Systems & Control Transactions

Original Research Article
Peer Reviewed Conference Proceeding
ESCAPE 36 - European Symposium on Computer Aided Process Engineering
Sheffield, UK. 21-24 June 2026

PSE PRESS

Simulation-Optimization vs. MILP Approaches for Real-Time Scheduling of Multiproduct Batch Plants

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

[a] INOSIM Software GmbH, Joseph-von-Fraunhofer-Str. 20, 44227 Dortmund, Germany
[b] TU Dortmund, Department of Biochemical and Chemical Engineering, Emil-Figge-Str. 70, 44221 Dortmund, Germany
[c] ZEDO e.V., Joseph-von-Fraunhofer-Str. 20, 44227 Dortmund, Germany
* Corresponding Author: engelbert.pasieka@inosim.com.

ABSTRACT

Production scheduling in the process industry is often treated as a static optimization problem, although real plants require frequent rescheduling due to disturbances such as rush orders, equipment breakdowns, and changes in processing times. This paper compares a simulation-optimization approach that couples a discrete-event simulator with an evolutionary algorithm (EA) with a sequence-based mixed-integer linear programming (MILP) formulation for real-time scheduling of multistage batch systems. Both methods are embedded in an event-driven rolling-horizon framework under strict computation time limits.In static experiments for a 3-stage, 2-machine flow-shop setting (10 products, 20 orders, random processing times), the EA achieved lower makespans across all tested time budgets, improving results by about 7–13% on average compared to the MILP approach. In real-time experiments (40 initial orders, maintenance, three rush orders, 10 s and 60 s periodic updates), the solution quality of the MILP approach was lower after disturbances under restricted computation times. Each experiment was repeated 25 times with identical randomly generated processing times across methods per run; in 4% of all runs no solution was obtained within the available response time. Including explicit allocation decisions for the EA increases flexibility but can reduce short-term responsiveness within the rolling-horizon setting.Overall, the results indicate that MILP can be competitive in stable scenarios when sufficient solver time is available, whereas simulation-optimization is better suited to reactive scheduling in real-time settings where rapid schedule adaptation is critical.

Keywords: Simulation, Scheduling, Rolling Horizon Optimization

INTRODUCTION

Production scheduling is used in the process industry to improve resource utilization and to increase operational efficiency in dynamic situations. In the scientific literature, scheduling problems are often addressed as static optimization problems, where a fixed set of orders, resources, and processing times is assumed, and the problem is solved in advance for a given state of the plant, or for set of predefined scenarios to evaluate the performance of the plant. In these cases MILP is effective in computing high-quality or even optimal schedules if enough time is given or if the problem is decomposed.

In industrial practice, production environments encounter frequent and unforeseen disturbances, such as arrivals of rush orders, machine breakdowns, maintenance activities, and variations in processing times. These events render precomputed schedules suboptimal or even infeasible, and they require adaptations of the schedule of operations, often under strict time constraints, to regain feasibility and to reduce losses in plant efficiency. In reality, production scheduling systems must frequently react to changes in the state of the production system rather than solve a one-time static optimization task [1]. This setting is commonly referred to as real-time or online scheduling.

Some research effort has been devoted to extend MILP formulations to online or rolling-horizon scheduling frameworks [2, 3]. Nevertheless, the performance of MILP in real-time environments is hampered by the need to frequently re-start the optimization with limited computation time in reaction to disturbances and structural

changes, such as new order arrivals or unavailability of resources, modify the objective or the feasible region and preclude a continuation of the branch-and-bound search, ruling out a reuse of the previously explored search tree [4].

Simulation-optimization provides an alternative paradigm for real-time scheduling. By coupling a detailed discrete-event simulation model with a metaheuristic optimization algorithm, it can incorporate complex plant constraints and adapt to changing system states [5]. EAs coupled to schedule builders generate feasible and executable schedules quickly and retain the state of the search across updates, which is advantageous in reactive environments [6] and can be used for high frequency scheduling tasks [7].

In this paper, we compare a standard MILP approach with a simulation-optimization approach based on an EA [6] across problem instances of increasing complexity. The available computation time is varied to reflect the requirement of real-time scheduling that feasible and high-quality solutions must be generated within strict time limits, even for complex problem instances. Static scheduling experiments are used to analyze the performance of both methods in the non-reactive case, while real-time scheduling experiments are used to assess their responsiveness and robustness in the presence of disturbances.

The MILP-based approach uses a sequence-based formulation, because sequence-based models are among the most compact and widely used formulations for scheduling problems. This formulation avoids the use of a discrete time grid and eliminates time discretization errors which facilitates initialization from an existing system state in online or real-time settings. Compared to time-indexed formulations, sequence-based models require fewer decision variables and constraints, which can lead to reduced computational effort and improved scalability.

SCHEDULING APPROACHES

We consider a multistage batch scheduling problem with a set of orders that have to be processed sequentially through multiple production stages. Each stage consists of one or more parallel units, and each order is assigned to one unit at every stage. Units process at most one order at a time, and preemption is not allowed. Processing times differ between units, and all orders are subject to release-time constraints.

Sequence-Based MILP Formulation

The MILP scheduling approach is a sequence-based formulation for multistage batch plants as in [8]. We set M to an upper bound on the scheduling horizon, e.g., $M = \sum_i \max_j \tau_{ij}$.

Sets and Indices

- $i, i' \in I$: orders
- $k \in K$: processing stages
- $j \in \mathrm{J}$: units

Decision Variables

- $x_{ij} \in \{0,1\}$: equals 1 if i is assigned to j
- $y_{ii'j} \in \{0,1\}$: equals 1 if i precedes i' on j
- $s_{ij} \geq 0$: start time of order i on j
- $s_{ik} \geq 0$: start time of i on k
- $ms \geq 0$: makespan

Parameters

- τ_{ij}: Processing time of order i on unit j

Objective Function

The objective is to minimize the makespan which is the completion time of the last processed operation across all orders (completion at final stage):

$$MS \geq s_{ik_{last}} + \sum_{j \in J_{ik_{last}}} \tau_{ij} x_{ij} \ \forall i \qquad (1)$$

Constraints

Assignment constraints

Each order must be assigned to one unit at each stage:

$$\sum_{j \in J_{ik}} x_{ij} = 1 \ \forall \ \mathrm{i}, \mathrm{k} \qquad (2)$$

Sequencing consistency constraints

One order must precede the other on a unit:

$$y_{ii'j} + y_{i'ij} \geq x_{ij} + x_{i'j} - 1 \ \forall \ i, i', j, k \ with \ i < i' \qquad (3)$$

Start Time per Stage

$$s_{ik} = \sum_{j \in J_{ik}} s_{ij} \ \forall \ i, k \qquad (4)$$

Activation of start time

$$s_{ij} \leq M x_{ij} \ \forall \ i, j \qquad (5)$$

Disjunctive (non-overlap) constraints

Orders assigned to the same unit cannot overlap in time.

$$s_{ij} + \tau_{ij} \leq s_{i'j} + M\left(1 - y_{ii'j}\right) \forall \ i, i', j, k \qquad (6)$$

Stage precedence constraints

Each order must complete processing at stage k:

$$s_{ik} + \sum_{j \in J_{ik}} \tau_{ij} x_{ij} \leq s_{ik+1} \ \forall \ i, k \qquad (7)$$

Release-time constraints

Orders cannot start before their release times:

$$s_{ik} \geq \rho_i \ \forall i, k \qquad (8)$$

Real-Time Scheduling Loop for the MILP Approach

The MILP formulation is embedded in an event-driven rolling-horizon framework. Scheduling updates are triggered periodically as well as by disruptive events such as rush orders.

At each update event, the start time of the scheduling horizon is aligned with the current state of the production system. Orders that have completed or are currently being processed are fixed by constraining their assignment X_{ij} and start-times S_{ij}. Newly released orders are included by updating the order set and the corresponding constraints. Lower bounds on start times are imposed to ensure consistency with executed operations and release dates.

When a maintenance event occurs on a machine, an additional disjunctive time constraint is introduced dynamically to ensure that no operation is scheduled within the maintenance window. For each operation o that can be assigned to machine j, a binary decision variable $z_{oj}(o,j)$ is created together with two bounding equations. The lower and upper constraints are formulated as

$$s_{oj} + \tau_{oj} \leq \mathrm{t}_j^{\mathrm{maint,start}} + M \cdot (z_{oj} + (1 - \mathrm{x_{oj}}))$$
$$s_{oj} \geq \mathrm{t}_j^{\mathrm{maint,end}} - M((1 - z_{oj}) + (1 - x_{oj}))$$

The constraint is conditioned on the assignment variable $x_{oj}(o,j)$ and is therefore enforced only if operation o is scheduled on machine j where $s_{oj}(o,j)$ denotes the start time of operation o on machine $j, \tau_{oj}(o,j)$ represents its processing time, and M is a sufficiently large constant. The interval $[\mathrm{t}_j^{\mathrm{maint,start}}, \mathrm{t}_j^{\mathrm{maint,end}}]$ defines the maintenance period on the respective unit j. By incorporating the processing time in the first constraint, the model ensures that an operation assigned before maintenance must also be completed before the maintenance window begins. If the operation is assigned to the affected machine, the binary variable activates a big-M disjunction that forces it to be scheduled either entirely before or entirely after the maintenance period. If $z_{oj}(o,j) = 0$, the completion time is restricted to occur before maintenance starts; if $z_{oj}(o,j) = 1$, the first inequality is relaxed, and the operation is forced to start after the maintenance period.

The MILP model is rebuilt at each update and solved within the available computation time. To improve performance, warm-start information from the previously computed solution is provided to the solver in the form of initial values for assignment variables and, optionally, for start times. However, due to structural changes in the model and the limited ability of MILP solvers to retain internal search state across re-optimizations, the solver effectively restarts after each disturbance.

The real-time MILP scheduling procedure can be summarized as follows:

1. Initialize the scheduling horizon based on the current plant state
2. Generate an event list consisting of periodic and event-driven updates
3. At each update event, fix the completed and the ongoing operations
4. Update orders, resource availability, and constraints
5. Initialize the solver using information from the previous solution
6. Solve the MILP within the available computation time and transfer the best available schedule to the production system
7. Repeat until the end of the simulation horizon.

Simulation-Optimization Approach

The simulation-optimization–based real-time scheduling approach used in this work is based on a previously published framework [6] that couples a detailed discrete-event simulator with an EA. The method operates as a continuously running optimization process that preserves the search state across updates and incrementally adapts schedules in response to changes in the production system.

This approach is used as a reference method for comparison with the MILP-based approach. The EA design, population management strategies, and simulation integration are described in detail in the original publication and are only briefly explained here.

The EA is embedded in a simulation–optimization real-time scheduling loop: schedules are encoded as genotypes (primarily a global order sequence π_{seq} and, possibly resource-allocation decisions π_{alloc}), and each genome is evaluated by executing it in a discrete-event simulator with a detailed model that represents the current state of the production system and acts as a schedule builder, using priority rules and enforcing logic or plant constraints (including repair mechanisms to keep the schedules executable). The EA runs continuously, iterating through selection, recombination, and mutation to gradually improve schedule quality on the current simulation model. At periodic update times and when disruptive events occur (e.g., rush orders, maintenance, delays), the production system feeds new state information to the scheduler, which updates the simulation model and reinitializes the population using strategies such as carrying over the full population, keeping the best individual, or random restart. The best available schedule can already be exported after one iteration and is then further improved as optimization continues.

CASE STUDY

The experimental study is designed to compare the performance of the sequence-based MILP approach and simulation-optimization based on an EA under strict computation time limits. Two types of experiments are conducted. First, static scheduling experiments are used to analyze the behavior of both approaches independent of the effect of real-time updates due to disturbances. Second, real-time scheduling experiments are performed to evaluate the performance under dynamic operating conditions with disruptive events. In all experiments, the fitness criterion is the minimization of the makespan.

To ensure a fair comparison, both approaches are implemented on a single CPU core, and the schedules generated by the EA are evaluated sequentially. All experiments were performed on a Ryzen 9800X3D CPU with 64GB of RAM. The MILP model was solved using the IBM ILOG CPLEX Optimizer 20.1.0.0 via the GAMS command-line interface.

Static Scheduling Experiments

The static experiments are performed for a multistage batch scheduling problems as described above. The production system consists of three stages, each equipped with two parallel units of different efficiency. A total of ten product types and twenty orders are considered in each experiment.

The processing times were generated randomly for each order–machine combination, with values uniformly distributed between 0 and 4 time units.

Objective and Constraints

Both the MILP and EA approaches solve identical scheduling problem All precedence constraints, machine capacity constraints, and order release constraints are enforced in both approaches.

Computation Time Limits

To reflect the requirements of real-time and near-real-time scheduling, the available computation time is strictly limited. Both methods are executed with computation time limits of 10, 15, 20, and 25 seconds. Each configuration is repeated ten times using different randomly chosen processing times.

The MILP solver runs until the specified time limit is reached or an optimal solution has been found. The EA runs continuously during the available time and improves the candidate schedules incrementally.

Evolutionary Algorithm Configuration

The EA uses a custom programmed compiled discrete-event simulator for schedule building and evaluation. In the static scheduling experiments, the decision variables of the EA comprise either only the sequencing of orders by a global priority parameter or in addition also the allocation of orders to machines. If the allocation is not controlled by the EA, the schedule builder assigns the machine with the earliest completion time. The EA parameter settings are taken from the prior study reported in [6] and remain fixed across all experiments.

Real-Time Scheduling Experiments

The real-time scheduling experiments are performed for the same multistage batch plant with three stages and two units per stage. Forty orders are released at the beginning of the simulation. All experiments follow an event-driven rolling-horizon scheme with a periodic update interval of either 10 seconds or 60 seconds. At each update event, rescheduling is performed based on the current state of the production system. The update interval also defines the maximum computation time available for solving the scheduling problem at each event. In addition, rush orders and maintenance occur within the production horizon, these are not known a priori.

For each update interval, three scheduling approaches are evaluated: the MILP formulation, the EA with allocation genome, and the EA without allocation genome. Each experiment is repeated 25 times with randomly generated processing times. The processing times for each order–unit combination are sampled randomly with a uniform distribution in the interval [1, 12] minutes.

The EA is implemented with the same settings as in the static experiments.

Disruptive Events and Update Policy

During execution in the real-time setting, three rush orders are introduced, representing unplanned disturbances that require immediate rescheduling. Rush orders become known at the 1st, 3rd, and 5th periodic update events.

In addition, a maintenance request is introduced temporarily disabling individual machines. A maintenance downtime of 500 seconds starting at simulation time 2000 seconds is requested for machine M01 at the second update event. At the fourth update event, a maintenance downtime of 500 seconds starting at simulation time 3000 seconds is requested for machine M04. Operations that are still running at the requested maintenance times are completed before the maintenance started but during the full maintenance interval no new operation can be started.

Experimental Variants

The EA-based simulation-optimization approach is evaluated in two configurations:

1. Using a genome encoding only of the sequence of orders, and
2. Using a genome encoding of both the sequence of orders and the allocation of orders to machines.

In the first configuration, machine allocation is

determined by a First-In-First-Out (FIFO) (earliest completion time) dispatching rule.

The MILP-based approach employs the sequence-based formulation described in Section 2 and is embedded in the same real-time scheduling framework. At each update event, the MILP model is rebuilt so that it reflects the current system state and then is solved within the available computation time.

RESULTS

In this section we present and discuss the results of the static and real-time scheduling experiments described in Section 3.

Table 1: Static scheduling results for a fixed problem structure under varying computation time limits. For each time limit, 10 instances with randomized processing times were solved using the MILP-based approach and the EA. The table reports the average relative difference in solution quality over all instances. The objective was the minimization of the makespan (see eq. 1).

Time limit (s)	EA better (count)	MILP better (count)	Mean relative difference (%)	Std. dev. of rel. diff. (%)
10	9 / 10	1 / 10	12.49	11.30
15	9 / 10	1 / 10	13.12	8.97
20	8 / 10	2 / 10	7.37	12.86
25	10 / 10	0 / 10	9.79	5.18

Table 2: Real-time scheduling results for update intervals of 10 seconds and 60 seconds. A Each experiment was repeated 25 times. In every run, the processing times were randomly generated but kept identical across the compared methods for the same run.

Update interval	Method	Mean makespan [h]	Std. dev.	Improvement vs. MILP %
10 s	MILP	4.72	0.57	-
	EA w. alloc.	2.24	0.22	52.4
	EA w.o. alloc.	2.06	0.21	56.2
60 s	MILP	2.94	0.50	-
	EA w. alloc.	2.17	0.24	26.1
	EA w.o. alloc.	1.96	0.26	33.5

Static Scheduling Results

Table 1 summarizes the results of the static scheduling experiments for a fixed problem structure under varying computation time limits for the minimization of the makespan. For each time limit, multiple problem instances were generated, and the table reports the resulting average solution quality obtained by each approach.

For all computation time limits, the EA achieves a better average solution quality than the MILP-based approach. As the available computation time increases, the performance gap between the two approaches does not decrease monotonically.

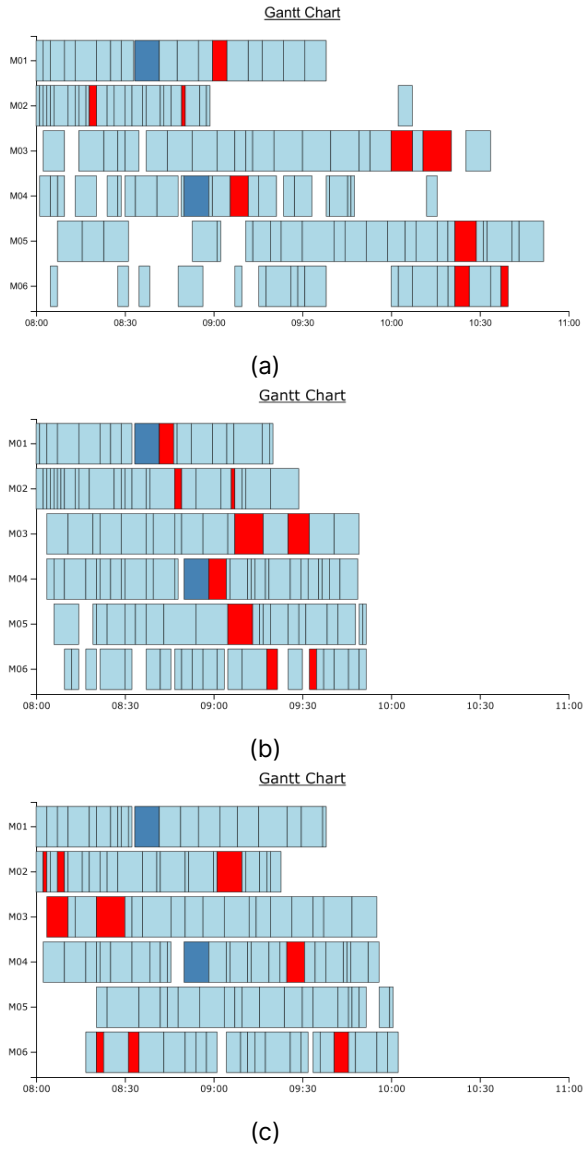

Figure 1: Gantt charts of the schedules generated by the MILP approach (a), EA with allocation genome (b), and EA without allocation genome (c) after 10 periodic updates with an update interval of 60 seconds. The rush orders are highlighted in red, and the maintenance downtimes are highlighted in dark blue.

Real-Time Scheduling Results

Table 2 presents the results of the real-time

scheduling experiments for the two different update intervals. The number of update cycles is fixed at 10 for all experiments. After the tenth update, no further rescheduling is performed, and the last computed schedule is simulated until the end of the production horizon. The table reports the average makespan values for the full simulation horizon. Runs in which the MILP solver did not return a feasible solution within the specified response time were excluded from the averaging procedure, as no valid schedule could be extracted in these cases. Across all experiments, this occurred in 4% of all runs.

The EA-based approaches achieved a better performance under real-time conditions with limited computation time per update.

When the update interval is increased to 60 seconds, the gap between the performance of the EA and of the MILP formulation becomes smaller. In 4% of all runs, the MILP approach did not provide a feasible solution within the available computation time. This occurred equally across both response-time settings.

For both response-times, the inclusion of allocation decisions in the EA genome led to slightly higher makespan values compared to the sequence-only encoding, so the higher flexibility of this formulation is outweighed by the increased size of the search-space.

The results are illustrated by the Gantt charts shown in Figures 1 a-c. All figures show results for the same simulation run, i.e., they are based on identical randomly generated processing times and the same disruptive events, including rush-order arrivals and maintenance downtimes.

The schedule generated by the MILP-based approach (Figure 1a) exhibits an uneven distribution of the workload across parallel units, showing idle periods on some machines. In contrast, the EA-based schedules (Figures 1b and 1c) show a more balanced allocation of processing tasks across the available machines.

CONCLUSIONS

This paper compared a sequence-based MILP approach with a simulation-optimization approach based on an EA for scheduling of multiproduct batch plants under strict computation time limits. The comparison was performed using static experiments as well as real-time scheduling scenarios with periodic updates and disruptive events.

The static results show that the EA achieves competitive or superior makespan values under tight time budgets.

In real-time scheduling, the MILP-based approach also exhibits a lower solution quality under restricted computation times. Increasing the available solver time improves the performance of the MILP formulation, however, the EA variants show a consistent advantage across both response-time settings. In a small number of cases (4% of all runs across all experiments), no feasible solutions were obtained by the MILP formulation within the available computation time. This is an additional weakness of the MILP approach which is not reflected in the quantitative comparisons where these runs were ignored. These infeasibilities can be related to the limited ability of the MILP solver to retain internal search state information across successive re-optimizations.

In contrast, the EA preserves information on the search state between updates and adapts the schedules incrementally. The EA only handles few degrees of freedom, the remaining ones are handled by a rule-based schedule builder. While theoretically including allocation decisions should provide better solutions than handling the allocations by the rule-based schedule builder, the increase of the size of the search space outweighs this advantage for short computation times. The use of a schedule builder also increases the robustness of the optimization, at the price of a loss of optimality. Overall, the results indicate that simulation-optimization is a competitive approach for real-time scheduling when fast schedule adaptations are required.

Future work will extend the analysis to different problem settings and will investigate hybrid strategies combining MILP-derived baseline schedules with simulation-optimization for real-time adaptation.

REFERENCES

1. Espinaco F, Henning GP. Industrial rescheduling approaches: where are we and what is missing?. Proceedings of the 11th International Conference on Production Research – Americas :461-467 (2023). https://doi.org/10.1007/978-3-031-36121-0_58
2. Subramanian K, Maravelias CT, Rawlings JB. A state-space model for chemical production scheduling. Computers & Chemical Engineering 47:97-110 (2012). https://doi.org/10.1016/j.compchemeng.2012.06.025
3. Chu Y, You F. Rolling horizon optimization for process scheduling under uncertainty. *AIChE J* 59:2110–2127 (2013) https://doi.org/10.1002/aic.13963
4. Gamrath G, Hiller B, Koch T, Pfetsch ME. Reoptimization techniques for mixed integer programming. *Math Program Comput* 8:1–29 (2016) https://doi.org/10.1007/s12532-015-0094-1
5. Klanke C, Engell S. Scheduling and batching with evolutionary algorithms in simulation–optimization of an industrial formulation plant. Computers & Industrial Engineering 174:108760 (2022). https://doi.org/10.1016/j.cie.2022.108760

6. Pasieka E, Engell S. Evolutionary algorithm based real-time scheduling via simulation-optimization for multiproduct batch plants. Systems and Control Transactions 4:894-899 (2025). https://doi.org/10.69997/sct.183349
7. Gaina RD, Devlin S, Lucas SM, Perez-Liebana D. Rolling horizon evolutionary algorithms for general video game playing. IEEE Trans. Games 14:232-242 (2022). https://doi.org/10.1109/tg.2021.3060282
8. Maravelias CT. Chemical production scheduling. Cambridge University Press (2021). https://doi.org/10.1017/9781316650998