# “Online Scheduling via Simulation-Optimization for Multiproduct Batch Plants”

Scheduling in the process industry determines the sequence and timing of operations to optimize objectives such as minimizing order tardiness and improving plant utilization. In research, scheduling problems are traditionally solved “batch-wise”, i.e. for an idle plant and a given set of orders, production recipes and due dates, optimal schedules are computed. However, this does not reflect reality of production planning and scheduling which is a continuous process, where new orders arrive periodically or at unknown instances, the real operations take longer or shorter periods of time than specified in the recipes, pieces or equipment break down, or operations cannot be executed as planned because resources are not available. All these aspects could be covered by infinitely fast re-computation of optimal schedules whenever an event happens or new information becomes available, but this is practically impossible for realistic problems due to the required computation time.

In online or real-time scheduling, a continuous exchange of information between the scheduling system and the control system of the production plant is necessary. The scheduling model must be updated frequently to reflect the current state of the production system and of the orders. The scheduling algorithm must react to events and disturbances fast, but also utilize the available computing power such that the schedule is near optimal.

We present an online iterative simulation-optimization approach which is tailored to handle these challenges. It builds on our previous work on simulation-optimization using evolutionary algorithms, as described in [1]. The evolutionary algorithm continuously searches for better schedules while the simulation model is updated with the latest information so that the evaluation of each generation of solutions reflects the current situation. After a pre-specified reaction time, a new solution is available after major disturbances. While the first operations of this solution are started, the schedule is further improved continuously and each assignment and timing of an operation that has not been started is based on the currently best solution.

We validate our approach using a multiproduct, multistage batch plant from the pharmaceutical industry, as in the work of Kopanos et al. [2], and demonstrate that it can generate high-quality solutions in the presence of new order arrivals and disturbances. The results are compared with those provided by an idealized clairvoyant scheduler which has access to the full information before the schedule is computed. The influence of the choice of the reaction time after a disturbance which involves a compromise between a fast reaction and better decisions in the immediate future is studied in detail.

## Introduction

* Industrial scheduling involves the allocation of the orders to resources and the sequencing and timing of operations
* The goal is to meet delivery dates and to improve the performance of the production system
* Traditional approaches use static or offline scheduling methods to find a good solution for the underlying problem
* If something unexpected happens like, e.g., failures or new order arrivals, the original schedule diverges from the current schedule which makes it unusable
* Therefore rescheduling after unexpected events that influence the decisions on the current schedule is necessary
* For rescheduling the state of the system needs to be fed into the problem solver
* This necessitates real-time data retrieval
* Even if iterative rescheduling is performed there would remain a major problem
* Scheduling problems are among the most difficult optimization problems and take fairly long to be solved
* Our approach is a real-time scheduling system that tries to overcome this hurdle by
  + Taking advantage of the unique solution approach of evolutionary algorithms
  + Using accurate simulation models to mimic the system as much as possible
* Even though evolutionary algorithms are among the less efficient solution methods for scheduling problems
* We take advantage of their behavior during the search
* Our approach goes as follows
  + The evolutionary algorithm is constantly solving the problem
  + When unexpected events occur the solver is initialized with the new state and the current best solutions of the solver
  + This merge ensures that new solutions are quickly found
* One drawback is that we cannot guarantee the optimality of the solution
* However, finding good solutions quickly is in our opinion of greater importance
* Therefore we accept the tradeoff between speed and solution quality happily

### Practical Implementation Details

* Discrete-event Simulation
  + A detailed representation of the production system
  + Only the start and end events of operations are important
  + They define if resources are occupied
* Recipes or routes for the products
  + Constraint the system
  + Which resources are necessary
  + Which steps are necessary
* Order list
  + Priorities
  + Due dates
  + Release dates
* Resources
  + Availabilty
* The Real-time scheduling system consist of two parts
  + The scheduling system
  + The production system
* Each component is communicates via data
  + Production system sends production state to the scheduling system
  + Scheduling systems sends predictions to the production system
* The scheduling system contains a simulation model of the production system and a problem solver
* The simulation model and the problem solver communicate with each other
  + The problem solver sends the current production system state to the model with a variety of decisions made by the problem solver
  + The simulation model initializes with the current production system state and evaluates the decisions
  + The problem solver evaluates the simulation results and finds new decisions
  + Once the decisions converge they are sent back to the production system

## Introduction (ChatGPT)

In industrial production environments, scheduling is a critical decision-making process that aims to allocate orders to resources and to determine the sequencing and timing of operations so that due dates are met and the utilization of valuable production assets is improved. In the process industries, batch plants typically operate under strict timing constraints and complex recipes that define the sequence of tasks and the required equipment. Established approaches to scheduling often focus on static or offline problem settings, where a given set of orders and resources is assumed to be available at a known initial time, and a single, optimal schedule is computed before execution starts. For large-scale, multiproduct batch plants, these static schedules are frequently computed via mathematical programming techniques or heuristics, and the effort invested ensures that the initial production plan is as efficient as possible.

However, real industrial operations are inherently dynamic. Shortly after execution starts, schedules become outdated due to unforeseen events, such as the arrival of new rush orders with tight due dates, unexpected deviations in processing times, temporary equipment failures that lead to unplanned maintenance activities, or resource unavailabilities caused by inventory shortages or operator constraints. These disruptions force the original schedule to deviate from the planned trajectory, often rendering it partially or entirely infeasible. As a result, reactive measures must be taken to reschedule the plant operations so that production stays on track with respect to strategic objectives, such as minimizing order tardiness or achieving a high throughput.

Dynamic, or online, scheduling methods have gained increasing attention as a means to address the complexity and uncertainty encountered in real-time plant operation. In online scheduling, the scheduling model and its input data are continuously updated as new information becomes available. This enables the plant to respond flexibly to disturbances and to incorporate fresh data on resource availability, order status, or equipment breakdowns into the decision-making process. While this approach promises better alignment with actual plant conditions, it also poses significant computational challenges. Scheduling problems belong to the class of combinatorial optimization problems and can take a substantial amount of time to solve even once. Continuous or frequent re-optimization as new events occur may not be feasible within the tight time constraints of real-world operations.

A promising strategy for real-time reactive scheduling is to integrate simulation models with a metaheuristic optimization procedure. Such simulation-optimization approaches allow for very detailed models of the production process that accurately represent operational constraints and resource interactions. A simulation model can efficiently evaluate candidate schedules proposed by an optimization algorithm, ensuring that the resulting solutions are executable and reflect the current state of the plant. The use of evolutionary algorithms (EAs) as the core optimization engine provides a flexible and robust search mechanism that can explore large solution spaces and adapt quickly to changes in problem structure. Although EAs typically do not guarantee optimality and may be computationally expensive, their inherent population-based and parallelizable search processes make them well suited for continuous improvement under dynamic conditions.

In this contribution, we present an online, iterative simulation-optimization approach that continuously updates a detailed simulation model of a multiproduct batch plant with the latest plant data, order arrivals, and information on disruptions. In parallel, an evolutionary algorithm searches for improved schedules, using the simulation model to evaluate each candidate solution. Our method reacts to events quickly: once a disruption occurs, the algorithm rapidly re-initializes the population of candidate solutions with the current plant state and best solutions found so far. This ensures that new, improved schedules can be identified within the available reaction time. Subsequently, while the initial operations of this updated schedule are being executed, the EA continues to refine the still-flexible parts of the schedule, improving solution quality in a continuous fashion.

We validate our approach using a case study of a multiproduct, multistage batch plant from the pharmaceutical industry, a reference problem introduced by Kopanos et al. [2]. We assess the algorithm’s capability to handle new order arrivals and operational disruptions and compare its performance to that of an idealized clairvoyant scheduler, which has complete knowledge of future events before computing the schedule. Moreover, we investigate the influence of the chosen reaction time after a disturbance and quantify the tradeoff between rapid responses and improved decision-making for the immediate future. Our study demonstrates that continuous simulation-optimization using evolutionary algorithms can provide high-quality, feasible schedules in near real-time and can effectively cope with the dynamics of modern manufacturing environments.

# Practical Implementation Details

Questions to be answered:

* How is the simulation set up and what parameters/data does it capture/needs from the real plant
* How does data from the control center of the plant feed into the real-time scheduling framework
  + In which frequencies does the data come in
  + How seamlessly does the information flow
* the degree of fidelity with which the simulation captures real-world constraints

The reactive real-time scheduling framework consists of a scheduling system and a production system. The production system includes the control center and the production plant. The production plant executes schedules which are based on decisions of the control center. It continuosly generates plant data which it sends back to the control center. The control center takes the plant data and starts the decision making process by starting the evolutionary algorithm of the scheduling system. It also parametrizes the evolutionary algorithm and triggers updates which are either periodic or reactive.

The scheduling system contains the evolutionary algorithm and the discrete-event simulator. Together they form the simulation-optimization framework. The first step of the EA is to modify the genotypes to account for the most recent data from the production plant sent by the control center. The genotypes are decision variables which the simulation can process. They can be of various kind like sequencing and timing of orders, allocation of resources, or splitting and merging of batches. In our case we included two genotypes. They are the sequence genomes, which represent the sequence in which orders are processed as well as the timing of operations in the simulation, and the resource state, which define the allocation of orders in different stages and the resource availability. Through the sequence and allocation genomes disruptions coming from the control center can be included in the decision making process. For example, if a new order arrives which has a high priority it can be included in the sequence genome and then further processed by the simulation. However, despite direct alterations of decisions some disruptive events may only subtly change the production plant. For example, maintenance or breakdowns of machines may be a temporary phenomenon which do not alter the decision space directly. To include them into the simulation model, despite the modified genotypes, the plant state is also handed into the scheduling system.

The plant state itself captures following information:

* Set of orders which are currently processed or need to be processed in the future as well as their relevant data like due dates, release dates, recipes, resources, etc.
* Set of resources which are currently or in the future available for the processing of orders as well as information about possible breakdowns, maintenance, and their timings.

The data is used to initialize the simulation model with the most recent state and to define the available genotypes. E.g., all orders that are currently running cannot be included in the genotype except if they have operations which lay in the future and which are changeable. However, it is possible to cancel orders at any time, which might free some resources. Orders which did not start yet can be fully integrated into the decision making process.

The frequency in which the plant data reaches the scheduling system depends on several factors. The data needs to be collected from different sources like the MES system, sensors, etc. All this data needs then to be formatted into a protocol that can be send around and used by the scheduling system. In our study we assume that data can be fetched infinitely fast from the production system. However, in reality this might be an assumption which is clearly not reasonable. But there are many ways to speed up the data fetching. Examples could be to fetch the data parallel to the execution of the simulation. So the least amount of time it would take for a periodical update would either be the time it takes to fetch the data or the time it takes to run a generation plus the time it takes to run a generation plus the time it takes to push the data back to the control center.

The main idea is that even though we send the latest decisions and predictions to the control center we still continue to run the simulation optimization with the latest data coming from the control center, even though the real state might have changed. We run the optimization with the currently available information until a new update trigger arrives from the control center. Then we fetch the data and update the genotypes of the EA and initialize the simulation model. Even though the EA did not run on the latest information the discrepancy might be only slightly and therefore it is highly likely that a good solution found with the last information might still be a good solution or at least a good starting point for the search of the EA: This way we warm up the search process of the EA. We will demonstrate that this approach leads to significant advances in the efficiency of the EA and shows that combined with this strategy the EA is a legitimate tool for real-time scheduling. On top of that we also show that a moving horizon approach were we change the environment constantly in a rolling horizon does not influence the performance of the EA negatively. In fact, due to the freezing of decisions which lie behind the current horizon the decision search space reduced and therefore the complexity of the problem reduces as well. This leads to faster convergences of the EA. It is generally questionable if a rolling horizon approach for large decision spaces could not compete with an open horizon approach.

### Version of ChatGPT

The proposed reactive real-time scheduling framework comprises two principal components: a scheduling system and a production system. The production system includes both the control center and the production plant. Within this configuration, the production plant executes schedules determined by the control center’s decisions. Concurrently, it continuously generates plant data, which it transmits back to the control center. The control center, upon receiving this data, initiates the decision-making process by activating the evolutionary algorithm (EA) within the scheduling system. Furthermore, it parameterizes the EA and triggers updates at either periodic intervals or in response to disruptive events.

The scheduling system contains both the evolutionary algorithm and a discrete-event simulator, together forming a simulation-optimization framework. The EA’s initial step involves adapting its genotypes—representations of decision variables—to incorporate the most recent plant data provided by the control center. These genotypes can represent various decision-making aspects, including sequencing and timing of orders, resource allocation, or batch splitting and merging. In this work, we focus on two specific genotypes: (1) the sequence genomes, which define the order in which production tasks are executed and their timing, and (2) the resource states, which reflect the allocation of orders across different stages as well as resource availability. By updating these genotypes in accordance with incoming data, the framework can integrate both direct and subtle disruptions originating from the control center. For instance, a newly introduced high-priority order can be incorporated directly into the sequence genome, while other disruptions, such as temporary machine maintenance or breakdowns, may indirectly affect the system without fundamentally changing the decision space. In such cases, the plant state is also communicated to the scheduling system to ensure that the simulator accurately reflects the current production environment.

The plant state encompasses information on (a) the set of ongoing and forthcoming orders, along with their associated parameters (e.g., due dates, release dates, recipes, resource requirements), and (b) the set of resources currently or potentially available, including data on downtime, maintenance, and timing of resource unavailability. This information serves to initialize the simulation model and to define the feasible genotypes. For example, ongoing operations cannot be retrospectively altered in the genotype unless future operations remain unaffected and modifiable. Conversely, orders not yet started can be fully integrated into the decision-making process. It is also possible to cancel orders at any point, thereby releasing resources and altering the system’s future state.

The frequency at which plant data is transmitted to the scheduling system is influenced by various practical factors, such as the speed of data retrieval from manufacturing execution systems (MES), sensors, and other data sources, as well as the required data formatting and transfer protocols. For analytical convenience, our study assumes that data retrieval occurs with negligible delay, although this assumption may not hold in real-world settings. In practice, parallelized data fetching or other optimization strategies could be employed to minimize retrieval time. Ultimately, the minimum update interval could be bounded by the data fetching duration, the execution time of one EA generation, and the subsequent data transfer back to the control center.

Conceptually, even after communicating updated decisions and predictions to the control center, the scheduling system continues to operate on the latest known information until a new update is triggered. When an update arrives, the data are fetched, the EA genotypes and the simulation model are reinitialized, and the optimization proceeds with more current information. Although the EA may not have been running on the most recent data during the interim, the discrepancy is often minor. As a result, solutions derived from the previously available information frequently provide strong starting points for subsequent optimization runs. This “warm start” approach can significantly enhance the EA’s overall efficiency, thereby reinforcing its suitability for real-time scheduling applications.

Additionally, we demonstrate that employing a moving horizon approach—continuously adjusting the environment within a rolling time frame—does not adversely affect the EA’s performance. In fact, due to the freezing of completed decisions behind the current horizon, the overall decision space is effectively reduced, decreasing the complexity of the optimization problem and thus accelerating the convergence of the EA. These findings suggest that, for large decision spaces, a rolling horizon strategy may be at least as competitive as an open horizon approach.

### Comments (ChatGPT)

**Key Points from the Passage:**

* **Overall Methodology:**
  + Uses a simulation-optimization framework, integrating a high-fidelity discrete-event simulation with a tailored evolutionary algorithm (EA) for dynamic real-time scheduling.
  + The simulation provides a detailed representation of the production process, including resources, operations, and implicit tasks not directly modeled in the scheduling problem.
  + The EA optimizes key decision variables (e.g., assignment of operations to units, global order priority) while the simulation handles the execution details and heuristic rules.
* **Dynamic Updating and Real-time Data Integration:**
  + The production units send data about the current plant state to the control center, which then forwards relevant information to the scheduling system.
  + The scheduling system updates the simulation model and the EA’s representation of the scheduling problem based on incoming real-time data.
  + The EA continuously runs between data updates, always maintaining a set of good solutions for a “warm start” upon the next disruptive event or update.
* **Evolutionary Algorithm Adaptations for Dynamic Scheduling:**
  + The EA’s genome (solution representation) is dynamically updated when plant conditions change:
    - Removal of already-started operations from the genome.
    - Addition of new orders.
    - Adjustments for changed resource availability, operation times, or due dates.
  + Re-initialization mechanisms after updates ensure that previous good solutions are leveraged in new conditions.
* **Simulation Model Updates and Communication Frequency:**
  + Periodic synchronization events and disruptive events trigger model updates.
  + After each update, the simulation model reflects the current plant state accurately, adjusting operation times and schedules accordingly.
  + Different intervals exist:
    - **∆teval**: Minimum interval tied to the EA’s generation evaluation cycles.
    - **∆tsync**: Synchronization interval for periodic updates.
    - **∆tIRS**: Immediate Response actions interval for urgent changes.
* **Interplay Between Scheduling System and Control Center:**
  + The simulator uses updated plant states from the control center to re-calculate and re-optimize schedules.
  + The scheduling decisions are then sent back to the control center for verification and implementation.

**Answers to the Questions:**

1. **How is the simulation set up and what parameters/data does it capture/needs from the real plant?**
   * **From the text:**
     + The simulation is set up as a high-fidelity discrete-event model that captures detailed operations, resource states (e.g., unit availability, maintenance), ongoing production orders, and their execution logic. It requires plant state information, including which operations have started or finished, current workload, and the status of units.
   * **More explanation possible:**
     + The text does not explicitly detail all the exact parameters or how these parameters are systematically extracted (beyond resource states, started/finished operations, etc.). Additional clarity could be provided on exact data formatting, parameter calibration, and how often resource availability or operator tasks are updated.
2. **How does data from the control center of the plant feed into the real-time scheduling framework?**
   * **From the text:**
     + Data flows from the production units to the control center, which extracts relevant state information and forwards it to the scheduling system. The scheduling system updates the simulation model and possibly the EA’s decision variables.
   * **In which frequencies does the data come in?**
     + **From the text:** Periodic synchronization intervals (∆tsync) and reactive updates (due to disruptive events) dictate when the model updates occur. There is also a minimum evaluation interval (∆teval) linked to completing an EA generation cycle. Immediate response actions (∆tIRS) occur as soon as possible.
   * **How seamlessly does the information flow?**
     + **From the text:** Information flow is described in a conceptual manner, showing a feedback loop between plant, control center, and scheduling system. It seems integrated but the text does not quantify the “seamlessness” or latency in detail. More explanation could be provided on data communication protocols, system latency, or reliability.
3. **The degree of fidelity with which the simulation captures real-world constraints:**
   * **From the text:**
     + The simulation is described as “high-fidelity,” incorporating detailed resource states, timing, sequencing, and even some tasks not explicitly modeled in the scheduling system (e.g., operator tasks).
     + The simulation can dynamically incorporate changing operation times, resource availabilities, and order priorities.
   * **More explanation possible:**
     + While labeled as “high-fidelity,” the exact limitations or simplifications are not fully described. Additional clarity could focus on what aspects of the plant are approximated or not captured, and whether stochastic aspects (e.g., unexpected downtimes, processing variability) are fully represented.

**Which Key Concepts are Explained and Which Key Answers are Provided?**

* **Key concepts explained:**
  + Integration of simulation and an evolutionary algorithm for dynamic, real-time scheduling.
  + The process of continuously updating the scheduling model and solution representation in response to real plant data.
  + Mechanisms for handling changes in orders, due dates, resource availability, and synchronizing schedules with actual plant operations.
* **Key answers provided:**
  + The text explains how plant data is integrated into the simulation (via control center) and how the EA and simulation model update based on new data.
  + It details the intervals for updates (periodic and event-driven) and how the simulation reflects real conditions at these intervals.
  + It describes the adjustment of the EA’s genome and how new conditions reinitialize the population.
* **Areas where more could be done:**
  + More explicit detail on data integration frequency and protocol.
  + A clearer explanation of the exact fidelity level and any assumptions or simplifications in the simulation model.
  + Additional operational details such as communication delays, data preprocessing, or specifics of data exchange formats would enhance understanding.

In summary, the passage provides a conceptual framework and methodology for simulation-based dynamic scheduling integrated with an EA, explains the updating mechanisms, and shows how data from the control center influences scheduling. It partially addresses the frequencies and nature of data exchange, but more detailed implementation specifics and fidelity measures could be beneficial.

## Comments

As a reader of the introduction, several aspects would pique my interest and shape my expectations for the subsequent sections of the paper:

Practical Implementation Details

The introduction promises a simulation-optimization framework that continuously adjusts to new information and events. I would be keen to understand the specifics of the integration:

* how exactly the simulation is set up and what parameters it captures (e.g., operation times, resource availability, maintenance events).
* I’d hope the next parts would explain how data from the plant control system feed into the model, how frequently and seamlessly the information flows,
* the degree of fidelity with which the simulation captures real-world constraints.

### Evolutionary Algorithm Configuration and Tailoring:

The introduction highlights the use of evolutionary algorithms for searching the solution space. I would be interested in learning how the algorithm is adapted or "tailored" to dynamic scheduling. Which genetic operators are used, and what modifications have been implemented to handle real-time updates and frequent rescheduling events? In the subsequent sections, I’d expect details on the chromosome representation, mutation/crossover operators, and any domain-specific heuristics that improve performance and convergence speed.

### Computational Performance and Response Times:

The introduction mentions that the objective is to find "near optimal" solutions rapidly enough to matter in real-time operation. As a reader, I would be eager to see quantitative results: reaction times after disturbances, average computation times per iteration, and scalability analyses. In the next sections, I’d hope for benchmarks of the computational effort relative to plant time scales, indications of how parallelization or approximation methods are employed, and clear metrics for evaluating the timeliness and reliability of solutions.

### Comparison to Other Methods and Idealized Schedulers:

The introduction references a comparison to an "idealized clairvoyant scheduler." I’d look forward to a well-structured comparison that highlights how much is lost by not knowing the future perfectly. Are there other well-established methods or baseline algorithms mentioned for comparison? I’d expect subsequent sections to include a rigorous performance evaluation, detailing how the proposed approach stacks up against other scheduling tools or methods. This should involve both quantitative metrics (e.g., tardiness, resource utilization) and qualitative assessments (e.g., robustness, adaptability).

### Case Study and Generalization Potential:

The introduction signals a focus on a pharmaceutical batch plant test case. While this is a strong application domain, I’d be curious whether the approach generalizes to other process industries, or even beyond process settings. I’d hope the next parts explain the chosen case study in depth, present its complexity, and then discuss how the proposed method might scale or adapt to other scenarios. What modifications are needed for different plant topologies, products, or event frequencies? Are there any limitations or assumptions that restrict its applicability?

### Tradeoffs and Sensitivity Analyses:

The introduction notes that there is a balance between reacting quickly and making better decisions. I’d be interested in seeing a more systematic exploration of these tradeoffs, perhaps through sensitivity analyses or scenarios with varying reaction times and levels of uncertainty. The following sections should show how the system behaves under different conditions, how stable the solutions are, and what the practical implications are for operators and decision-makers on the shop floor.

Overall, I would hope that the body of the paper provides a thorough, transparent account of how the framework is implemented and validated, how well it performs under realistic conditions, and whether it can be easily adapted to a range of industrial scheduling challenges.