Terms*— Discrete-Event Systems, Scheduling, Combinatorial Optimization**

I. INTRODUCTION

Production scheduling is an important decision-making process that improves the productivity and competitiveness of production plants by creating schedules that utilize the available resources in an optimal manner while meeting constraints, e.g. due dates for delivery. It involves to determine the allocation of orders to resources and the sequencing and timing of the operations. In principle, scheduling problems can be solved by rigorous methods as e.g. mixed-integer programming or constraint programming [1]. For real-world problems, however, the application of such methods is limited by the combinatorial explosion. Therefore the problems usually are decomposed in various ways [2, 3, 4] and the description of the real production processes is to some degree abstracted, i.e. it does not contain all details of the real production process. This leads to the problem that because of neglecting constraints that are relevant in the real plant (in particular the availability of further resources that are necessary for the execution of tasks) the computed schedules may not be executable without manual intervention.

The advantage of simulation-optimization is that in the simulation of the production processes, very detailed models can be used. Thus the simulated schedule can be executed without major modifications. On the other hand, such detailed models are in most cases not suited for rigorous optimization, due to their complexity and due to the effort needed to translate them into mathematical formalisms. Therefore in combination with simulation, usually meta-heuristics are applied that modify the most important degrees of freedom of the simulation (typically allocation and sequencing decisions) and evaluate the proposed choices by simulation. The simulator then contains execution rules that take care of the remaining degrees of freedom [5, 6, 7].

Both classes of optimization approaches to scheduling, rigorous methods and simulation-optimization, share the feature that the computation of a new schedule or production plan takes a significant period of time, usually in the order of magnitude of hours for real-world problems.

In reality, soon after the execution of a new schedule has started, unplanned deviations occur, from minor ones (longer or shorter execution or changeover times) to major disruptions as e.g. temporary equipment breakdowns, rush orders or new due dates.

Therefore rescheduling is necessary, which involves updating existing production schedules to respond to disruptions or changes in the production environment [8]. It can be event-based, triggered by specific occurrences of events, scheduled at fixed intervals, or both. Rescheduling has to ensure to maintain the flow and efficiency of the production process. It helps minimize impact of disruptions on the performance of the system, thereby reducing operating costs and improving productivity. By continuously synchronizing the production plan with the current state of the manufacturing system, rescheduling facilitates better resource utilization and coordination among various production activities.

Espinaco and Henning in [9] discussed the current state and challenges in industrial rescheduling, particularly in the context of the discrete manufacturing domain, against the backdrop of Industry 4.0. They identified significant gaps between academic proposals and industrial needs in rescheduling. Rescheduling methodologies must provide timely responses, address computational challenges, and provide feasible, if possible optimal solutions. The automatic development of rescheduling models and better integration with industrial systems are identified as crucial for the broader adoption of advanced rescheduling solutions in the context

*Process Dynamics and Operations Group, TU Dortmund University, Emil-Figge-Strasse 70, 44227 Dortmund, Germany. {engelbert.pasieka, sebastian.engell}@tu-dortmund.de

of Industry 4.0.

Schoppmeyer presented in [10] a reactive scheduling solution for real-time and real-world applications. It is proposed to establish an Immediate Response System (IRS) that generates fast decisions on the next actions while a rescheduling algorithm computes a new schedule over a limited prediction horizon fast, and periodically new schedules are computed over a longer prediction horizon. He proposed to employ scheduling based on timed-automata models [11] for short-term rescheduling. Timed-automata models are less expressive than simulation and rigorous models to use them for scheduling is limited to medium sized problems.

Recent mathematical programming approaches utilize state space formulation for online scheduling. These algorithms use state feedback and control, similar to how model predictive control functions. However, without advanced decomposition approaches, they are also limited to medium-sized problems [12].

Our approach addresses the challenges in rescheduling by utilizing simulation-optimization with detailed production models to enhance real-time decision-making on the shop floor and to facilitate the transferability of solutions to the actual process. It uses a dynamic rescheduling system that processes new information quasi-continously. An evolutionary algorithm continuously runs in parallel to the execution of the production process and improves the schedules based on the most recent information. The next decisions are taken based upon the currently best available solution. This constitutes a novel method for real-time reactive scheduling that leverages the features of simulation-optimization to generate good solutions fast for computationally complex problems. We demonstrate the effectiveness of our method with a case study in pharmaceutical batch production, a process that shares characteristics with many problems from the process industry.

II. SIMULATION-BASED DYNAMIC SCHEDULING BY AN EVOLUTIONARY ALGORITHM

Our methodology integrates simulation-optimization with dynamically adapted simulation models and a tailored evolutionary algorithm (EA) to address the dynamic nature of production scheduling. The production process is represented by a high-fidelity discrete-event simulation model that is used to obtain a detailed simulation of the execution of the proposed schedules. The simulator also provides a visualization of the currently best schedule to enable an assessment of the current state and the planned operations of the production. The scheduling system optimizes continuously between update events to always have a set of good solutions available for a warm start of the optimizer after disruptive events during plant operation. The interaction between the scheduling system and the control center of the production process is shown in Figure 1. The production units send plant data to the control center, which extracts the plant state information and forwards it to the scheduling system. The scheduling system updates the simulator model and modifies the representation of the scheduling problem in the EA if needed. By using

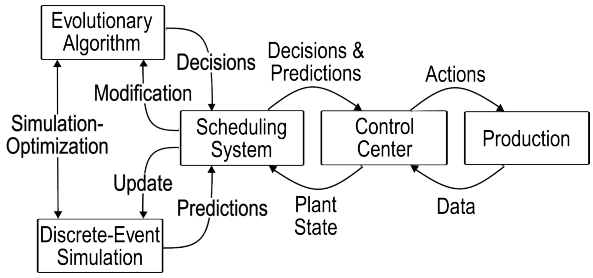

Fig. 1. Interaction between the scheduling system and the control center of the production process.

a simulation-optimization approach, the EA generates the major decisions and the simulator provides a detailed look-ahead that defines the allocation, sequencing and timing of the operations and resources, including resources that are not explicitly modelled in the scheduling system, e.g. tasks of the operators. The planned operation is then sent back, via the scheduling system, to the control center, which checks and implements the proposed actions.

A. Simulation-Optimization

The EA holds a population of individuals that represent the decisions in a so-called genome that the simulator uses to generate a detailed schedule for the future. In order to maintain a manageable size of the search space, the EA handles only the most important degrees of freedom, the remaining decisions are taken by the inherent logic or by heuristic decision rules that are defined in the simulation model and executed by the discrete event simulator [13]. As an example, the EA might optimize the allocation of operations to units and a global priority order of the different jobs (or production orders) and in the simulation the execution order of the operations which can be started for a production unit follows the global priorities, and the starting times are defined by the ”as soon as possible” heuristic.

The EA is tailored for dynamic scheduling and therefore has a different structure than the one described in our previous work [13]. Figure 2 shows a flowchart of the tailored EA. A cycle of the EA starts with collecting data about the production process and the orders and due dates. Based on this information, the simulation model and the degrees of freedom of the EA are updated. E.g., if additional orders have to be handled, the current set of jobs are updated, and temporal unavailabilities of resources are specified in the simulation model. If the EA handles the assignment of individual operations to production units, these decisions are removed from the genome if the operations have already been started. The simulator is initialized with the state of the units and the operations that have been started including the remaining processing time and the operations that wait for execution.

At the beginning of the optimization, the genomes are initialized randomly according to the degrees of freedom which are available at the initial time. In the following cycles, the individuals of each generation are the best candidates among the members of the previous generation and their

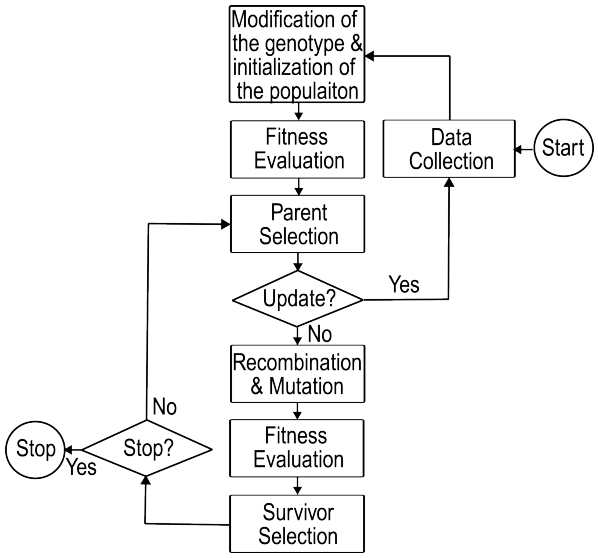

Fig. 2. Tailored evolutionary-algorithm for simulation-optimization-based real-time dynamic scheduling with model updates.

offspring which are generated by mutation and crossover, steered by the performance of the individuals.

Before modification, i.e. the generation of offspring by mutation and crossover, the EA checks whether or not updates from the control center have to be considered. Here it can be distinguished between major changes that need an immediate reconfiguration of the genotype and re-initialization of the population or minor changes that are ignored until a certain period of time or number of generations of the EA have passed. If updates have to be considered, the genotype is adapted and the EA is reinitialized. This reinitialization is based upon the latest population of individuals, i.e. the best schedules that had been found for the pre-existing situation. During reinitialization, the genome structure of the individuals in the population only change if the degrees of freedom change, e.g., if a new order has to be considered; otherwise, they remain the same as in the last generation.

The way the reinitialization is done is specific for the type of update. E.g. if new orders become known, they can be placed randomly in a global sequence or according to the order of the due dates. Allocations of operations to units that are not available any more can be distributed randomly to the units that still are available. Changes of due dates can be ignored, i.e. they only change the outcome of the evaluation of the genomes, or lead to a directed change of sequences according to the due dates or the due dates minus the remaining processing time. Changes in the operation times of orders can be considered by adopting parameters of the simulation model. After the reinitialization, the population is evaluated with the updated simulation model.

After the structure of the genomes and the values of the entries have possibly been changed and the evaluation with the updated model has been finished, the EA modifies the population in the usual manner. The modification includes parent selection, genome mutation and crossover between two parent individuals. After modification, the offspring are

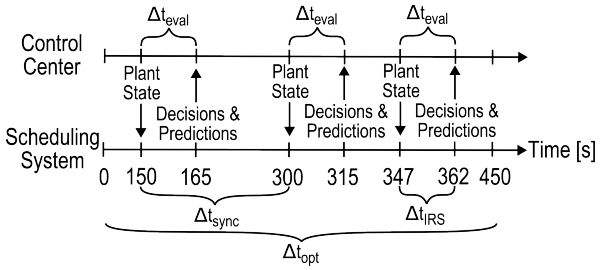

Fig. 3. Information exchange during updates between the control center and the scheduling system.

evaluated, and the survivor selection takes place. The EA will continue its iterations if not terminated, if a termination condition is met, e.g. a certain number of generations without improvements, it will wait for the next update.

B. Update of the scheduling model

Two events initiate model updates: periodic synchronization and disruptive events. Figure 3 shows this update process. The information coming from the control center considers the current plant state and everything that happened before the time of the update. This includes the list of orders, the started operations of orders, and the resource states, e.g., busy, idle, or in maintenance. This information is used to adjust the simulation model. The simulation considers the current update time as its starting point and executes the model until the end of the schedule. Running operations before the update are adjusted in the model (i.e. possibly changing the earliest starting times of subsequent operations and the expected duration of operations) such that it accurately reflects the current plant state. Operations that have been started and finished before the update are no longer considered in the simulation. The minimum interval for the periodic update is the termination of the evaluation and survivor selection of the current generation (Δt_{eval}). Longer intervals, e.g. for synchronization (Δt_{sync}), can be chosen to reduce the nervousness of the scheduling system. Immediate response actions (in the interval Δt_{IRS}) are executed as fast as possible. The interval for the communication of new schedules to the control center is a degree of freedom as a certain number of generations of the EA is needed to stabilize the solution.

III. CASE STUDY

The case study that is used here to test the proposed real-time reactive scheduling approach is a pharmaceutical batch plant model that was introduced by Kopanos et al. in [3]. The batch plant has many similarities with other problems from the process industry (e.g., restricted eligibility of resources, changeover times, etc.). They developed a two-stage mixed-integer programming decomposition strategy that is applicable to problems of realistic size. In the first, the ”constructive step”, orders are added to the schedule one at a time based on a specific rule. The second stage, the ”improvement step”, improves this schedule using mixed-integer linear programming.

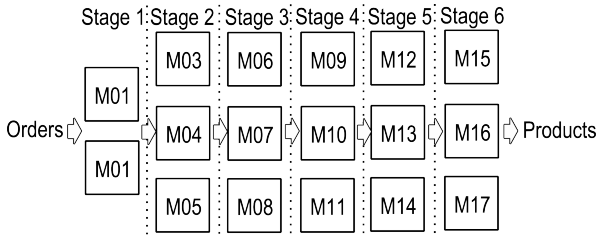

Fig. 4. Pharmaceutical batch plant layout of the case study described in [3]. The processing of the orders starts at the first stage and proceeds successively stage by stage. The unit eligibility depends on the recipe. The storage policy between the stages considered in this work is unlimited intermediate storage.

Figure 4 shows the plant layout of the case study. 10 different products are produced using six successive production stages that comprise 2 or 3 units, 17 units in total. The plant operates in a hybrid flow shop mode. Operations cannot be preempted. Key features of the pharmaceutical batch plant are multiproduct layout, variable equipment assignment, unlimited intermediate storage, instantaneous material transfer, fixed batch sizes, machine-dependent processing times, sequence-dependent changeovers, operational and changeover costs, and order due dates. The operations are performed sequentially with the products passing from one stage to the next. On each stage there is a set of units that are available to process the orders. It is possible to skip stages (according to the production recipes) and certain operations may only be executed on certain units of a stage. The durations of the processing times of the different units for the same task can be different and between two operations of different orders changeover times are necessary.

In the case considered by Kopanos et al., there are 30 orders for the 10 products with different due dates. A more detailed description of the problem and the necessary parameters for the replication of the study can be found in [3]. We modified the case study by adding rush orders that arrive as new orders of existing products in addition to the 30 orders of the base case. Furthermore, faults and subsequent maintenance which blocks certain units for a predicted time are included. The prediction of the repair time may also change dynamically.

Our study focuses on minimizing the tardiness in a real-time scheduling context with disruptive events, initially starting with 30 orders (P01 to P30), each with a due date ranging from 20 to 40 hours after the start time. The tardiness objective T is the sum of the delays of all orders defined as

$$T = \sum_{i=1}^{N_{Orders}} max(t_{i,end\,date} - t_{i,due\,date}, 0) \quad (1)$$

The exchange of decisions and predictions from the scheduling system to the control center is performed after every update, e.g., after synchronization or disruptive events, to provide the control center as fast as possible with predictions and decisions, and to generate timely responses. The periodic update interval between the scheduling system and the control center of the production is set to 2.5 minutes. Every order has 5 to 6 operations and 4 to 5 changeovers to consider. This leads to about 330 operations in the time span of 30 hours that it takes to finish 30 orders. So on the average there are 11 operations started and 11 operations finished per hour, which combines to approximately one event every 2.7 minutes. The timings of these operations motivates a synchronization every 2.5 minutes. At these times, the parameters of the simulation are updated to the current state of the plant before the current population is evaluated. After the update and one cycle of the EA, the best solution is provided to the control center. In this simulation, there are no uncertainties in the duration of operations, we focus on the disruptive events for clarity. The control center receives the result of the simulation of the schedules with the updated model and the improved decisions and from this information derives and implements the next decisions, e.g. which operation to start on a unit that becomes available and where operators have to be present at which time in the future.

Updates initiated by disruptive events are treated differently. First, the parameters of the simulation model and the genome type are adjusted. Then the last population is updated such that it is adapted to the new situation. The schedule system performs one cycle of the EA and transfers the solution to the control center.

In between updates, the scheduling system continues to optimize the schedules. The current best solutions are transferred to the control center one generation after the updates and disruptive events. The evolutionary algorithm maintains a population of 20 individuals with 20 offspring and 20 parents per generation. The offspring are evaluated using a discrete-event simulator, in this case INOSIM 14.0 from INOSIM GmbH, a commercial software. In each generation, the offspring are evaluated in parallel on a machine with an Intel(R) Core(TM) i9-13900K processor with 24 CPU cores and 64GB of RAM. The maximum possible number of parallel simulations on this machine is 20, which motivated the choice of 20 offspring per generation. On average, 20 parallelized fitness evaluations take about 15 seconds.

The EA utilizes a global sequence of orders that defines the execution on all stages, e.g., $\pi = \{$P01,P02,...,P30$\}$. The simulator decodes this genome by prioritizing the orders as they appear in the global sequence and applies this priority order for all decisions on which operation to start next on a unit and when to start the operation. The simulator dynamically handles the allocation and timing of operations to resources via the earliest possible start date heuristic, thus reducing the search space of the EA. In [13] it was shown that managing the allocation and timing dynamically by the simulator leads to an efficient search without sacrificing solution quality.

We use the permutation operator and the random mutation operator designed for sequence encodings [14]. The permutation mutation operator leads to a less drastic genome change and a balance between exploitation and exploration, whereas the random mutation drastically alters the genome, leading to more explorative behavior of the EA. The EA uses

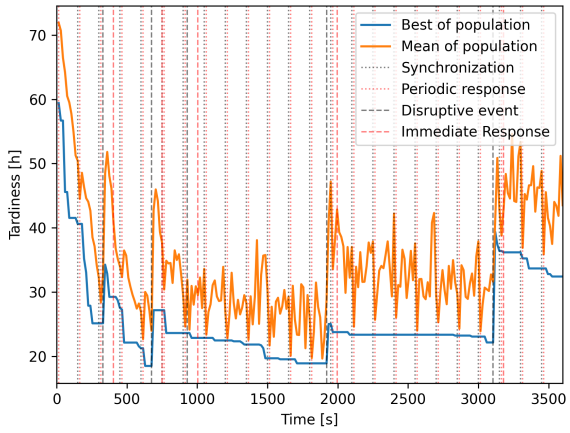

Fig. 5. Minimization of the tardiness of the example system shown in Figure 4. The scheduling system synchronizes with the production system every 2.5 minutes and sends back a valid schedule 15 seconds after the update event. During the hour of operation of the system five disturbances occur (see Table 1). An immediate response is generated 15 seconds, i.e., one cycle of the EA, after the disturbance.

the permutation mutation in 80% of cases and a random mutation in 20% of cases to balance between exploration and exploitation. The crossover here is the Cycle Crossover, which is designed to preserve positional information of two sequences genomes [14].

The parent and survivor selection operators are the same as those used in [13]. This includes a rank-based parent selection and a rank-based/elitist survivor selection where the best individuals of the population are selected for modification or survival. The elitist survivor selection ensures that the best individual always survives the generation, thus preventing the solution from worsening over time.

In Figure 5, one hour of plant operation is shown during which five events require adjustments in both the simulation model and of the optimization problem. The simulation is always performed until all orders are finished to determine the tardiness. The disturbances include unexpected maintenance and rush orders. Details regarding these events including the time of their occurrence are shown in Table 1.

TABLE I

DISRUPTIVE EVENTS THAT ARE HANDLED BY THE SCHEDULING SYSTEM DURING THE SIMULATION SHOWN IN FIG. 5

Time [s]	Event
330	Maintenance on ”M01” for a predicted period of 3 h
675	Arrival of a new order ”P01_Rush” of type ”P01” with a new due date of 20 hours
930	Predicted duration of maitnance on ”M01” reduced to 2.5 h
1920	Arrival of a new order ”P02_Rush” of type ”P02” with a new due date of 15 hours
3105	Maintenance on ”M02” for a predicted period of 4 hours

New orders are placed into the sequence genome at random positions. If, at update time, an order finished its operations, it is removed from the global sequence genome. Maintenance events do not alter the degrees of freedom of the evolutionary algorithm, since the allocation and timing

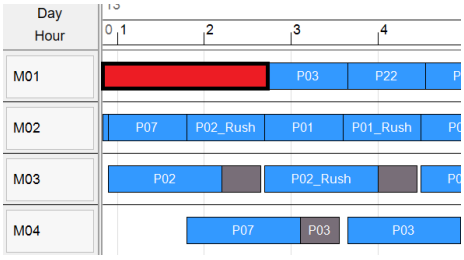

Fig. 6. Excerpt of the schedule that is simulated before the last disruptive event at 3090 seconds. The red filled rectangle is the maintenance on unit M01, the blue filled rectangles are the processing operations, and the grey filled rectangles are the changeover operations in between product changes.

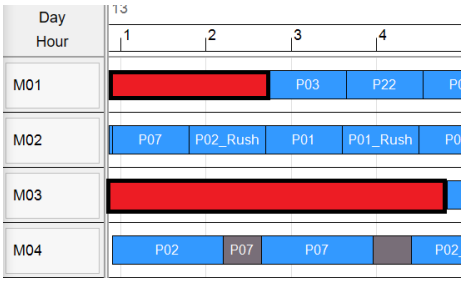

Fig. 7. Excerpt of the schedule that is simulated at 3105 seconds shortly after the last disruptive event and that is provided to the control center as an immediate response. The red filled rectangles are the maintenance activities on the units M01 and M03, the blue filled rectangles are the processing operations, and the grey filled rectangles are the changeover operations in between product changes.

of operations on resources are handled by the simulation heuristically. Maintenance is transferred to the simulation separately by defining the start and end times and on which resource it takes place. During the production process, the EA is running uninterrupted and constantly minimizes the tardiness to provide good solutions at every time and to timely react to unforeseen events.

IV. RESULTS

Figure 5 shows the simulation of the real-time reactive scheduling process over a period of one hour of operation for the disturbances listed in Table 1. The blue line shows the tardiness of the current best solution of the population, the orange line shows the mean tardiness of the population. The dotted black lines represent periodic updates that occur every 2.5 minutes. The dotted red lines represent the periodic responses, which occur one generation after the periodic update, i.e., every 15 seconds after the event. Disruptive events, as listed in Table 1, are shown by black dashed lines. The red dashed line is the time the immediate response takes to submit the solution to the control center. This happens one generation (i.e., 15 seconds) after the event. Throughout the one hour of operation, the best solution is always preserved and the EA constantly minimizes the tardiness.

Figure 6 shows an excerpt of the schedules generated by the scheduling system right before the last disruptive event, i.e., the start of maintenance on ”M3” after 3105 seconds of plant operation. Figure 7 shows the immediate response of the scheduling system to this event as an excerpt of the

schedule that is provided to the control center. The red filled rectangles in Figure 6 and Figure 7 are the maintenance operations that are scheduled on the machines ”M01” and ”M03” after 330 seconds and after 3105 seconds of plant operation. The blue filled rectangles show the timings of processing operations and the grey filled rectangles are the changeover operations in between product changes. The scheduling system reallocates the operations on ”M03”, e.g., ”P02” and ”P02_Rush”, to other eligible resources, i.e., ”P02” is reallocated to ”M04”, thus providing an immediate response to the disruptive event after one cycle of the EA.

Table 2 lists the tardiness of the solutions of the immediate responses right after the disruptive events, i.e., how they are submitted to the control center, and 10 generations later. Figure 8 shows the average best results and the standard deviation of the best results of ten different runs for the events in Table 1.

V. CONCLUSION

We described a new approach to real-time reactive scheduling based on simulation-optimization and using an EA for the optimization. The core idea is to update the simulation model and the representation of the schedules in the EA when disturbances occur and to continuously run the EA to generate increasingly better solutions. The interval at which these solutions are implemented can be chosen flexibly. We demonstrated our approach for the model of a pharmaceutical batch plant from [3] and showed how the scheduling system and the control center interact to implement dynamic real-time rescheduling. The EA maintains a set of potential solutions that are used for re-initialization when changes occur.

In real-world cases the primary limitation of simulation-optimization is that the execution of high-fidelity models is computationally demanding. To mitigate the delay caused by complex simulations, a possible strategy is to reduce the scheduling horizon and to implement a moving window approach. This however comes with specific problems as discussed e.g. in [15]. In our study, we only considered rush orders and maintenance with fixed durations, other disruptive events may necessitate other strategies to react efficiently. Further investigations on this topic as well as the consideration of more realistic case studies are the subject of our future research.

TABLE II

TARDINESS AFTER DISRUPTIVE EVENTS AND 10 GENERATIONS LATER

Time [s]	Tardiness [h]	Event
345	34.27	Maintenance on ”M01”
495	22.16	Maintenance on ”M01” (10 gen. later)
690	27.19	New Order ”P01_Rush”
840	23.64	New Order ”P01_Rush” (10 gen. later)
945	23.64	Maintenance change ”M01”
1095	22.89	Maintenance change ”M01” (10 gen. later)
1935	25.05	New Order ”P02_Rush”
2085	23.38	New Order ”P02_Rush” (10 gen. later)
3120	39.38	Maintenance on ”M03”
3270	36.19	Maintenance on ”M03” (10 gen. later)

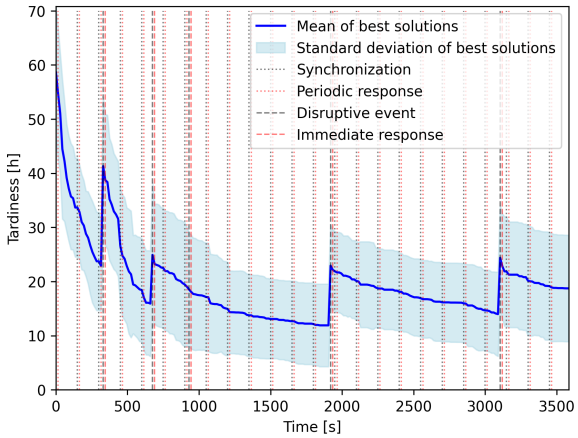

Fig. 8. The scheduling system synchronizes every 2.5 minutes and sends back a valid schedule 15 seconds after the update event. Five disturbances occur (see Table 1). Immediate responses are generated after 15 seconds.

REFERENCES

[1] I. Harjunkoski, C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, et al., ”Scope for industrial applications of production scheduling models and solution methods,” *Computers & Chemical Engineering*, vol. 62, pp. 161-193, Mar. 2014.

[2] C. Klanke, V. Yfantis, F. Corominas, S. Engell, ”Short-term scheduling of make-and-pack processes in the consumer goods industry using discrete-time and precedence-based MILP models,” *Computers & Chemical Engineering*, vol. 154, Nov. 2021.

[3] G. M. Kopanos, C. A. Méndez, L. Puigjaner, ”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*, vol. 207, pp. 644?655, Dec. 2010.

[4] D. Wu and M. G. Ierapetritou, ”Decomposition approaches for the efficient solution of short-term scheduling problems,” *Computers & Chemical Engineering*, vol. 27, no. 8-9. pp. 1261?1276, Sep. 2003.

[5] C. Klanke and S. Engell, ”Scheduling and batching with evolutionary algorithms in simulation-optimization of an industrial formulation plant,” *Computers & Industrial Engineering*, vol. 174, Dec. 2022.

[6] S. Amaran, N. V. Sahinidis, B. Sharda, S. J. Bury, ”Simulation optimization: a review of algorithms and applications,” *4OR*, vol. 12, no. 4, pp. 301-333, Nov. 2014.

[7] G. Tasoglu and G. Yildiz, ”Simulated annealing based simulation optimization method for solving integrated berth allocation and quay crane scheduling problems,” *Simulation Modelling Practice and Theory*, vol. 97, p. 101948, Dec. 2019.

[8] G. E. Vieira, J. W. Herrmann, E. Lin, ”Reschedulng Manufacturing Systems: A Framework of Strategies, Policies, and Methods,” *Journal of Scheduling*, vol. 6, pp. 39-62, 2003.

[9] F. Espinaco and G. P. Henning, ”Industrial Rescheduling Approaches: Where Are We and What is Missing?,” *Proceedings of the 11th International Conference on Production Research - Americas. ICPR 2022*, pp. 461-467, 2023.

[10] C. Schoppmeyer, *Reactive Scheduling Using Timed Automata Models and Integration with Sequential Control Logic*. Aachen: Shaker Verlag, 2015.

[11] S. Panek, S. Engell, S. Subbiah, O. Stursberg O., ”Scheduling of multi-product batch plants based upon timed automata models,” *Computers & Chemical Engineering*, vol. 32, no. 1-2, pp. 275-291, 2008.

[12] D. Gupta and C. T. Maravelias, ”A General State-Space Formulation for Online Scheduling,” *Processes*, vol. 5, no. 4, p. 69, Nov. 2017.

[13] C. Klanke, E. Pasieka, D. Bleidorn, C. Koslwoski, C. Sonntag, S. Engell, ”Evolutionary Algorithm-based Optimal Batch Production Scheduling,” *Computer Aided Chemical Engineering*, vol. 49, pp. 535-540, 2022.

[14] A. E. Eiben and J.E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Heidelberg: Springer Berlin, 2015.

[15] R. E. Franzoi, B. C. Menezes, J. D. Kelly, J. A. W. Gut, ”A moving horizon rescheduling framework for continuous nonlinear processes with disturbances,” *Chemical Engineering Research and Design*, vol. 174, pp. 276–293, Oct. 2021.