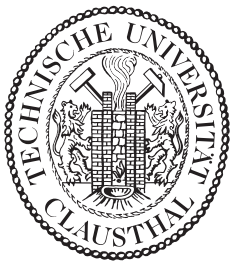
Institut für Wirtschaftswissenschaft
Technische Universität Clausthal

A priority-rule based method for predictive-reactive batch production scheduling in the process industries

Rafael Fink
Christoph Schwindt

Report PLC–3



TECHNICAL REPORT

Institut für Wirtschaftswissenschaft · TU Clausthal
Julius-Albert-Straße 2 · D–38678 Clausthal-Zellerfeld · Germany

Institut für Wirtschaftswissenschaft
Technische Universität Clausthal

A priority-rule based method for predictive-reactive batch production scheduling in the process industries

Rafael Fink
Christoph Schwindt

Report PLC–3

May 2007

Abstract. In this paper we present a priority-rule based method for predictive-reactive batch production scheduling in the process industries. In order to plan the processing of production operations on a multi-purpose batch production plant, we generate a predictive baseline production schedule. Given forecast values for the durations of the operations and the availabilities of the processing units, the problem consists in determining the start times of the operations such that the production makespan is minimized. In general, an operation may be executed on alternative processing units. Each operation transforms one or several input products into one or several output products. Certain intermediates must be stocked in storage facilities of limited capacity and chemically instable substances must be consumed immediately. Moreover, a cleaning of sequence-dependent duration may become necessary between the executions of different operations on the same processing unit.

During the execution of the baseline schedule, the durations of the operations may differ from their expected values. Furthermore, disruptions like unit breakdowns may occur. We show how to react on deviations from the predictive durations by updating the start and completion times of the operations and delaying the unloading of processing units, if necessary. For other types of disturbances we have to generate a new schedule, which is done in such a way that either the deviation from the baseline schedule or the makespan of the new plan is minimized. We have tested our method on a standard test set from literature.

Key words: Production scheduling, batch scheduling, predictive-reactive scheduling

This research has been supported by the Deutsche Forschungsgemeinschaft (Grant Schw 1178/1).

Contents

1 Introduction **1**

2 Predictive priority-rule based method **3**

3 Execution and reactive scheduling **6**

4 Experimental performance analysis **8**

5 Conclusions **10**

1 Introduction

In the process industries, final products arise from chemical and physical transformation processes, which are called tasks. Each task is executed on a processing unit and takes a certain processing time to transform the input materials into the output products. Often, a task can be executed on different alternative processing units. Moreover, a unit may be able to execute different tasks. Such a multi-purpose processing unit generally requires a changeover of sequence-dependent duration between the execution of different tasks. The products occurring at the different stages of the production processes can be classified into raw materials, intermediates, and final products. In literature, those products are also referred to as states. In the process industries, intermediates may need a quarantine time, which means that the intermediate cannot be consumed immediately after its production. The opposite case is possible as well: Products with a shelf-life time like perishable substances in the chemical industry have to be consumed within a certain amount of time. Intermediates with a shelf-life time of zero are referred to as zero-wait products. The remaining products can be stored in storage facilities of limited capacity.

In batch production mode, which we consider in this paper, material flows are discontinuous. Before starting the execution of a task, all input materials are loaded into a processing unit according to certain input proportions. After the completion of the task, which may produce several output products with different output proportions, the transformed materials are unloaded. The bulk of materials which is processed in one task is called a batch. The minimum and maximum filling levels of the processing units give rise to minimum and maximum batch sizes. The duration of a task is assumed to be independent of the batch size. Since batch sizes are bounded, each task has to be executed several times in general. The execution of a task with a given batch size is referred to as an operation. We assume operations to be non interruptible.

Batch production processes can be described using the state-task network representation proposed by Kondili et al. [11]. Figure 1 shows the state-task network belonging to a case study presented by Kallrath [10], to which we will refer to in Section 4.

The planning problem of batch production can be stated as follows. Given primary requirements for final products, compute a feasible production schedule with minimum makespan such that the batch sizes and the input and output proportions of the tasks are within their given bounds, the quarantine and shelf-life times of intermediates are observed, the processing units are executing at most one operation at a time, necessary changeover times are taken into account, a sufficient amount of input products is available at the start of each operation, and no capacity overflows occur in the storage facilities.

In literature, there exist different solution methods for the above planning problem. A considerable amount of research has been devoted to monolithic planning approaches. In particular, a broad variety of MILP formulations have been developed in the last decade. One can distinguish between discrete-time (see, e. g., Blömer and Günther [2] or Kondili et al. [11]) and continuous-time models (see, e. g., Ier-

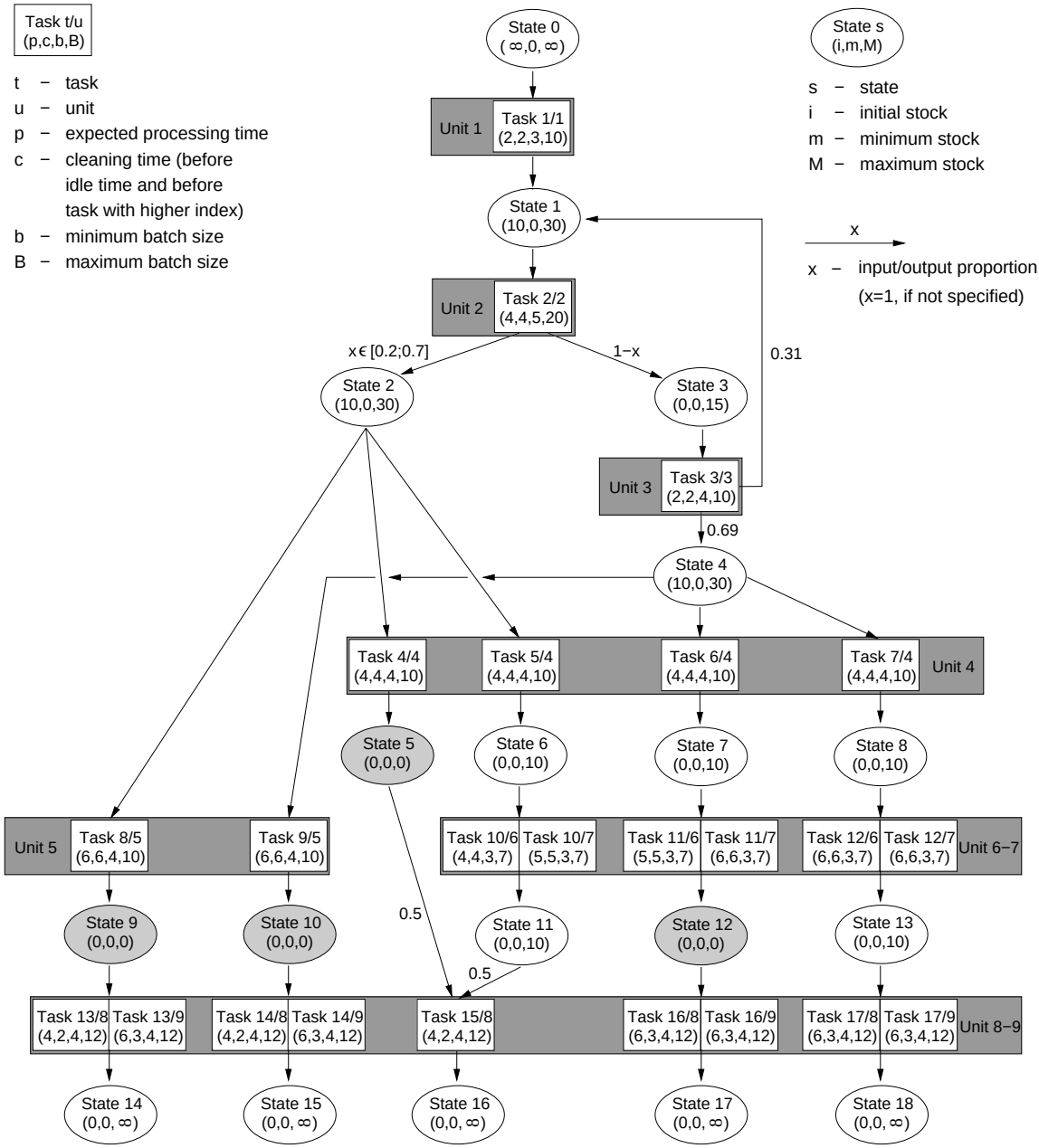


Figure 1: State-task network for the Kallrath example

apetritou and Floudas [7] or Maravelias and Grossmann [12]). A comparison of both model types can be found in the papers of Burkard and Hatzl [4] and Floudas and Lin [6]. Since most problems from practice cannot be solved to optimality in a reasonable amount of time, several heuristic approaches have been proposed to reduce the number of binary variables of the MILPs (see, e. g., Blömer and Günther [3]).

Neumann et al. [14] have developed a hierarchical planning approach decomposing the problem into a batching and a batch scheduling problem. A solution to the batching problem provides the set of operations to be scheduled in the batch scheduling problem. More precisely, the batching problem consists in determining the numbers and sizes of the batches and fixing the flexible input and output pro-

portions of the tasks. The batching problem can be formulated as an MINLP of moderate size. Once the batching problem has been solved, we assign a processing unit and a start time to each resulting operation at the batch scheduling level. Neumann et al. [14] have proposed a truncated branch-and-bound algorithm for solving the batch scheduling problem. An alternative heuristic procedure is due to Schwindt and Trautmann [17], who have developed a two-phase priority-rule based method. The latter algorithm has served us as a starting point for our research and as a benchmark for the experimental performance analysis.

The remainder of the paper is organized as follows. In Section 2, we describe a new single-phase priority-rule based method for constructing baseline schedules. The baseline schedule provides a basis for establishing temporal commitments like delivery dates of raw materials or customer-ordered final products. Section 3 addresses the case of uncertainty, when during the implementation of the predictive schedule different kinds of disruptions may occur. We explain how we can react on deviations between nominal and actual durations of the operations and on machine breakdowns. In Section 4 we present numerical results. Then we give some final remarks.

2 Predictive priority-rule based method

As mentioned above we follow the decomposition approach proposed by Neumann et al. [14]. In this paper we only focus on the batch scheduling problem, i. e., we are given a set of operations, denoted by $\mathcal{O}$, which must be scheduled. The formulation of the batching problem by Neumann et al. [14] ensures that operations of the same task always have the same batch size and the same input and output proportions. For computing the predictive baseline schedule, we assume that the durations of the operations equal the expected values of the respective task execution times. We illustrate our approach using the small example depicted in Figure 2.

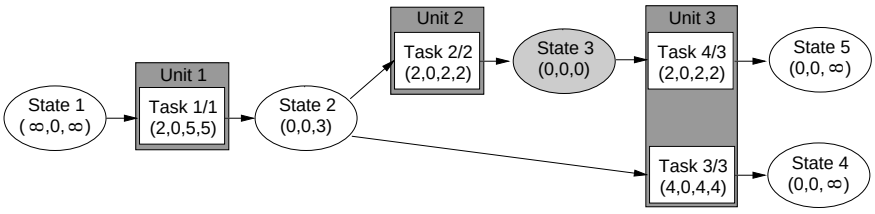


Figure 2: State-task network of the example

Assume that we have primary requirements of eight units for state 4 and of two units for state 5. Since all batch sizes are fixed, the solution to the batching problem is trivial. We obtain two operations of task 1, one operation of task 2, two operations of task 3, and one operation of task 4.

The first step of our algorithm consists in generating an event-on-node-network that corresponds to a type of networks introduced by Elmaghraby and Kamburowski [5]. Event nodes are connected by arcs representing temporal constraints between the events. A nonnegative weight $w_{ij} \geq 0$ of an arc (i, j) from node i to node j

corresponds to a minimum time lag of w_{ij} time units between the occurrence of the event belonging to node i and the occurrence of the event belonging to node j. A negative weight $w_{ji} < 0$ of an arc (j, i) from node j to node i means that the event belonging to node j must occur $-w_{ji}$ time units after the event belonging to node i at the latest.

The network is generated as follows. We introduce two nodes A and Ω for the production start and the production end and two nodes α and ω for each operation representing the operation's start and end. Operations belonging to the same task are arranged sequentially by arcs with weight 0. We use completion-to-start relationships if the corresponding task can be executed on only one unit. Otherwise, the arcs link the start nodes of the operations. Next, initial node A is linked with the start events α of the first operations of each task by an arc with weight 0. Symmetrically, the completion events ω of the last operations of each task are linked with terminal node Ω by an arc with weight 0. A fixed duration p of an operation is translated into a forward arc with weight p from α to ω and a backward arc with weight $-p$ from ω to α. For operations that can be executed on alternative processing units, the forward arc is weighted by the minimum and the backward arc by the negative maximum of the processing times on the different units.

Having generated the skeleton of the network, we may add further arcs in the following way. If an intermediate is produced by exactly one task (case of linear or divergent material flows), we can identify precedence relationships which are necessary for the timely availability of input materials. To this end, we separately consider each task consuming the intermediate. Starting with the first operation of this task, we calculate how many operations of the producing task must be completed before a sufficient amount of the intermediate is available to start the consuming operation. We then add an arc with weight zero from the end of the last required producing operation to the start of the consuming operation. Taking into account a possible residual stock of the intermediate, we proceed analogously with the remaining operations of the consuming task. If an intermediate is consumed by exactly one task (case of linear or convergent material flows), we can add arcs to avoid capacity overflows in a similar way. For our example we obtain the event-on-node-network depicted in Figure 3.

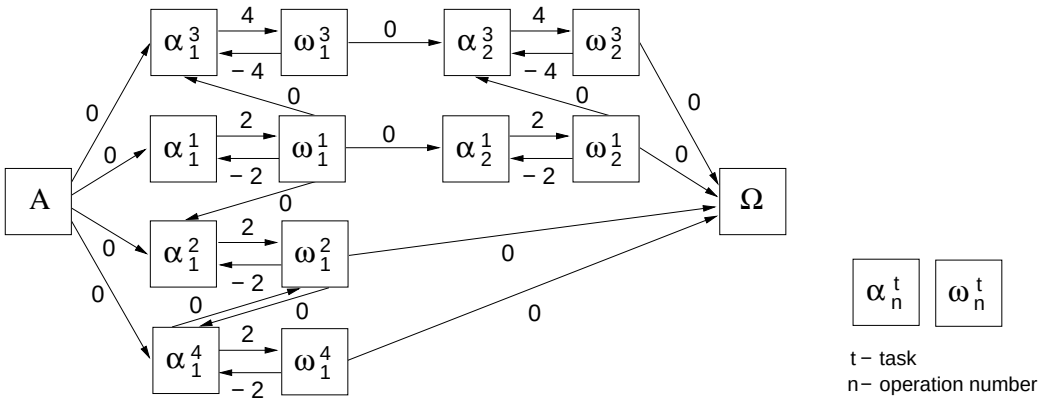


Figure 3: Event-on-node network for the example

Next, we determine the strong components of the network. Since every node of a strong component has a temporal relationship with each other node of the same strong component, in our method all operations belonging to a strong component are scheduled jointly. For our small example we obtain the strong components $\{A\}$, $\{\alpha_1^1, \omega_1^1\}$, $\{\alpha_2^1, \omega_2^1\}$, $\{\alpha_1^3, \omega_1^3\}$, $\{\alpha_2^3, \omega_2^3\}$, $\{\alpha_1^2, \omega_1^2, \alpha_1^4, \omega_1^4\}$, and $\{\Omega\}$.

The basic idea of the scheduling method is very simple. In each iteration we schedule one operation, the operations of a strong component being scheduled consecutively one after another. The start of each operation is scheduled at the earliest time for which the temporal constraints of the event-on-node-network are satisfied and a sufficient amount of input materials is available. The operations of a strong component are *eligible* to be scheduled if (i), all of those operations' predecessors in the event-on-node network outside the strong component have already been scheduled, (ii), there is enough input material available to process all operations of the strong component, and (iii), there is no other strong component for which some but not all operations have been scheduled. Condition (iii) ensures that all operations of a strong component are jointly added to the schedule in consecutive iterations.

So far we have not taken the limited capacity of the storage facilities into account. We consider the storage-capacity constraints via the concept of *capacity-driven latest start times*. If in some iteration of our method we have generated a capacity overflow in a storage facility at a time t, we temporary force eligible operations consuming the product stocked in this facility to start at time t at the latest. The capacity-driven latest start times are maintained until the capacity overflow has been removed. As a consequence it may happen that an eligible operation can no longer be scheduled because the capacity-driven latest start time is smaller than the earliest feasible start time. If no eligible operation can be scheduled, we perform an unscheduling step in the following way. We determine the origin of the capacity conflict, i. e., the operation i already scheduled that produced the material which cannot be stocked. Moreover, we select one of the eligible operations, say, operation j. For operation i causing the conflict we increase the earliest completion time to the earliest feasible start time t of operation j. This is done by introducing an arc from node A to the end node of i with weight t in the event-on-node network. If the conflict-causing operations i and j belong to the same strong component, we remove all operations belonging to this strong component from the schedule and resume the scheduling method. Otherwise, we restart the scheduling procedure from scratch. In both cases we apply the method to the expanded network containing the new arc. The unscheduling step may generate unnecessary idle times, which can easily be removed in a postprocessing step.

Algorithm 1 summarizes the priority-rule based method. Note that when updating the latest start times we not only have to take into account the capacity-driven latest start times but also the latest start times arising from precedence relationships between the operation j scheduled and the operations $j \in \mathcal{O} \setminus \mathcal{C}$ belonging to the same strong component. The algorithm can be implemented as a randomized multi-start procedure by stochastically disturbing the priority values of the operations.

For our example the algorithm yields the schedule shown in Figure 4, where we

Algorithm 1 Priority-rule based method

initialize $\mathcal{C} := \emptyset$ ($*$ set of scheduled operations $*$);
while $\mathcal{C} \neq \mathcal{O}$ **do**
  determine set $\mathcal{E}$ of eligible operations $i \in \mathcal{O} \setminus \mathcal{C}$;
  for all $i \in \mathcal{E}$ **do**
    compute priority value $v(i)$;
  repeat
    delete operation j with highest priority value $v(j)$ from set $\mathcal{E}$;
    if j can be started no later than its latest start time **then**
      schedule operation j at its earliest feasible start time;
      put $\mathcal{C} := \mathcal{C} \cup \{j\}$;
      update latest start times of operations $i \in \mathcal{O} \setminus \mathcal{C}$;
  until an operation j has been scheduled or $\mathcal{E} = \emptyset$;
  if $\mathcal{E} = \emptyset$ **then**
    perform an unscheduling step;

assume that the operations of task 3 have a higher priority than the operations of tasks 2 and 4. The second operation of task 1 has been unscheduled and delayed. A second unscheduling step has been performed for the operation belonging to task 2 while trying to schedule the stong component containing the operations of tasks 2 and 4.

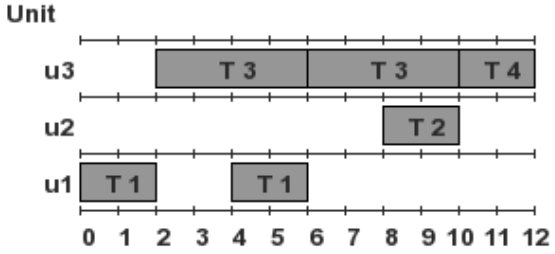


Figure 4: Predictive schedule

3 Execution and reactive scheduling

Although production in the process industries is typically subject to considerable uncertainty, the literature on production planning in the process industries primarily deals with deterministic scenarios. Schilling and Pantelides [15] have proposed a reactive batch scheduling procedure based on a discrete-time MILP. More recent approaches relying on continuous-time MILP's can be found in Mendez and Cerdá [13] and in Janak and Floudas [9]. In machine and project scheduling, a big deal of effort has been devoted to the case of uncertain durations. In principle, there are two different ways to cope with uncertainty: the proactive and the reactive approach. Proactive methods anticipate uncertainty, and the knowledge about uncertainty is included into the plan. Examples of proactive approaches are stochastic and robust

optimization (see, e. g., Scholl [16]). Reactive procedures serve to react on unforeseen events like deviations from predictive durations or machine breakdowns during the implementation of a baseline plan. Overviews over the different approaches are given in Aytug et al. [1], Herroelen and Leus [8], and Vieira et al. [18].

In this paper we focus on the reactive approach. At first, we explain how to execute the predictive schedule in face of uncertain durations of the operations. Then, we present a reactive scheduling method which can be used in case of unit breakdowns.

When executing the baseline schedule, the durations of the operations will almost never coincide with the predictive processing times and hence, the actual start and completion times of the operations will generally differ from the baseline schedule. That is why the baseline schedule can no longer be viewed as a binding specification of the operations' execution time intervals. Rather, we extract precedence relationships from the baseline schedule which guarantee the feasibility of the schedule and which have to be maintained during the execution. In more detail, we fix the sequence of the operations on the units, and for every product we link the producing and the consuming operations by temporal constraints according to a FIFO strategy in a way ensuring that shortages and capacity overflows are avoided. We then schedule all operations as early as possible with respect to the temporal constraints. When operations are finished, we either unload the batch and proceed with the next operations or we leave the produced material in the unit as long as it cannot be stocked or consumed by another operation. By using units as material buffers, this approach ensures that every feasible baseline schedule can actually be executed. The implementation of the schedule from Figure 4 with realized durations is shown in Figure 5. The black rectangles indicate the time during which the finished batch cannot be unloaded.

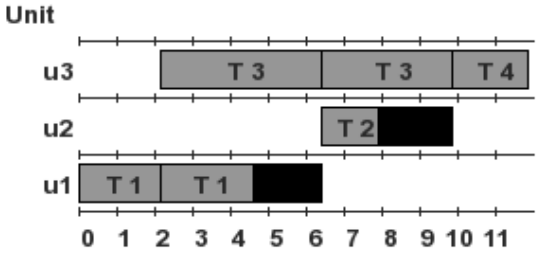


Figure 5: Implemented schedule

Other types of schedule disruptions, like the breakdown of a unit, cannot be handled by a simple adjustment of start and completion times. If a unit fails, we assume that the material, which has been in process, is lost. Consequently, we have to do some rework. This means that we must compute a new schedule, which can be done in the following way. We virtually interrupt the execution of the current schedule by starting a repair job on the broken unit. Then, we take the current inventory levels of the products including the output of the currently running operations as initial stocks for a new planning problem.

The new problem is treated in the same way as the original problem. At first, we solve the new batching problem for the residual primary requirements and sub-

sequently schedule the resulting operations with the priority-rule based method sketched in Section 2. With respect to the batch scheduling problem, we distinguish between two different types of reactive strategies. Either we try to minimize the makespan again, or we strive at constructing a schedule which resembles the old predictive schedule to the largest extent possible and hence maximizes planning stability. A simple way to pursue the latter objective is list scheduling. In our context list scheduling means that we generate a task list of the operations not yet started from the previous predictive schedule according to nondecreasing start times. When computing the revised schedule with the priority-rule based method, we always schedule an operation belonging to the first eligible task in the list. If running operations produce material that exceeds the capacity of a storage facility, we sometimes may not be able to generate a new feasible schedule without disposing the amount of material that cannot be stored. More precisely, this case occurs when during the generation of the revised schedule we perform an unscheduling step in which we try to delay the completion of an operation that is already in progress. Finally, we note that an analogous procedure can be used for other disruption types like unexpected yield losses.

We return to our small example of Figure 5 and suppose that unit 1 breaks down after 2.89 units of time, i. e., while executing the second operation of task 1. Independently of the priority values, the schedules generated with the both reactive strategies do not differ. The repaired schedule, which is depicted in Figure 6, then serves as the new baseline schedule.

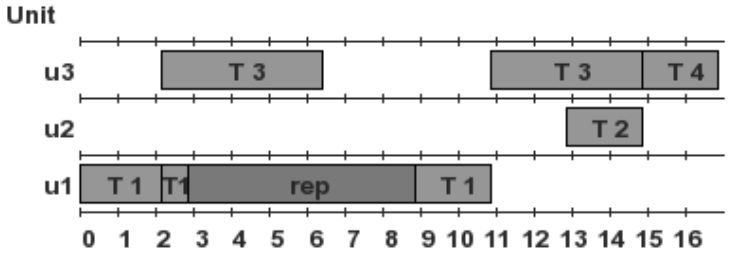


Figure 6: Reactive schedule

4 Experimental performance analysis

We have tested our algorithm on a set of instances that are based on Kallrath's case study illustrated in Figure 1. The 22 instances, which differ in the primary requirements for the five final products (see Table 1), have been created by Blömer and Günther [3]. We compare the results for our predictive method to the results obtained by Schwindt and Trautmann [17], who stopped their algorithm after 60 seconds. Using a faster PC (2.08 GHz), we have set a time limit of 23 seconds for our randomized multi-start procedure. Table 2 lists the best makespans obtained. $C_{\max}^{ST}$ is the best solution found by Schwindt and Trautmann, and $C_{\max}^{FS}$ stands for the best makespan found with our method. The underlying solutions to the batching problems were the same for both algorithms. Table 2 indicates that the new method is able to achieve significantly better results, especially for the large instances.

Inst.	1	2	3	4	5	6	7	8	9	10	11
$PR14$	20	20	20	20	20	20	0	0	0	0	10
$PR15$	20	20	20	0	0	0	20	20	20	0	10
$PR16$	20	0	0	20	20	0	20	20	0	20	20
$PR17$	0	20	0	20	0	20	20	0	20	20	20
$PR18$	0	0	20	0	20	20	0	20	20	20	30

Inst.	12	13	14	15	16	17	18	19	20	21	22
$PR14$	30	10	18	15	45	15	27	20	60	20	36
$PR15$	20	20	18	15	30	30	27	20	40	40	36
$PR16$	20	30	18	30	30	45	27	40	40	60	36
$PR17$	10	20	18	30	15	30	27	40	20	40	36
$PR18$	10	10	18	45	15	15	27	60	20	20	36

Table 1: 22 instances of the Kallrath case study

Inst.	1	2	3	4	5	6	7	8	9	10	11
$C^{ST}_{\max}$	39	47	48	42	41	49	44	43	49	54	64
$C^{F}_{\max}$	36	42	42	39	39	47	40	40	46	49	56

Inst.	12	13	14	15	16	17	18	19	20	21	22
$C^{ST}_{\max}$	48	57	61	45	80	92	90	159	97	124	114
$C^{F}_{\max}$	45	50	53	80	66	72	70	98	78	82	83

Table 2: Results for 22 instances of the Kallrath case study: Predictive case

Inst.	1	2	3	4	5	6	7	8	9	10	11
π	0.063	0.069	0.103	0.093	0.072	0.045	0.063	0.041	0.073	0.045	0.086

Inst.	12	13	14	15	16	17	18	19	20	21	22
π	0.082	0.044	0.087	0.044	0.036	0.050	0.045	0.038	0.052	0.037	0.059

Table 3: Results for the Kallrath case study: Reactive case for uncertain durations only

For the procedure coping with uncertain durations, we have assumed the durations of the operations to be distributed according to a right-skewed beta-distribution and have performed an ex-post analysis. In this type of analysis, we compare the implemented schedule with the ex-post schedule that we would have obtained if we had known the realized durations in advance. To evaluate the results we have calculated the price of incomplete information π, i. e., the relative deviation between the realized makespan and the ex-post benchmark. Table 3 shows the mean price of incomplete information obtained in 20 independent passes for each instance.

Inst.	1	2	3	4	5	6	7	8	9	10	11
π^{MS}	0.157	0.155	0.144	0.128	0.112	0.162	0.176	0.118	0.170	0.115	0.164
π^{ST}	0.173	0.167	0.151	0.156	0.122	0.174	0.194	0.132	0.200	0.124	0.171
σ^{MS}	0.572	0.590	0.655	0.592	0.579	0.682	0.582	0.598	0.663	0.670	0.625
σ^{ST}	0.655	0.693	0.713	0.680	0.687	0.787	0.698	0.675	0.753	0.729	0.782

Inst.	12	13	14	15	16	17	18	19	20	21	22
π^{MS}	0.114	0.125	0.139	0.112	0.105	0.107	0.103	0.117	0.106	0.112	0.110
π^{ST}	0.130	0.167	0.180	0.138	0.131	0.153	0.123	0.164	0.129	0.179	0.144
σ^{MS}	0.611	0.637	0.649	0.607	0.681	0.601	0.663	0.691	0.617	0.633	0.641
σ^{ST}	0.768	0.736	0.785	0.798	0.817	0.770	0.801	0.822	0.817	0.789	0.811

Table 4: Results for the Kallrath case study: Reactive case for unit breakdowns and uncertain durations

For the case of unit breakdowns we have assumed exponentially distributed times between failure. Again, we have evaluated the procedure based on an ex-post analysis, where breakdown and repair times are given. To measure the similarity between the original baseline and the actually executed schedule we use schedule-induced reflexive linear orders. Let $\mathcal{P}^u(S)$ denote the reflexive linear order that is induced by the sequence of operations on unit u within schedule S. The similarity σ of two schedules S^1 and S^2 is then defined to be the ratio of the number of precedence relationships that are induced by both schedules to the mean number of precedence relationships induced by schedule S^1 and S^2, i. e.,

$$\sigma = \frac{\sum_u \mid \mathcal{P}^u(S^1) \cap \mathcal{P}^u(S^2) \mid}{\frac{1}{2}(\sum_u \mid \mathcal{P}^u(S^1) \mid + \sum_u \mid \mathcal{P}^u(S^2) \mid)}$$

The results are reported in Table 4. Depending on the reactive strategy used, the price of incomplete information π and the similarity σ are labeled with MS for the makespan strategy and ST for the stability strategy. Again, the results refer to 20 independent passes for each instance. Compared to the case where we have exclusively considered uncertain durations, π is now generally much greater. The reason for this is that often the amount of rework becomes considerably large.

5 Conclusions

In this paper we have presented a new priority-rule based method for predictive-reactive batch production scheduling in the process industries. The predictive method is based on an event-on-node network and its strong components. Since the operations of a strong component have a temporal relationship to each other they are scheduled jointly one after another. When scheduling the operations we strictly avoid shortages in the storage facilities. Capacity overflows that may have been generated are resolved via the concept of capacity-driven latest start times.

When implementing the baseline schedule the processing times of the operations may differ from their predictive values. We have shown how to react on these deviations by adjusting the start and completion times of the operations. Moreover, we have explained how to generate a revised schedule by generating and solving a new planning problem when units break down.

The results and the very short CPU times (less than one minute) indicate that the proposed method offers a promising approach to dealing with uncertainty in batch production scheduling.

The proposed procedure can be adapted to other types of disruptions like unexpected yield losses. Another interesting extension of the method would be a robust planning of the set operations to be scheduled with respect to the possible material deficits caused by unit breakdowns or yield losses. Such a set of operations can be obtained, for example, by formulating the batching problem as a chance-constrained program.

References

[1] Aytug H, Lawley MA, McKay K, Mohan S, Uzsoy R (2005) Executing production schedules in the face of uncertainties: A review and some future directions. European Journal of Operational Research 161:86–110

[2] Blömer F, Günther HO (1998) Scheduling of a multi-product batchprocess in the chemical industry. Computers in Industry 36:245–259

[3] Blömer F, Günther HO (2000) LP-based heuristics for scheduling chemical batch plants. International Journal of Production Research 38:1029–1051

[4] Burkard R, Hatzl J (2005) Review, extensions and computational comparison of MILP formulations for scheduling of batch processes. Computers and Chemical Engineering 29:1752–1769

[5] Elmaghraby SE, Kamburowski J (1992) The analysis of activity networks under generalized precedence relations (GPRs). Management Science 38:1245–1263

[6] Floudas C, Lin X (2004) Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. Computers and Chemical Engineering 28:2109–2129

[7] Ierapetritou M, Floudas C (1998) Effective continuous-time formulation for short-term scheduling: 1. Multipurpose batch processes. Industrial and Engineering Chemistry Research 37:4241–4359

[8] Herroelen W, Leus R (2005) Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research 165:289–306

[9] Janak S, Floudas C, Kallrath J, Vormbrock N (2006) Production Scheduling of a Large-Scale Industrial Batch Plant. II. Reactive Scheduling. Industrial Engineering Chemistry Research 45:8253–8269

[10] Kallrath J (2002) Planning and scheduling in the process industry. OR Spectrum 24:219–250

[11] Kondili E, Pantelides CC, Sargent RWH (1993) A general algorithm for short-term scheduling of batch operations: I. MILP formulation. Computers and Chemical Engineering 17:211–227

[12] Maravelias C, Grossmann I (2003) New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial and Engineering Chemistry Research 42:3056–3074

[13] Mendez C, Cerdá J (2004) An MILP framework for batch reactive scheduling with limited discrete resources. Computers and Chemical Engineering 28:1059–1068

[14] Neumann K, Schwindt C, Trautmann N (2002) Advanced production scheduling for batch plants in process industries. OR Spectrum 24:251–279

[15] Schilling G, Pantelides CC (1997) General algorithms for reactive rescheduling of multipurpose plants. Working paper, Imperial College of Science, Technology and Medicine, London

[16] Scholl A (2001) Robuste Planung und Optimierung. Physica, Heidelberg

[17] Schwindt C, Trautmann N (2004) A priority-rule based method for batch production scheduling in the process industries. In: Ahr D, Fahrion R, Oswarld M, Reinelt G (eds.) Operations Research Proceedings 2003, Springer, Berlin, 111–118

[18] Vieira G, Herrmann JW, Lin E (2003) Rescheduling manufacturing systems: A framework of strategies, policies, and methods. Journal of Scheduling 6:39–62