Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Evolutionary Algorithm-Based Real-Time Scheduling via Simulation-Optimization for Multiproduct Batch Plants

Engelbert Pasieka[a], Sebastian Engell[bs]*

[a] Provincial University, Department of Chemical Engineering, City, Province, Country
[b] State College, Engineering Department, City, State, Country
* Corresponding Author: author.email@someuniversity.ca.

ABSTRACT

Scheduling in the process industry determines the sequence and timing of operations to optimize objectives such as minimizing order tardiness and improving plant performance. Traditionally, scheduling is performed offline and infrequently, with production data manually transferred to the scheduling system. This delayed approach can lead to discrepancies between the plant's actual state and the scheduling model. For example, disturbances during this time may render the solution ineffective due to mismatches between the plant and its model. Other problems may arise in environments where frequent disturbances and fast responses are necessary, as the traditional scheduling approach often lacks the flexibility to adapt in real time. Because of this inertia, traditional methods often fail to respond quickly enough, making them unsuitable for dynamic production environments. To address this problem, real-time scheduling establishes a continuous exchange of information between the scheduling system and the production plant's control system. This ensures the model is regularly updated to reflect the current production state, generate accurate predictions, and facilitate fast decision-making. For effective real-time integration, the scheduling system must accurately model the plant's behavior and quickly generate solutions to prevent disturbances from propagating, which would otherwise lead to further discrepancies.

We present a simulation-optimization approach tailored for real-time requirements, combining an evolutionary algorithm with a discrete-event simulator. The simulation model is continuously updated with the latest plant data, enabling the optimizer to make well-informed decisions in real time. We validate our approach using a multiproduct, multistage batch plant in the pharmaceutical industry, demonstrating that it can generate high-quality solutions quickly through continuous synchronization and rapid disturbance responses. We further test the system by simulating disturbances, showing that with appropriate parameter tuning and parallelization, our approach achieves near-optimal solution quality. We compare our results with those of an idealized scheduler, demonstrating that our method performs competitively under time constraints and effectively manages unforeseen disturbances.

Keywords: Modelling and Simulation, Large Scale Desing, Planning/Scheduling,

INTRODUCTION

Production scheduling is a decision-making process that determines the sequence and allocation of order operations and resources in production processes, improves plant efficiency, and reduces operational costs. Scheduling is typically performed offline where it generates static production plans in advance which operators of the production plant use to execute operations on the shop floor.

Real-world production environments are subject to unforeseen events such as machine breakdowns, new order arrivals, or uncertain processing times and operation costs. Such events render offline schedules often impractical without further revisions or even infeasible without fully reschedules the current agenda.

Online scheduling systems produce production plans in real-time and maintain a continuous information exchange with the control center. It requires responsiveness to unforeseen events, modifiability to adapt to evolving constraints and objectives, and accuracy in its predictions to continual align the scheduling system with the plant state and to minimize the impact of disruptions. Many existing approaches fall short to deliver timely responses and to generate schedules that are interpretable and executable by plant operators [1].

Online and offline scheduling complement each other in which offline scheduling systems provide the plans for a period of time which online scheduling dynamically adapts if uncertainties are present or disruptive events occur. In a feedback control system, offline scheduling is analogous to providing the reference trajectory or setpoint, e.g., it defines the desired outcome or goal over a period of time. The online scheduler acts as the controller that continuously monitors the current state and applies corrective actions to keep the system on track toward the reference trajectory [2].

Simulation-optimization (SO) is well suited for this role by using high-fidelity simulation models to represent the production process in detail and generates production plans, the plant operators can follow without much modification. Simulation models can automatically adapt to changing plant states and production plans, e.g., which orders to produce first, and are simple to modify when the plant structure and parameters of the process change, e.g., resource availability and operation uncertainties [3].

SO methods with metaheuristic are inherently inefficient due to the sample inefficiency of the algorithm and the computational burden of the simulation (Performance and Limitations of Metaheuristics 2018). Recent advances in general video game playing (GVGA) have shown that evolutionary algorithms (EA)si with simulation models compete well in environments, where changes of the decision space are frequent and fast responses are crucial [4].

We present a tailored evolutionary algorithm for reactive-real time scheduling and show that our approach can be used for real-time scheduling systems. We demonstrate our approach on a complex pharmaceutical batch production case study and simulate a typical production shift of 8 hours with major disruptions.

APPROACH

Our online scheduling framework uses simulation-optimization with real-time feedback from the production process to refine decisions and update the simulation model to align with the state of the production. Figure 1 shows the components and interactions of our online scheduling framework.

The framework is designed to operate in real time, ensuring that scheduling decisions remain both feasible and efficient as production conditions evolve. It consists of two tightly interconnected components: the Scheduling System and the Production System.

Scheduling System

The scheduling system generates scheduling solutions, represented as genotypes. These genotypes encode both the sequence of orders , e.g. $\pi_{seq} = \{P1, P2, P3, \dots\}$,and the allocation of orders to resources, e.g., $\pi_{alloc} = \{P1{:}\{M1\}, P2{:}\{M1, M2\}, \dots\}$. The EA systematically explores and optimizes schedules through selection, recombination, and mutation. To evaluate the quality of the schedules, the system employs a discrete-event simulation that decodes genotypes and simulates their execution under realistic production conditions.

The EA operates iteratively, starting with the modification and initialization of genotypes. When necessary, new degrees of freedom are injected, such as adding new tasks or adjusting allocations, to address changing production requirements. At each iteration, the algorithm evaluates the fitness of each solution, selects the most promising candidates, and applies recombination and mutation to generate new solutions. A survivor selection process ensures that only the best-performing solutions are carried forward. This iterative process continues until a stopping criterion is met.

The scheduling system adapts in real time, e.g., periodic and reactive triggers from the production system prompt updates to the algorithm, and ensures that the schedules remain aligned with the current state of the production plant.

Production System

The production system provides real-time feedback to the scheduling system. It comprises a control center and the production plant itself. The control center initiates and parametrizes the evolutionary algorithm and monitors real-time data from the production plant. Updates to the scheduling process are triggered either periodically, such as when delays occur, or reactively in response to unforeseen events, such as rush orders or maintenance requirements. The production plant executes the schedules generated by the algorithm and provides detailed operational data, including task progress and resource availability. The EA ensures that the schedules are efficient, while the real-time feedback mechanism guarantees that the decisions remain feasible and responsive to the current state of the production system. This synergy enables the framework to address dynamic and complex scheduling problems in real-world manufacturing environments effectively.

EXPERIMENTS

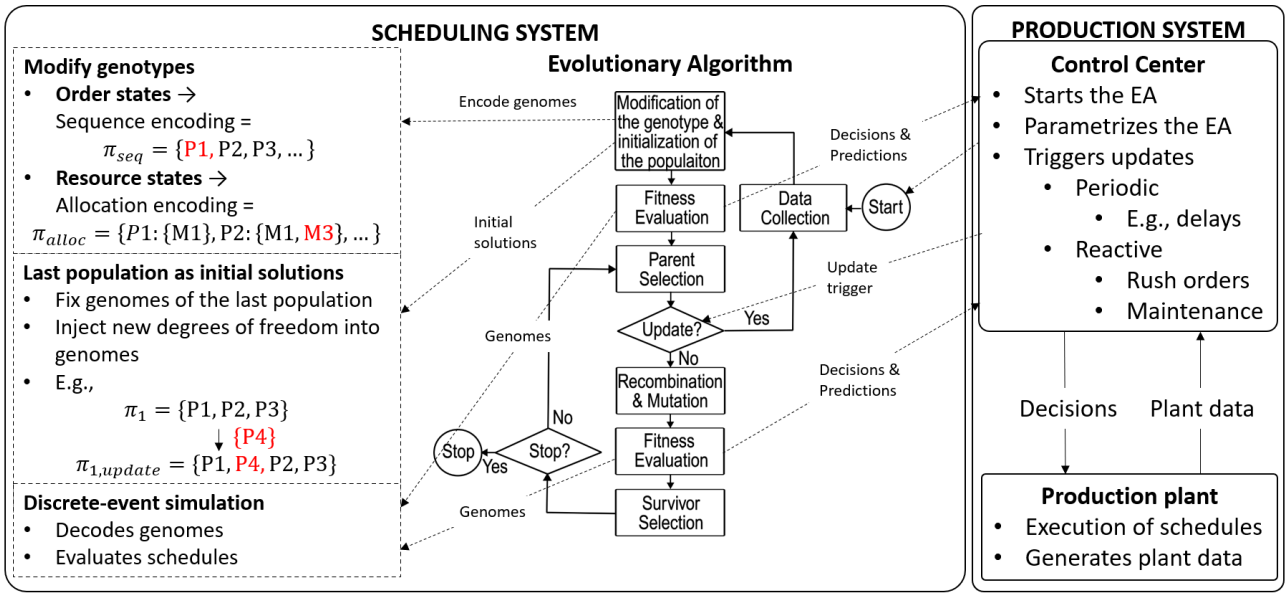

Figure 1: The reactive real-time scheduling framework integrates a scheduling system based on an evolutionary algorithm with a production system to dynamically adapt to disruptions in a multiproduct batch plant. The scheduling system modifies genotypes representing order sequences and resource allocations and uses a discrete-event simulation to evaluate schedules. The evolutionary algorithm iteratively updates and refines schedules based on real-time feedback from the production system. The production system provides data from the plant and triggers updates in response to events such as delays, rush orders, and maintenance tasks, ensuring continuous alignment between schedules and the current plant state.

We conducted a series of experiments to evaluate the performance of our proposed real-time scheduling system. The experiments were designed to simulate a realistic production shift in a multiproduct batch plant and to test the system's ability to handle various disruptive events efficiently. In this section, we describe the fixed parameters used across all experiments, the variable parameters that were adjusted to test different configurations, and the scenarios that were considered. Additionally, we compare the results of our experiments to a clairvoyant scheduler. This scheduler has perfect knowledge of future events and, therefore, inherently performs better. For each section between events, we define a constant clairvoyant case to compare how close the experimental runs are to the perfect scheduler.

The case study addressed in this work is taken from Kopanos, et al., (2010). It is a multiproduct batch plant with 17 units (machines) that are organized in 6 stages (Figure 2). The problem, which is a variant of a hybrid flow shop problem, comprises 12 instances that vary in the number of orders, the objectives, and in the storage policy. The features of the problem include limited product-unit flexibility, machine-dependent processing times, sequence-dependent changeover times, and product-specific recipes, meaning that certain jobs are not processed on some of the available stages.

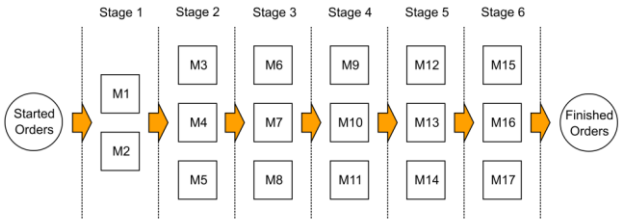

Figure 2: Plant layout of the case study.

In total, 54 experimental scenarios were evaluated, with each scenario being repeated 20 times to account for the variability of the evolutionary algorithm. These scenarios allowed us to test the scheduling system's performance across a wide range of conditions, ensuring robustness and reliability in dynamic production environments.

Fixed Parameters

The following parameters remained consistent across all experiments to ensure comparability:

Simulation Time Span: Each simulation consists of 8 hours (28,800 seconds) of production, representing a typical production shift.

Initial Solution: The simulation was initialized with a high-quality scheduling solution. The initial solution has 30 orders with an initial objective value of -1.34 (all orders

finish early). The total time to complete the production schedule is 30 hours and 23 minutes.

Disruptive Events: Table 1 lists the eight disruptive events that were introduced during each simulation to mimic real-world production disturbances.

Table 1: Disruptive events and their arrival times.

Event	Time (sec)	Info
Release of P01_Rush	2400	Due in 12h
Release of P03_Rush	4200	Due in 12h
Maintenance of M1	4800	1h duration
Maintenance of M3	7200	1h duration
Maintenance of M19	12000	1h duration
Release of P05_Rush	13800	Due in 12h
Maintenance of M13	18000	1h duration
Maintenance of M16	21600	0.5h duration

All maintenance orders were non-preemptive, meaning they were only executed after the current operation on the respective machine had completed.

Population Size: Each EA run consisted of 32 individuals, which were evaluated in parallel using 32 threads on a machine with 16 cores (Intel Core i9-14900K). The offspring size was set to 32, with an elitist size of 16.

Mutation Operator: We used a permutation mutation, which selects a subsection of the gene and shuffles it randomly. The mutation rate was set to 1, meaning every individual underwent mutation.

Crossover Operator: The cycle crossover operator was applied, leveraging the positional information from two parent individuals. The crossover rate was set to 1.

Parent Selection: Rank-based selection was used to choose parent individuals for the next generation.

Survivor Selection: Random survivor selection was employed, with an elitist fraction of 50%. The top 50% of individuals from the previous generation were retained, while the remaining individuals were selected randomly from the remaining population.

Objective Function: The objective was the minimization of tardiness (Equation (1)). If the tardiness was below 0, the objective shifted to minimizing the largest earliness (Equation (2)). This leads to a smooth transition of the objective into the negative range pushes the orders further away from their due dates.

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

$$L = \sum_{i=1}^{N_{Orders}} \min\max\left(t_{i,due} - t_{i,end}\right) \quad (2)$$

Decision Variables of the EA: The sequence of orders, e.g., $\pi_{seq} = \{P1, P2, P3, \dots\}$, is the decision variable of the EA. The allocation of orders to machines was solved by the simulator with a greedy FIFO rule. The first operation to arrive on a machine is scheduled on the machine if and only if the operation would not otherwise finish earlier on other machines where it has eligibility.

Variable Parameters

To evaluate the adaptability and performance of the scheduling system, we varied the following parameters:

Update Interval: The interval at which the scheduling system received data from the production system and sent updated decisions back to the plant. The following intervals were tested: 60 seconds, 120 seconds, 300 seconds, 600 seconds, 1,200 seconds, and 2,400 seconds.

Seconds per Generation: This parameter represented the computational load of the parallelized simulations, determining how long each iteration of the evolutionary algorithm took. We tested three different values: 3 seconds, 6 seconds, and 12 seconds. For example, with an update interval of 60 seconds and 3 seconds per generation, the algorithm would complete 20 iterations per update.

Continuity: This parameter controlled how the population was initialized after a disruptive event (e.g., a rush order or maintenance event):

- **Best:** The population was initialized by cloning the last best individual, with random modifications applied to the clones.
- **Default:** The population retained its state from before the disruptive event, with random modifications applied to the existing genotypes.
- **Random:** The population was completely reinitialized randomly, with modifications applied to the new genotypes.

RESULTS

In this section, we present the results of our experiments, and compare the performance of our proposed real-time scheduling system across various configurations and scenarios. The results are analyzed in terms of correlation analysis, convergence speed, and adaptability to disruptions. Additionally, we compare the performance of our system to that of a clairvoyant scheduler to assess how close the experimental runs are to the optimal solution.

Correlation Analysis

We conducted a correlation analysis to explore the relationships between key variable parameters (Seconds per Generation, Update Interval, Continuity strategies) and performance metrics (Mean Fitness and Standard Deviation) at important time points during the simulation. These time points include the start of the simulation, the start and end of disruptive events, and the end of the simulation.

The correlation analysis in Table 2 highlights key relationships between variable parameters and

Table 2: Correlation between variable parameters and performance metrics at distinct time points during the simulation. The "All events" column represents the overall correlation calculated across all distinct time points during the simulation, including the start of the simulation, before and after each disruptive event, and at the end of the simulation. The other columns show the correlations at specific key moments in time.

Metric	All events	Before events	After events	After orders	End
Sec. per Gen.	0.021	0.103	0.012	0.003	0.393
Update Interval	0.198	0.064	0.309	0.569	0.249
Continuity Best	-0.099	-0.199	-0.122	-0.187	-0.048
Continuity Default	-0.122	-0.215	-0.156	-0.249	-0.277
Continuity Random	0.221	0.414	0.278	0.436	0.326

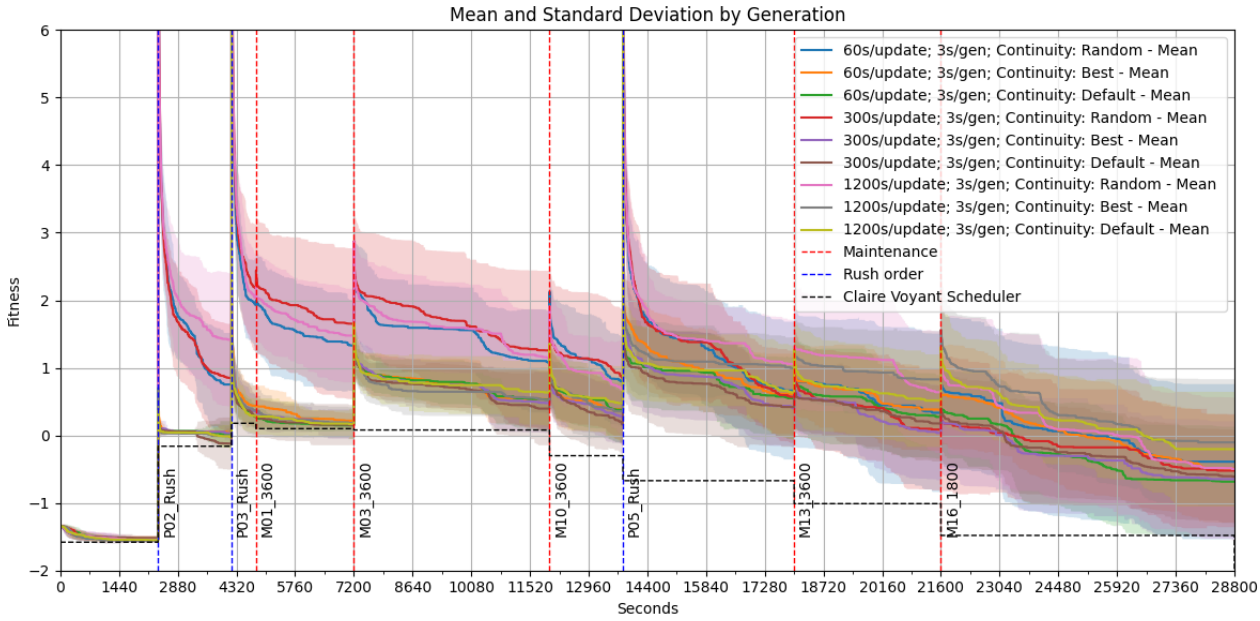

Figure 3: Mean and Standard Deviation by Generation for Different Configurations. The plot highlights the impact of update intervals, continuity strategies, and disruptive events on scheduling performance over the course of the simulation. The best value achieved by the clairvoyant scheduler in each stage is taken as a reference point. Stages are defined as the periods between two consecutive events, such as from the start of the simulation to the first rush order (P02_Rush).

performance metrics at specific time points during the simulation. Before disruptive events, longer Seconds per Generation correlates positively with better mean performance (0.103). After events, the Update Interval shows a strong positive correlation (0.309), indicating that frequent updates reduce the fitness. This effect is most pronounced after rush orders, where Update Interval shows a correlation of 0.569 with mean performance, emphasizing the importance of quick feedback loops in adapting to sudden changes. At the end of the simulation, Seconds per Generation again correlates strongly with mean performance (0.393), suggesting that shorter computation times per generation contribute to better performance.

Simulation Runs

Figure 3 shows the evolution of mean fitness values and their standard deviations across different configurations over time, highlighting the impact of update intervals, continuity strategies, and disruptive events on scheduling performance. The different lines in the plot represent various combinations of update intervals, continuity strategies, and seconds per generation, while the shaded areas indicate the standard deviation. Additionally, the performance of a clairvoyant scheduler is included as a reference for comparison.

The results demonstrate that the Continuity strategy Best consistently outperforms other approaches in terms of stability and convergence speed, which aligns with the findings from the correlation analysis in Table 2. Disruptions caused by new order arrivals, such as rush orders, tend to impact the system more significantly than

maintenance orders. This is evident from the larger spikes in fitness values following rush orders compared to maintenance events. Despite the initial disruptions, given sufficient time, all configurations converge to nearly the same mean fitness values by the end of the simulation.

Shorter update intervals lead to faster recovery from disruptions, as seen in the quicker drops in fitness values after rush orders. The Random continuity strategy introduces more variability in the results, as indicated by the wider shaded areas in the plot. This increased variability can occasionally help the system find better solutions in specific stages, but it comes at the cost of stability. On the other hand, frequent updates, combined with longer computational times per generation, result in more stable and reliable performance over time, reinforcing the importance of these parameter settings.

While the scheduling system performs competitively, it remains slightly behind the clairvoyant baseline in all stages and none of the tested configurations consistently outperform the clairvoyant scheduler across all stages of the simulation.

DISCUSSION

The experimental results demonstrate the strengths and limitations of the proposed real-time scheduling system. The system shows promise for managing dynamic production environments, particularly in adapting to disruptions such as rush orders and maintenance events. However, further research is needed to explore its broader applicability and address remaining challenges.

A key strength of the framework is its responsiveness to sudden disruptions. Frequent update intervals allow quick recovery, minimizing disruption impacts. Our analysis shows that reducing update time is more impactful than minimizing computation time per generation, ensuring the system stays aligned with real-time data.

The comparison with a clairvoyant scheduler serves as a valuable benchmark. While the clairvoyant scheduler outperforms our approach, the performance gap remains small, suggesting near-optimal results under realistic constraints. However, none of the tested configurations surpassed the clairvoyant baseline, indicating room for improving the evolutionary algorithm's balance between exploration and exploitation.

The analysis of continuity strategies provides additional insights. The Best strategy delivers more stable and reliable performance compared to Random and Default strategies. Building upon the best previous solutions enhances stability, whereas introducing variability through random reinitialization can lead to less predictable results. Balancing exploration and exploitation remains critical for improving robustness.

Figure 3 underscores the importance of tailoring the scheduling system's parameters based on the expected types of disruptions. Frequent updates and careful selection of continuity strategies can significantly enhance the system's adaptability and performance, especially in dynamic production environments. The trade-off between stability and exploration must be carefully managed to achieve optimal scheduling outcomes across different stages of the simulation.

Despite promising results, the study is limited to one reference case, raising questions about generalizability. Future research should evaluate whether the framework adapts across different problem instances with varying initial conditions. Additionally, the influence of initial solutions on system performance remains unclear, warranting further exploration.

In conclusion, the proposed framework shows considerable potential for improving production efficiency in dynamic environments. Its adaptability to unforeseen events, combined with continuity strategies and high-fidelity simulations, ensures competitive performance. Future research should focus on expanding case studies, optimizing algorithm efficiency, and exploring adaptive mechanisms for handling frequent disruptions more effectively.

ACKNOWLEDGEMENTS

Please acknowledge your funders in this section.

REFERENCES

1. Espinaco, F., Henning, P., Industrial Rescheduling Approaches: Where Are We and What is Missing? *Production Research – Americas, pp .461-467* (2023).
2. Engell, S., Uncertainty, decomposition and feedback in batch production scheduling *ESCAPE19* (2009).
3. Klanke, C., Engell, S., Scheduling and batching with evolutionary algorithms in simulation–optimization of an industrial formulation plant *Computers & Industrial Engineering* (2022).
4. Gaina, R.D., Devlin, S., Lucas, S. M., Perez-Liebana, D., Rolling Horizon Evolutionary Algorithms for General Videa Game Playing. *IEEE TRANSACTIONS ON GAMES* (2020).