A hybrid method for large-scale short-term scheduling of make-and-pack production processes

Philipp Baumann[a], Norbert Trautmann[a,*]

[a]*Schützenmattstrasse 14, 3012 Bern, Switzerland*

Abstract

Due to the ongoing trend towards increased product variety, fast-moving consumer goods such as food and beverages, pharmaceuticals, and chemicals are typically manufactured through so-called make-and-pack processes. These processes consist of a make stage, a pack stage, and intermediate storage facilities that decouple these two stages. In operations scheduling, complex technological constraints must be considered, e.g., non-identical parallel processing units, sequence-dependent changeovers, batch splitting, no-wait restrictions, material transfer times, minimum storage times, and finite storage capacity. The short-term scheduling problem is to compute a production schedule such that a given demand for products is fulfilled, all technological constraints are met, and the production makespan is minimised. A production schedule typically comprises 500–1500 operations. Due to the problem size and complexity of the technological constraints, the performance of known mixed-integer linear programming (MILP) formulations and heuristic approaches is often insufficient.

We present a hybrid method consisting of three phases. First, the set of operations is divided into several subsets. Second, these subsets are iteratively scheduled using a generic and flexible MILP formulation. Third, a novel critical path-based improvement procedure is applied to the resulting schedule. We develop several strategies for the integration of the MILP model into this heuristic framework. Using these strategies, high-quality feasible solutions to large-scale instances can be obtained within reasonable CPU times using standard optimisation software. We have applied the proposed hybrid method to a set of industrial problem instances and found that the method outperforms state-of-the-art methods.

Keywords: scheduling, make-and-pack production, hybrid method, real-world production process

*Corresponding author

Email addresses: philipp.baumann@pqm.unibe.ch (Philipp Baumann), norbert.trautmann@pqm.unibe.ch (Norbert Trautmann)

1. Introduction

In Europe, the process industries account for 59% of the gross value added by all manufacturing industries[1]. Many companies in these industries are increasing their product range to better meet the individual needs of their customers, especially in the fast-moving consumer goods sector, in which products such as beer, ice-cream, dairy products, candy, condiments, drugs, toothpaste, detergents, and hair dyes are sold in various versions and package types. The production process generally consists of a make stage and a pack stage, which are decoupled by storage facilities. In the make stage, various intermediates are manufactured. In the pack stage, these intermediates are packed into different package types. The efficient utilisation of the available production resources is of particular importance because of the competitiveness and low unit contribution margins in these industries.

In make-and-pack production processes, raw materials are transformed into (final) products through a series of transformation tasks, e.g., mixing, curing, and packing. The typical operating conditions include non-identical parallel processing units, partial equipment connectivity, sequence-dependent changeovers, batch splitting, no-wait restrictions, material transfer times, minimum storage times, and multipurpose storage units with limited capacities. Because of the maximum filling level of the processing units and storage tanks, the total demand must be divided into batches. In the following, we assume that the size of these batches is predetermined. This is often the case in practice, e.g., to utilise the maximum capacity of the processing and storage units or to satisfy safety regulations (in the pharmaceutical industry, for instance). We refer to the combination of a task and a batch as an operation. The planning problem discussed in this paper consists of assigning a start time and processing unit to each operation such that all technological constraints are met, the demand is fulfilled and a certain planning objective is optimised. Typical planning objectives are the minimisation of the makespan or total weighted tardiness. In practical applications, a large number of batches are processed, rendering the short-term scheduling of make-and-pack production processes a difficult task.

In the literature, a large variety of mixed-integer linear programming (MILP) formulations has been proposed for this short-term scheduling problem. Owing to improvements in modelling techniques, optimisation software, and computer hardware, nowadays such MILP formulations are not only applicable to small, but also to medium-sized problem instances. However, the performance of these formulations for large-scale problem instances is still insufficient for practical applications. In addition to these MILP formulations, various heuristics and metaheuristics have also been proposed; however, these algorithms are designed for relatively simple production processes. Heuristic approaches become less efficient in the presence of complex technological constraints because

[1] Source: Statistics on the production of manufactured goods 2011, Eurostat, as of 2012-10-04

it is already difficult to devise feasible schedules. In addition, the quality of the solutions cannot be controlled systematically. Eventually, some hybrid methods have been successfully applied to medium-sized problem instances. These heuristic algorithms combine exact methods with decomposition or aggregation techniques, allowing for systematic control of the size of the search space to balance the trade-off between solution quality and CPU time. The techniques for reducing the search space include, e.g., preordering rules (cf. Méndez and Cerdá 2002b), block planning (cf. Günther et al. 2006) and iterative scheduling (cf. Roslöf et al. 2002; Kopanos et al. 2010).

In this paper, we present a hybrid method designed for large-scale instances that consists of three phases: decomposition, construction, and improvement. In the decomposition phase, the set of all batches is decomposed into groups of a predefined size. In the construction phase, an initial schedule is generated by iteratively scheduling the groups of batches using an MILP model. To obtain a suitable model formulation, we start from the MILP model introduced in Baumann and Trautmann (2013); this precedence-based model is designed for complex make-and-pack production processes and can be applied to small and medium-sized problem instances. We propose two adaptations of this model. First, we modify the objective function to account for situations in which several alternativ solutions exist in a given iteration. Second, we show how to eliminate redundant variables and constraints to reduce the CPU time required per iteration. In the improvement phase, the initial schedule is iteratively improved by identifying and rescheduling critical groups of batches. We show how to identify the critical groups of batches efficiently by solving two LP models that are derived directly from the MILP model. We apply this hybrid method to a set of real-world instances introduced in Honkomp et al. (2000). The results indicate that the heuristic outperforms the best solution method found in the literature (cf. Fündeling and Trautmann 2006). We also compare the proposed hybrid method to an exact MILP model using a set of small- and medium-sized instances derived from the same real-world production process. The hybrid method (approximately) solves all of these instances to optimality.

Due to the algorithmic limitations of standard optimisation software, most MILP formulations of scheduling problems cannot achieve optimal or even feasible solutions for large-scale problem instances. However, a major advantage of MILP formulations is the flexibility to account for the specific structures and operating conditions involved in complex production processes. We therefore use the MILP approach as our starting point for developing a solution methodology that can be applied to large-scale instances of real problems. The general aim of this paper is to provide simple and effective strategies for integrating a generic MILP formulation into the construction and improvement phases of a hybrid method, such that the resulting MILP models can be solved using standard optimisation software within reasonable CPU times.

The remainder of this paper is organised as follows. We describe the characteristics of make-and-pack production processes in Section 2, using the real-world process presented in Honkomp et al. (2000) as an example. In Section 3, we review the related literature. In Section 4, we present the solution strategy

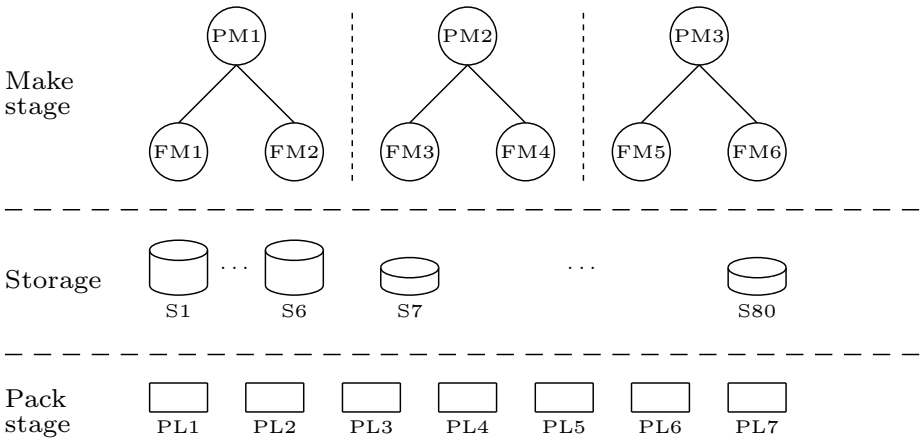

Figure 1: Equipment structure

of our hybrid method and describe the implementation of this strategy in the decomposition, construction and improvement phases in detail. We describe the MILP and LP formulations employed in Section 5; in particular, we introduce the modified objective function and demonstrate how to eliminate redundant variables and constraints. We report our computational results in Section 6 and provide concluding remarks and directions for further research in Section 7.

2. Planning situation

In this section, we illustrate the characteristics of make-and-pack production processes using the real-world process described in Honkomp et al. (2000). All of the data were provided by the Procter & Gamble Company. In Subsection 2.1, we sketch the structure of the production process. In Subsections 2.2, 2.3, and 2.4, we describe the operating conditions for the make stage, storage, and pack stage, respectively. In Subsection 2.5, we state the resulting short-term scheduling problem.

2.1. Equipment structure

The equipment structure of the production process is depicted in Figure 1. The make stage consists of three non-identical groups of premix units and final-mix units. A premix unit can only be connected to a final-mix unit that belongs to the same group. The pack stage consists of seven parallel, non-identical packing lines. Eighty storage tanks with different capacities decouple the make stage from the pack stage.

2.2. Make stage

In the make stage, 59 different intermediates are produced in batches of size ten. Most of the intermediates require both a premix and a final-mix task (see

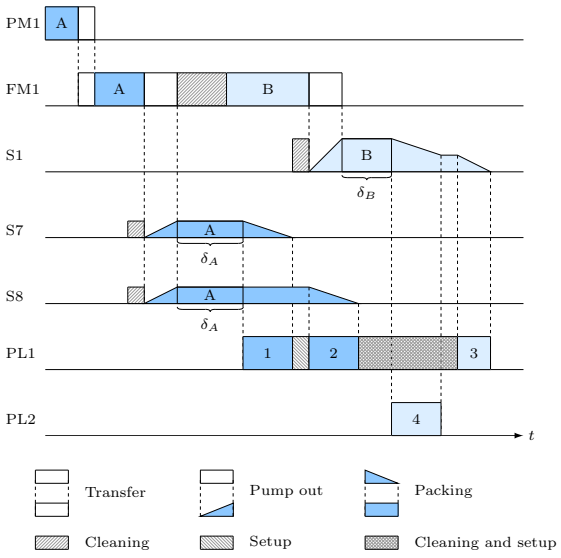

Figure 2: Operating conditions

Figure 2, Batch A). First, the raw material is converted into a compound in an appropriate premix unit. The compound is then directly transferred into a final-mix unit in which further material is added. During the transfer, both involved processing units are occupied. The remaining intermediates only require a final-mix task (Batch B). For those intermediates, the entire raw material transformation takes place in the final-mix unit. It can be assumed that raw material is available in sufficient quantity.

Each compound and each intermediate belongs to a wash-out family. Between the processing of batches belonging to specific pairs of wash-out families, the premix and final-mix units must be cleaned. The duration of the cleaning is neither unit- nor intermediate-dependent. The processing times of the premix and final-mix operations depend on the compound and intermediate but not on the processing unit.

2.3. Storage

All of the storage tanks are multipurpose, i.e., different intermediates can be stored in the same tank, but only one at a time. In this process, every intermediate can be stored in every storage tank. Six of the tanks have a capacity of ten, and the remaining 74 tanks have a capacity of five. Immediately following the completion of the final-mix task, the intermediate is pumped into either two tanks with capacity five (Batch A) or one tank with capacity ten (Batch B). The intermediate must stay within the tank(s) for at least a specified quarantine time (δ_A or δ_B). Before a batch is pumped out into a storage tank, the tank must be cleaned. The duration of the pumping and quarantine time depend on the intermediate, but not on the storage tank.

2.4. Pack stage

In the pack stage, the intermediates are packed in batches of size five, and 22 different package types are used. We refer to the combination of an intermediate and a package type as a product. In total, 203 different products are sold to customers. Intermediates stored in tanks with capacity ten are split into two portions of size five, which can be packed on the same packing line or on two different packing lines (Batch B). During the packing operation, the intermediate is continuously extracted from the storage tank. It can be assumed that unlimited storage space is available for all products.

The duration of a packing operation depends on the intermediate, package type and packing line. The packing lines must be cleaned between the packing of intermediates belonging to specific pairs of wash-out families. In addition, a changeover is required when the package type is switched. The cleaning and changeover can be performed in a single operation, which has a shorter duration than the sum of the cleaning and changeover times. In Figure 2, we assume that Batch A is packed in two different types of packages and that Batch B is packed in a third type of package.

2.5. Planning problem

A given set of operations is required to fulfil the product demand for a planning horizon of one week; the due date of the demand coincides with the end of the planning horizon. There are typically 500–1500 operations to be scheduled.

We seek (a) an assignment of the processing units and storage tanks to the operations and (b) a start time for the processing of each operation such that all technological constraints are satisfied and the makespan of the production schedule is minimised. As noted in Honkomp et al. (2000), a low makespan results in lower labour costs. Moreover, minimising the makespan generally reduces the total changeover time and thereby contributes to an efficient utilisation of the production resources.

We assume that the changeover times satisfy the weak triangular inequality, i.e., the changeover time from batch A to batch C is less than or equal to the sum of the processing time of batch B and the total changeover time from A to B followed by B to C. The data provided by the Procter & Gamble Company for the production process described above satisfies this assumption.

3. Related literature

In this section, we provide an overview of various models and solution methods for the short-term scheduling of make-and-pack production processes. In Subsections 3.1 and 3.2, we review MILP models and heuristic methods for the scheduling of multi-product batch processes, respectively. In Subsection 3.3, we cover hybrid methods that combine heuristic techniques with exact methods. In Subsection 3.4, we describe hybrid approaches that rely on a decomposition approach in detail.

3.1. Exact methods

A large variety of MILP models for multi-stage batch scheduling has been proposed in the literature; for general reviews, we refer the reader to Kallrath (2002), Floudas and Lin (2004), and Méndez et al. (2006). In these models, the planning horizon is divided into a set of time intervals. In continuous-time models, the length of these intervals is determined implicitly in the course of the solution of the MILP. In discrete-time models, the length of the time intervals is fixed and is usually determined as the highest common factor of the processing times, material transfer times, quarantine times, and changeover times. A large number of time intervals is required for most real-world problems, rendering the resulting discrete-time model prohibitively large (see, e.g., Stefansson et al. 2011). In the following, we therefore concentrate on continuous-time models, which we further classify into network-based and batch-based models.

In network-based models, the production process is represented either by a so-called state-task network (cf. Kondili et al. 1993) or by a resource-task network (cf. Pantelides 1994). Models of the former type have been proposed by, e.g., Maravelias and Grossmann (2003), Sundaramoorthy and Karimi (2005), and Erdirik-Dogan and Grossmann (2008); for models of the latter type, we refer the reader to, e.g., Castro and Grossmann (2005), Castro et al. (2006), Castro and Novais (2009), and Shaik and Floudas (2008). The model described in Giménez et al. (2009a,b) covers the broadest range of technological constraints occurring in make-and-pack production processes. A major advantage of network-based models is that batching and scheduling decisions are considered simultaneously, i.e., the decision regarding the number and size of the batches is part of the optimisation. Therefore, network-based models are typically used to address production processes that involve flexible batch mixing and splitting, cyclic material flows, and multiple storage policies. However, additional tasks are required to model partial equipment connectivity, sequence-dependent changeovers, and time-consuming material transfers. These additional tasks usually deteriorate the computational performance of the models considerably.

In batch-based models, batches are treated as discrete entities that move sequentially through the stages of the production process. In most models, the processing sequence of the batches is defined by immediate- or general-precedence relationships. In immediate precedence-based models (cf. e.g., Cerdá et al. 1997 and Gupta and Karimi 2003), an immediate predecessor batch is defined for each batch. This approach makes it possible to efficiently consider sequence-dependent changeover times or costs. In general precedence-based models (cf. e.g., Jain and Grossmann 2000, Kopanos et al. 2009, Kopanos et al. 2011, Méndez and Cerdá 2000, 2002a,b, 2003b, Méndez et al. 2001, Sundaramoorthy and Maravelias 2008a,b, and Elzakker et al. 2012), all direct and indirect predecessor batches are considered for each batch. The latter approach requires fewer binary variables and allows shared resources such as storage tanks to be handled without the need for additional variables. To our knowledge, the general precedence-based model proposed by Baumann and Trautmann (2013) is the only MILP model that accounts for all of the technological constraints in make-and-pack production processes such as that described in Honkomp et al.

(2000). Despite the sparse use of binary variables and introduction of symmetry-breaking constraints, the model can only be applied to instances with up to 180 operations. For larger instances, the computational cost becomes prohibitively high.

We conclude that MILP models offer the flexibility to easily accommodate complex technological constraints, but they are not appropriate for large-scale instances. For our hybrid approach we use the general-precedence framework for two reasons. First, it has been shown that general-precedence based models generally require less computational effort than network-based models. Second, a broad range of production processes and important operating conditions can be modeled using general-precedence variables.

3.2. Heuristics

Compared with the large variety of MILP models, only a few heuristic approaches have been proposed for short-term scheduling of make-and-pack production processes. For a particular make-and-pack production process in the shampoo industry, Belaïd et al. (2010, 2012) and Belaïd et al. (2011) propose specific heuristics for the scheduling of the storage and make stage, respectively. The start times for the batches in the pack stage are derived from the due dates of the customer orders and represent deadlines for the corresponding operations in the preceding stage. The objective is to minimise the number of cleanings of the processing units and storage tanks. Fündeling and Trautmann (2006) develop a priority rule-based heuristic tailored to the production process described in Honkomp et al. (2000). The priority rule determines the order in which the batches are scheduled, and a problem-specific selection rule is used to assign the batches to processing units and storage tanks. Fündeling and Trautmann (2006) propose 287 different multi-level priority rules for determining the scheduling order of batches. The heuristic can be applied as a single-pass method using one specific priority rule or as a multi-pass sampling method that computes for the same problem instance multiple schedules, each with a different priority rule. The computational results reported in Baumann and Trautmann (2013) indicate a considerable performance gap relative to the MILP models even when multi-pass sampling is employed.

For metaheuristics such as simulated annealing, tabu search, and population-based search methods, good performance has been reported for various scheduling problems. However, no such metaheuristics have been developed for make-and-pack production processes, possibly because these methods often generate infeasible solutions during the search when complex technological constraints are imposed. Three different approaches have been proposed to overcome this drawback. The first approach is to repair infeasible solutions. Raaymakers and Hoogeveen (2000) propose a simulated annealing method with such a repair procedure for a scheduling problem with no-wait restrictions between operations. Depending on the number of constraint violations, the repair procedure may require a considerable amount of CPU time. The second approach is to avoid infeasibility. Venditti et al. (2010) developed a tabu search algorithm that uses an acyclic graph to represent a schedule. Prior to each schedule modification, a

feasibility test is performed to ensure that the resulting graph does not contain any cycles. Although the feasibility requires little CPU time, the performance is better in less constrained problems. Ruiz and Maroto (2006) present a genetic algorithm in which a solution is represented as a permutation of batches. A schedule-generation scheme translates the permutation into a feasible schedule. Due to the simplified representation, the search is limited to a specific area of the solution space. In the so-called random-key genetic algorithm of Kurz and Askin (2004), the representation is less simplified because both the sequencing and unit assignment decisions are included. A real number is assigned to each batch, whose integer part defines the assigned machine and whose fractional part is used to order the jobs assigned to each machine. A larger area of the solution space is thereby covered. However, in the presence of unit assignment restrictions, infeasible schedules may be generated. The third approach to handle infeasibility is to impose a penalty on constraint violations, cf. e.g., Ramteke and Srinivasan (2011). Although this approach is attractive in that it does not involve additional computational cost, the performance of the search is hardly improved by this modification, as penalties generally do not provide direct information on promising search directions.

We conclude that heuristic approaches are either problem-specific or designed for relatively simple production processes. They become less efficient in the presence of complex technological constraints because it is already difficult to devise feasible schedules. In addition, the quality of the solutions cannot be controlled systematically.

3.3. Hybrid methods: Overview

The use of exact methods for solving large-scale scheduling problems with complex constraints has become more attractive with the development of hybrid methods such as model-reduction methods, aggregation techniques, and decomposition methods. The key idea of these methods is to exploit the flexibility provided by mixed-integer linear programming to easily accommodate complex technological constraints, while limiting the number of simultaneous decisions to achieve reasonable CPU times. In the following, we sketch these three types of hybrid methods; for our approach we apply a decomposition method. Therefore, we will review related decomposition methods in more detail in Subsection 3.4.

Model reduction methods reduce the dimensionality of the problem by fixing a large number of the decision variables of the original MILP model so that only critical decisions are to be taken. Méndez and Cerdá (2002b) use preordering rules to reduce the size of a process-specific MILP model. Similarly, Günther et al. (2006) apply the block-planning technique, in which the sequence of batches within a block is determined in advance based on an analysis of the sequence-dependent changeover times. Bilgen and Günther (2010) extend this work to account for the transportation between the plants and distribution centres. For problems with a bottleneck stage, Marchetti and Cerdá (2009) propose to replace the individual general-precedence sequencing variables for each stage by sequencing variables that define a unique ordering of the batches for all stages.

Aggregation techniques replace groups of related decision variables by aggregate variables. Wilkinson et al. (1995) propose a temporal aggregation technique for discrete-time models in which short time periods are aggregated into longer time periods. Dimitriadis et al. (1997) separate the planning horizon into two time intervals; the first interval is modelled in detail, and the second interval is modelled using the aggregation technique proposed by Wilkinson et al. (1995).

Decomposition methods divide the problem at hand into several smaller subproblems that can be solved within short CPU times. Harjunkoski and Grossmann (2002) decompose single- and multi-stage scheduling problems into an assignment and a scheduling subproblem, which are solved using mixed-integer linear programming and constraint programming, respectively. Maravelias (2006) proposes a similar decomposition strategy for a problem in which the sequencing subproblems can be solved efficiently. For a review of decomposition methods that are more closely related to the method proposed in the present paper, we refer the reader to the next subsection.

3.4. Hybrid iterative scheduling methods

In this section, we review hybrid decomposition methods that iteratively schedule groups of one or more batches. We focus on those methods that are methodologically related to our approach in the sense that an MILP formulation is used to solve the resulting scheduling problems.

Roslöf et al. (2001) address a rescheduling problem for a single-stage, single-unit production process. They propose an MILP formulation for inserting a given group of batches into a given schedule; the relative order of the batches in that schedule is maintained by a set of (direct and indirect) precedence constraints. Méndez and Cerdá (2003a) propose a similar approach for the rescheduling of a single-stage production process with parallel units. Roslöf et al. (2002) adapt the approach presented in Roslöf et al. (2001) to the problem of generating an initial schedule. In each iteration, new batches are added to the current partial schedule using an MILP formulation, and the relative order of the batches that have already been scheduled is thereby maintained, in a similar fashion to the method of Roslöf et al. (2001). The proposed approach is applied to an industrial paper-converting process. In contrast to the present paper, the three papers mentioned above report computational results for relatively small problem instances.

The technique for maintaining the relative order of the batches that was proposed in the papers cited above cannot be extended straightforwardly to multi-stage processes with more complex structures. Instead, various authors have proposed to maintain the relative order by fixing the values of the corresponding decision variables while maintaining the corresponding variables and constraints in the MILP model. However, the preprocessing algorithms of commercial solvers may not eliminate all redundant variables and constraints, resulting in models that are considerably larger than necessary. In Section 6, we will provide computational results for such a variable-fixing strategy showing that the preprocessing procedure of the Gurobi solver does not always eliminate all

variables and constraints that become redundant when values of decision variables are fixed. Castro et al. (2009) consider a multi-stage, multi-product batch process with parallel, non-identical processing units for each stage, sequence-independent changeover times and unlimited intermediate storage space. The groups to be scheduled iteratively are computed and sorted using some simple priority rules. After all of the groups have been scheduled, the groups are iteratively rescheduled in the same order. For the scheduling of the individual groups, both a batch-based and a network-based MILP model are proposed, but the computational cost of the latter turns out to be significantly higher. Kopanos et al. (2010) consider the same type of production process, but additionally account for sequence-dependent changeover times. In an experimental analysis of the effect of group sizes, the best results were obtained when only a single batch was scheduled per iteration. The method of Kopanos et al. (2010) comprises a constructive step and an improvement step. In the constructive step, the batches are ordered according to the number of suitable processing units; for the scheduling of the batches, two alternative batch-based MILP formulations are proposed. The improvement step consists of a reordering and a reinsertion stage. In the reordering stage, an improvement is attempted by swapping the order of batches that are processed consecutively. In the reinsertion stage, single batches are rescheduled. Both stages are repeated until no improvement is achieved within a predefined number of iterations. The computational results are presented for a pharmaceutical production process with six stages, 17 processing units and up to 336 operations. Kopanos et al. (2012) and Aguirre et al. (2012) apply a similar solution method to an ice-cream production process of the make-and-pack type and a multi-stage, single-unit production process in the semiconductor manufacturing industry, respectively. For a production process with identical parallel processing units but without changeover times, Gomes et al. (2010) propose to combine a similar strategy with a discrete-time network-based model for inserting additional batches into an existing schedule.

4. Solution strategy of the hybrid method

In this section, we present the solution strategy of our hybrid method for large-scale short-term scheduling of make-and-pack production processes. The method consists of three phases: decomposition, construction, and improvement. In the decomposition phase, the set of all batches is decomposed into groups of a predefined size such that (a) batches of the same group can form a feasible partial schedule and (b) the groups of batches can be scheduled independently of one another. In the construction phase, an initial feasible schedule is generated by iteratively scheduling the groups of batches. Similar to the related papers mentioned in Subsection 3.4, we apply a batch-based MILP formulation to the solution of the scheduling problem in each iteration. However, in contrast to the methods of Castro et al. (2009) and Kopanos et al. (2010), we eliminate a large number of the variables and constraints that have become redundant due to the decisions taken in previous iterations. For large-scale instances, this elimination technique reduces the overall computational cost considerably. In the

Table 1: Data of the illustrative example

Product	Demand	Intermediate	Package	Pack batch
P_1	10	I_1	K_1	1,2
P_2	10	I_2	K_2	3,4
P_3	10	I_3	K_2	5,6
P_4	20	I_3	K_3	7,8,9,10
P_5	10	I_4	K_4	11,12

improvement phase, a local search heuristic is applied to the initial schedule. In each iteration of this heuristic, we identify a group of critical batches by solving two linear programs derived directly from the batch-based MILP formulation of the construction phase. This group of batches is then removed from the current schedule and reinserted, again by applying an appropriate MILP formulation.

The remainder of this section is organised as follows. In Subsection 4.1, we introduce an example that will be used to illustrate the individual phases of our hybrid method. In Subsections 4.2, 4.3, and 4.4, we describe the decomposition, construction, and improvement phases in detail.

A preliminary version of the decomposition and construction phases can be found in Baumann and Trautmann (2011). In the present paper, we provide an enhanced version of these phases that requires less computational effort and provides superior results.

4.1. Illustrative example

To illustrate the individual phases of the hybrid method, we use an example introduced by Baumann and Trautmann (2013). In this subsection, we provide the main data of this example.

The demand for five products (P_1–P_5) that must be produced from four different intermediates (I_1–I_4) is given. The production facility consists of one premix unit (PM1), which is connected to two final-mix units (FM1, FM2), three storage tanks (S1, S2, S3) and two packing lines (PL1, PL2). Table 1 lists the demand, intermediate, package type, and index of the pack batches for each product. The complete data of the example can be found in Baumann and Trautmann (2013). Given the predetermined sizes of the make batches (10 units) and pack batches (5 units), the total number of required make and pack batches can be derived directly from the total demand. In this case, 12 pack batches and six make batches must be scheduled to meet the total demand of 60 units.

4.2. Decomposition phase

The size of an instance of the problem discussed in the present paper is driven primarily by the number of batches to be scheduled. We therefore propose to decompose the set of batches into smaller groups as follows.

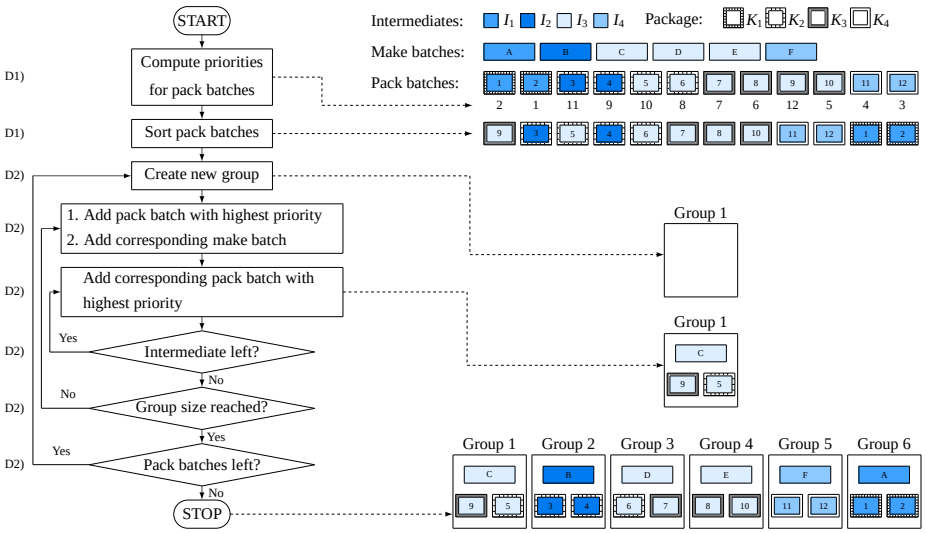

Figure 3: Flowchart of the decomposition phase

D1) A priority value is computed for each pack batch. The pack batches are then sorted according to these values. In preliminary tests, we have analysed various priority rules, such as the number of suitable packing lines, the wash-out family, or the package type. However, we have obtained the best results using a random number for the priority value, possibly because the order of the batches in the resulting schedule may be quite different from the order in which the batches are scheduled.

D2) The batches are assigned to groups based on the order determined in step D1). The number of make batches within a group must be defined as an input parameter to the hybrid method. The assignment of make and pack batches to groups proceeds as follows. First, a pack batch with the highest priority value that has not yet been assigned is added to the current group. Second, a make batch that produces the required intermediate and has not yet been assigned is added. Third, further pack batches that have not yet been assigned and require the same intermediate are added in order of non-increasing priority values, such that the entire quantity of the intermediate provided by the make batch is consumed by these pack batches. These three steps are repeated until the specified group size has been reached. In Figure 3, the set of batches is decomposed into six groups of size one.

Figure 3 provides a flowchart of the decomposition phase and visualisation of each step of this phase for the illustrative example.

4.3. Construction phase

In the construction phase, an initial feasible schedule is generated as follows.

C1) The first or next group of batches is scheduled by solving MILP model (C) (cf. Subsection 5.5). The main decision variables of this model are (a) allocation variables for new batches and (b) general-precedence variables among new batches and between new and scheduled batches. The unit allocation and relative ordering of the scheduled batches is preserved by imposing timing constraints on the scheduled batches that are processed consecutively. In each iteration, a new instance of the MILP model is generated based on updated sets of new and already-scheduled batches. Using this strategy, the size of the MILP model grows only slowly from iteration to iteration because a large number of constraints and variables that have become redundant are eliminated.

C2) From the resulting (partial) schedule, the final values of the allocation and general-precedence variables are retrieved and translated into immediate-precedence relationships (cf. Algorithm 1 to 4, see Appendix). This information is used to set up model (C) in the next execution of step C1).

C3) The construction phase terminates when all groups have been scheduled. Otherwise, the next group of batches is selected and scheduled (step C1).

Figure 4 provides a flowchart of the construction phase and visualisation of each step of this phase for the illustrative example. In each schedule, new operations are marked with a triangle.

4.4. Improvement phase

Once the construction phase is complete, an iterative improvement procedure is applied to the initial schedule. This algorithm proceeds as follows. At the beginning of the procedure, all make batches are unmarked. During the procedure, a make batch is temporarily marked if removing and reinserting the related operations has not resulted in an improvement of the makespan.

I1) All critical operations of the current schedule are identified by solving two LP models. First, model (E) (cf. Subsection 5.5) is solved to compute the earliest possible start times of the premix, final-mix and packing operations, given the unit allocation and order of the operations in the current schedule. Second, model (L) (cf. Subsection 5.5) is solved to compute the latest possible start times of the operations, given the unit allocation, order of the operations in the current schedule, and makespan of the current schedule. Both models can be solved efficiently because they contain only continuous decision variables. Operations are considered critical when their earliest and latest start times coincide, i.e., when a delay of the operation causes an increase in the makespan. In other words, the makespan is determined by the set of critical operations. In the following, we consider a make batch as critical if at least one of its related operations (premix, final-mix, storage or assigned packing operation(s)) is critical.

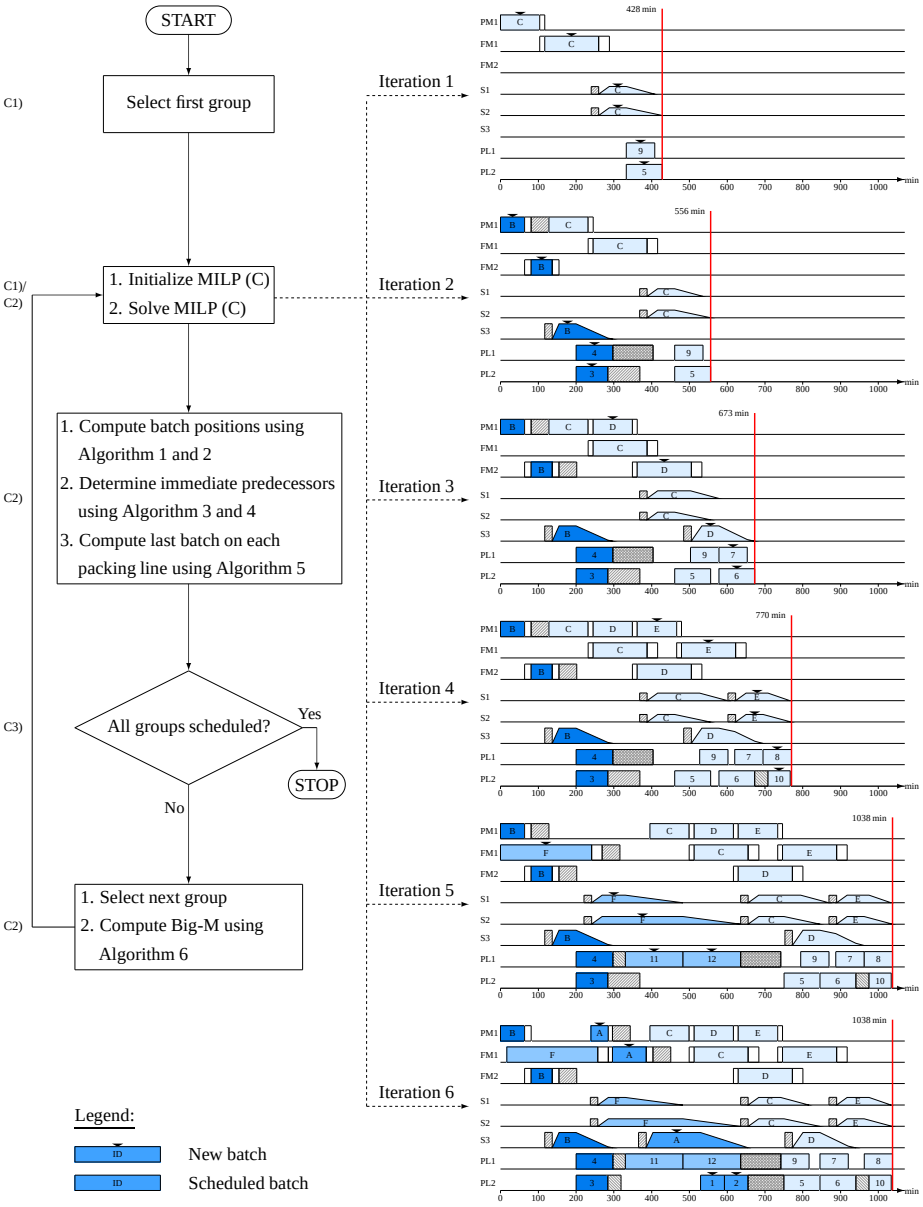

Figure 4: Flowchart of the construction phase

I2) We determine the set of make batches that are critical and unmarked. If this set is empty, then the procedure is terminated. Otherwise, we randomly select and mark one of these batches.

I3) All operations related to the make batch selected in step I2) are unscheduled. The immediate-precedence relations are then updated using Algorithm 7 (cf. Appendix). Finally, the unscheduled operations are reinserted by solving MILP model (I) (cf. Subsection 5.5).

I4) If the makespan can be improved, then we unmark all make batches and proceed with set I1). Otherwise, the procedure directly continues with step I1).

Note that in each iteration, the unscheduled operations may be reinserted at the same position; therefore, the makespan does not deteriorate during the improvement phase. Figure 5 provides a flowchart of the improvement phase and visualisation of each step of this phase for the illustrative example and the initial schedule shown in Figure 4. The critical operations in the respective schedules are accentuated by a thick border, and the operations selected for removal are marked with a cross.

5. Scheduling models used in the hybrid method

The scheduling models used in the construction and improvement phases of the hybrid method are derived from the model presented in Baumann and Trautmann (2013). The model of Baumann and Trautmann (2013) can be applied to production processes such as presented in Honkomp et al. (2000) and therefore covers a wide range of operating conditions that are typical for make-and-pack plants.

The main difference of the model proposed in the present paper is that we divide the set of all batches into two sets, I and N. Set I consists of the batches that have already been scheduled in an earlier iteration, and set N consists of the batches that are to be scheduled in the current iteration. This distinction gives rise to three different groups of constraints. The first group of constraints involves only new batches. The second group of constraints involves both new and scheduled batches, and the third group involves only scheduled batches. In each iteration of the construction and improvement phases, the sets of new and already-scheduled batches and their parameters are initialised, and a new instance of the model is generated.

In Subsections 5.1, 5.2, and 5.3, we provide a detailed description of these three groups of constraints. In Subsection 5.4, we introduce the objective function used in the model for the construction phase. In Subsection 5.5, we summarise the various MILP and LP models used for the construction and improvement phase.

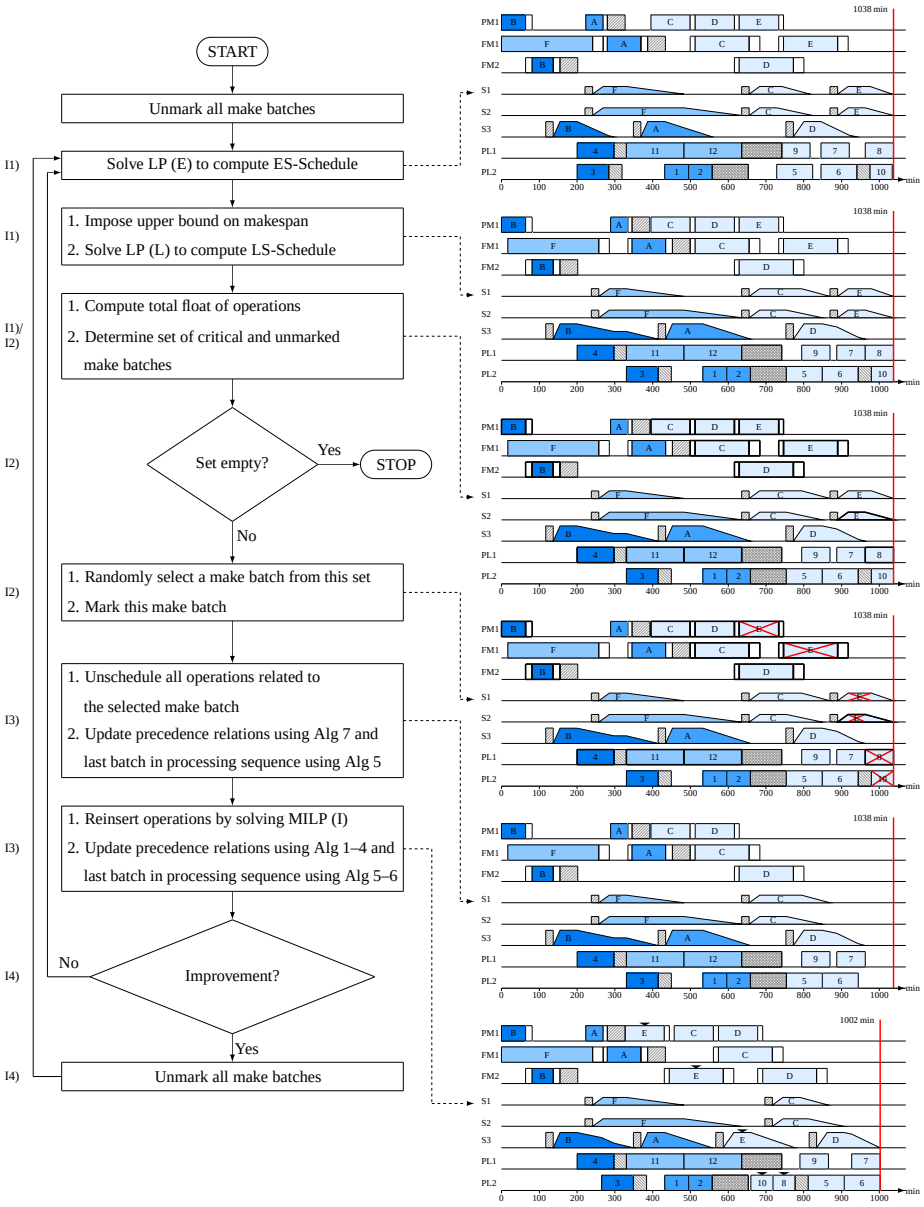

Figure 5: Flowchart of the improvement phase

Notation

Sets

I	Scheduled batches
N	New batches
K	Tasks {PM, FM, S, P}
J	Units
G	Groups of identical batches
I^M	Scheduled make batches
N^M	New make batches
I^P	Scheduled pack batches
N^P	New pack batches
I^k	Scheduled batches that require task $k \in K$
I^k_j	Scheduled batches whose task $k \in K$ is performed on unit/tank $j \in J^k$
N^k	New batches that require task $k \in K$
N^k_j	New batches that require task $k \in K$ and can be assigned to unit $j \in J^k$
N^k_g	New batches that require task k and belong to group $g \in G$.
I^M_i	Scheduled make batches that supply scheduled pack batch $i \in I^P$
N^M_i	New make batches that can supply new pack batch $i \in N^P$
I^P_i	Scheduled pack batches that are supplied by scheduled make batch $i \in I^M$
N^P_i	New pack batches that can be supplied by new make batch $i \in N^M$
J^k	Units that perform task $k \in K$
J^k_i	Assigned unit for scheduled batch $i \in I^k$ and available units for new batch $i \in N^k$ ($k \in K$)
J^{FM}_j	Final-mix units that can be connected to premix unit $j \in J^{PM}$
SUC^k_i	Immediate successor(s) of batch $i \in I^k$ ($k \in K$)
$LAST^P_j$	Last pack batch in the processing sequence of unit $j \in J^P$

Parameters

α^k_i	Duration of task $k \in \{PM, FM\}$ for batch $i \in I^k \cup N^k$
α^k_{ij}	Duration of task $k \in K \setminus \{S\}$ for batch $i \in I^k \cup N^k$ on unit $j \in J^k_i$ ($= \alpha^k_i$ for $k \in \{PM, FM\}$)
δ_i	Quarantine time of make batch $i \in I^M \cup N^k$
τ_i	Transfer time of make batch $i \in I^{PM} \cup N^{PM}$
ρ_i	Pump out time of make batch $i \in I^M \cup N^M$
$\omega^k_{ii'j}$	Changeover time between batch $i \in I^k \cup N^k$ and $i' \in I^k \cup N^k$ on unit $j \in J^k_i \cap J^k_{i'}$
c_j	Capacity of tank $j \in J^S$
β^M	Size of make batch $i \in I^M \cup N^M$ (batch independent)
β^P	Size of pack batch $i \in I^P \cup N^P$ (batch independent)
p^k_i	Position of task $k \in K \setminus \{S\}$ of batch $i \in I^k \cup N^k$ in processing sequence
$p^{prev,k}_i$	Value of p^k_i in previous iteration; in the first iteration, $p^{prev,k}_i = 1$
p^S_{ij}	Position of storage task of batch $i \in I^M \cup N^M$ in tank j in storing sequence
$p^{prev,S}_{ij}$	Value of p^S_{ij} in previous iteration; in the first iteration, $p^{prev,S}_{ij} = 1$
C^{prev}	Makespan of previous iteration; in the first iteration, $C^{prev} = 0$
C^{prev}_j	Makespan of packing line $j \in J^P$ of previous iteration; in the first iteration, $C^{prev}_j = 0$
M	Sufficiently large number

Continuous variables (all non-negative)

S^k_i	Start time of task $k \in K \setminus \{S\}$ of batch $i \in I^k \cup N^k$
$F_{ii'j}$	Amount of material that make batch $i \in N^M$ supplies to pack batch $i' \in N^P_i$ through tank $j \in J^S$
C	Makespan of production plan
C_j	Makespan of packing line $j \in J^P$

Binary variables

$U_{ii'}$	= 1, if make batch $i \in N^M$ supplies pack batch $i' \in N^P_i$; =0, otherwise
Y^k_{ij}	= 1, if batch $i \in N$ is assigned to unit $j \in J^k_i$ for task $k \in K$; =0, otherwise
$X^k_{ii'}$	$\begin{cases} = 1, & \text{if batch } i \in N^k \text{ is processed/stored before batch } i' \in I^k \cup N^k \text{ for task } k \in K \setminus \{S\} \\ = 0, & \text{if batch } i \in N^k \text{ is processed/stored after batch } i' \in I^k \cup N^k \text{ for task } k \in K \setminus \{S\} \end{cases}$

5.1. Constraints for new batches

In this subsection, we present the constraints related to the allocation (cf. Subsection 5.1.1), material flow (cf. Subsection 5.1.2), timing (cf. Subsection 5.1.3), sequencing (cf. Subsection 5.1.4), and symmetry-breaking decisions (cf. Subsection 5.1.5).

5.1.1. Allocation constraints

Constraints (1) allocate a suitable final-mix unit $j \in J_i^{FM}$ to the final-mix task of each new make batch $i \in N^{FM}$ and a suitable packing line $j \in J_i^P$ to the packing task of each new pack batch $i \in N^P$.

$$\sum_{j \in J_i^k} Y_{ij}^k = 1 \qquad (k \in \{FM, P\};\ i \in N^k) \tag{1}$$

For make batches $i \in N^{PM}$ requiring a premix task, constraints (2) ensure that the premix and final-mix tasks are allocated to a pair of premix and final-mix units $j \in J_i^{PM}, j' \in J_j^{FM}$ that can be connected to one another.

$$Y_{ij}^{PM} = \sum_{j' \in J_j^{FM}} Y_{ij'}^{FM} \qquad (i \in N^{PM};\ j \in J_i^{PM}) \tag{2}$$

Constraints (3) allocate one or two storage tanks $j \in J^S$ to every new make batch $i \in N^M$, such that the total capacity of the allocated tanks is equal to the size β^M of batch i.

$$\sum_{j \in J^S} c_j Y_{ij}^S = \beta^M \qquad (i \in N^M) \tag{3}$$

Constraints (4) allocate exactly one storage tank $j \in J^S$ to each new pack batch $i \in N^P$. The allocation of a single tank is sufficient because each tank has at least the capacity to store an entire pack batch of size β^P.

$$\sum_{j \in J^S} Y_{ij}^S = 1 \qquad (i \in N^P) \tag{4}$$

5.1.2. Material flow constraints

Material flow constraints ensure that each new make batch $i \in N^M$ supplies two new pack batches $i' \in N_i^P$ with material. We use the continuous variable $F_{ii'j}$ to denote the amount of material that make batch i delivers to pack batch i' through storage tank j. Constraints (5) ensure that every make batch i delivers all of its material β^M to some set of pack batches i'.

$$\sum_{i' \in N_i^P;\ j \in J^S} F_{ii'j} = \beta^M \qquad (i \in N^M) \tag{5}$$

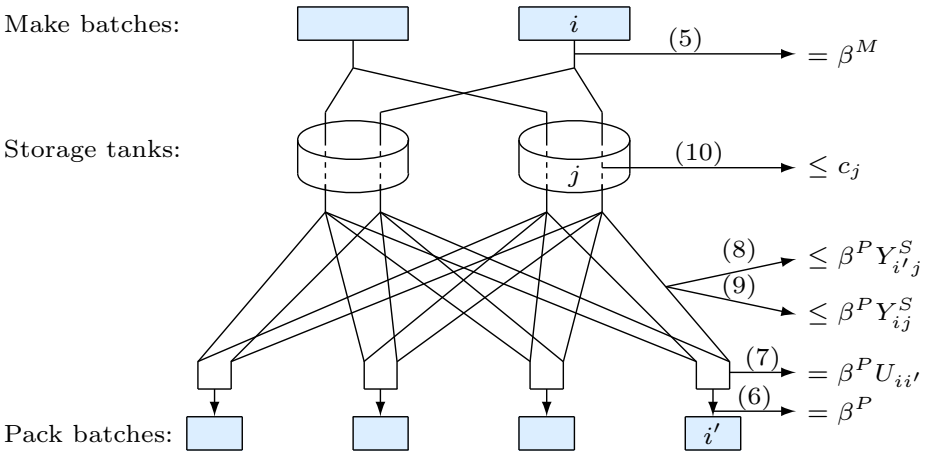

Figure 6: Illustration of constraints (5)–(10)

Similarly, constraints (6) guarantee that the total amount of material supplied to each pack batch $i' \in N^P$ equals β^P.

$$\sum_{i \in N^M_{i'};\ j \in J^S} F_{ii'j} = \beta^P \qquad (i' \in N^P) \tag{6}$$

Material can only flow between make batch i and pack batch i' if the binary variable $U_{ii'}$ equals one, as expressed by constraints (7).

$$\sum_{j \in J^S} F_{ii'j} = \beta^P U_{ii'} \qquad (i \in N^M;\ i' \in N^P_i) \tag{7}$$

Moreover, constraints (8) and (9) ensure that the same storage tank $j \in J^S$ is allocated to both make batch i and pack batch i' whenever i delivers material to i'.

$$F_{ii'j} \le \beta^P Y^S_{i'j} \qquad (i \in N^M;\ i' \in N^P_i;\ j \in J^S) \tag{8}$$

$$F_{ii'j} \le \beta^P Y^S_{ij} \qquad (i \in N^M;\ i' \in N^P_i;\ j \in J^S) \tag{9}$$

Constraints (10) prevent the capacity c_j of storage tank $j \in J^S$ from being exceeded.

$$\sum_{i' \in N^P_i} F_{ii'j} \le c_j \qquad (i \in N^M;\ j \in J^S) \tag{10}$$

Figure 6 illustrates the material-flow constraints for a situation with two identical make batches, two storage tanks, and four identical pack batches.

5.1.3. Timing constraints

Constraints (11) represent the no-wait restriction between the premix and final-mix task of each make batch $i \in N^{PM}$ requiring a premix task. The start

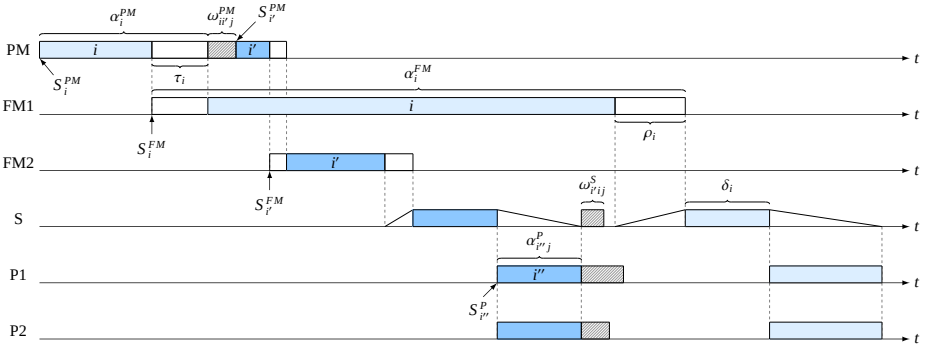

Figure 7: Timing variables and parameters

time of the final-mix task S_i^{FM} is computed such that the premix and final-mix tasks overlap for exactly the transfer time τ_i (see Figure 7).

$$S_i^{FM} = S_i^{PM} + \alpha_i^{PM} - \tau_i \qquad (i \in N^{PM}) \tag{11}$$

Constraints (12) ensure that the processing of a pack batch $i' \in N_i^P$ that consumes material from make batch $i \in N^M$ cannot start before the supplying make batch has been stored for at least its quarantine time δ_i.

$$S_{i'}^P \geq S_i^{FM} + \alpha_i^{FM} + \delta_i - M(1 - U_{ii'}) \qquad (i \in N^M;\ i' \in N_i^P) \tag{12}$$

5.1.4. Sequencing constraints

We use the general-precedence concept to derive the processing sequence of batches at each processing unit or storage tank. In this concept, one sequencing variable $X_{ii'}^k$ is defined for each pair of batches $i, i' \in N^k$ that can be allocated to the same processing unit or storage tank $j \in J_i^k \cap J_{i'}^k$. If batches i and i' are processed on different units, then constraints (13) and (14) both become redundant and the variable $X_{ii'}^k$ is meaningless for unit j (see final-mix tasks in Figure 7). Otherwise, if both batches are processed at the same unit ($Y_{ij}^k = Y_{i'j}^k = 1$), then either constraint (13) or constraint (14) becomes active. If $X_{ii'}^k = 1$, then constraint (13) becomes active and forces the start time $S_{i'}^k$ of batch i' to be greater than or equal to the completion time of batch i plus the duration $\omega_{ii'j}^k$ of the subsequent changeover (see premix tasks in Figure 7). Constraint (14) becomes redundant in that case.

$$S_{i'}^k \geq S_i^k + \alpha_{ij}^k + \omega_{ii'j}^k - M(1 - X_{ii'}^k) - M(2 - Y_{ij}^k - Y_{i'j}^k) \qquad (k \in K \setminus \{S\};\ i \in N^k;\ i' \in N^k;\ j \in J_i^k \cap J_{i'}^k :\ i < i') \tag{13}$$

In the opposite case, where $X^k_{ii'} = 0$, constraint (13) becomes redundant, and constraint (14) becomes active and ensures that batch i' is processed before batch i.

$$S^k_i \geq S^k_{i'} + \alpha^k_{i'j} + \omega^k_{i'ij} - MX^k_{ii'} - M(2 - Y^k_{ij} - Y^k_{i'j}) \\ (k \in K \setminus \{S\};\ i \in N^k;\ i' \in N^k;\ j \in J^k_i \cap J^k_{i'} :\ i < i') \quad (14)$$

A major advantage of the general-precedence concept is that we can use the final-mix sequencing variables $X^{FM}_{ii'}$ to sequence the storage operations without compromising the optimality of the solution. However, this strategy requires that we also define a variable $X^{FM}_{ii'}$ for pairs of make batches that do not share a common final-mix unit because they can still be allocated to the same storage tank. Again, we define two sets of sequencing constraints, constraints (15) and (16), for each pair of new make batches $i, i' \in N^M$ and each storage tank $j \in J^S$. Whenever i and i' are allocated to the same tank, the variable $X^{FM}_{ii'}$ defines their relative storage sequence. If $X^{FM}_{ii'} = 1$, then constraint (15) ensures that the storage task of make batch i' begins after both the storage task of batch i and the subsequent changeover of duration $\omega^S_{ii'j}$ are complete. Due to the no-wait restriction between the final-mix task and storage task of batch i, we derive the start time of the storage task directly from the start time of the corresponding final-mix task. As illustrated in Figure 7, the start time of the storage task is equal to the start time of the final-mix task S^{FM}_i plus its duration α^{FM}_{ij} minus the pump out time ρ_i. Similarly, the completion time of the storage task of batch i is derived from the completion time of the corresponding packing task, which is equal to the start time $S^P_{i''}$ plus the unit-dependent packing time $\alpha^P_{i''j'}$. Because we do not know a priori which pack batch i'' will be supplied by make batch i, we must impose constraints (15) and (16) for all pack batches $i'' \in N^P_i$ that could be supplied by make batch i. However, the constraints are relaxed if make batch i does not supply pack batch i'' ($U_{ii''} = 0$). Moreover, the constraints are only active if both make batch i and pack batch i'' are allocated to the same storage tank $j \in J^S$. Constraint (16) holds if $X^{FM}_{ii'} = 0$.

$$S^{FM}_{i'} + \alpha^{FM}_{i'} - \rho_{i'} \geq S^P_{i''} + \sum_{j' \in J^P_{i''}} \alpha^P_{i''j'} Y^P_{i''j'} + \omega^S_{ii'j} \\ - M(1 - X^{FM}_{ii'}) - M(2 - Y^S_{i'j} - Y^S_{i''j}) - M(1 - U_{ii''}) \\ (i \in N^{FM};\ i' \in N^{FM};\ i'' \in N^P_i;\ j \in J^S :\ i < i') \quad (15)$$

$$S^{FM}_i + \alpha^{FM}_i - \rho_i \geq S^P_{i''} + \sum_{j' \in J^P_{i''}} \alpha^P_{i''j'} Y^P_{i''j'} + \omega^S_{i'ij} \\ - MX^{FM}_{ii'} - M(2 - Y^S_{ij} - Y^S_{i''j}) - M(1 - U_{i'i''}) \\ (i \in N^{FM};\ i' \in N^{FM};\ i'' \in N^P_{i'};\ j \in J^S :\ i < i') \quad (16)$$

In other production processes (cf., e.g.,, Bongers and Bakker 2006), the intermediate is continuously pumped into the storage tank as soon as the mixing task starts. With some minor modifications to constraints (15) and (16), namely the substitution of $\alpha_{i'}^{FM}$ - $\rho_{i'}$ with $\tau_{i'}$ and the substitution of α_{i}^{FM} - ρ_{i} with τ_{i}, respectively, the model could also account for continuous material flow on the make stage.

5.1.5. Symmetry-breaking constraints

In a given schedule, identical batches, i.e., batches that produce the same intermediate or product, can be interchanged with no impact on value of the objective function. The elimination of such symmetries generally accelerates the solution algorithm. Constraint (17) removes these symmetries by imposing an arbitrary sequence for each group $g \in G$ of identical new batches and each task $k \in K \setminus \{S\}$ required by these batches.

$$S_i^k \leq S_{i'}^k \qquad (g \in G;\ k \in K \setminus \{S\};\ i \in N_g^k;\ i' \in N_g^k :\ i < i') \tag{17}$$

5.2. Constraints on new and scheduled batches

This group of constraints is concerned with the proper insertion of new batches among scheduled ones. Whenever a task $k \in \{PM, FM, P\}$ of a new batch i is allocated to a unit designated to process a previously scheduled batch i', the variable $X_{ii'}^k$ defines the processing order of i and i'. If $X_{ii'}^k = 1$, then constraint (18) requires that the new batch i delays the previously scheduled batch i'. The constraint is relaxed if batch i is not allocated to the same processing unit as batch i'.

$$S_{i'}^k \geq S_i^k + \alpha_{ij}^k + \omega_{ii'j}^k - M(1 - X_{ii'}^k) - M(1 - Y_{ij}^k) \\ (k \in K \setminus \{S\};\ i \in N^k;\ i' \in I^k;\ j \in J_i^k \cap J_{i'}^k) \tag{18}$$

In the opposite case, where $X_{ii'}^k = 0$, constraint (19) ensures that batch i is processed after batch i'.

$$S_i^k \geq S_{i'}^k + \alpha_{i'j}^k + \omega_{i'ij}^k - MX_{ii'}^k - M(1 - Y_{ij}^k) \\ (k \in K \setminus \{S\};\ i \in N^k;\ i' \in I^k;\ j \in J_i^k \cap J_{i'}^k) \tag{19}$$

Note that for the premix and packing tasks, the variable $X_{ii'}^k$ is only defined when the processing unit $j \in J_{i'}^k$ that is allocated to the scheduled batch i' is among the available processing units $j \in J_i^k$ for the new batch i.

Whenever a new make batch i is allocated to a storage tank designated to process a previously scheduled make batch i', constraints (20) and (21) ensure that the respective storage tasks do not overlap. If $X_{ii'}^{FM} = 1$, then constraint (20) ensures that the storage task of batch i' begins after both the storage task of batch i and the subsequent changeover are complete. Recall that the completion time of the storage task of batch i is determined by the completion time of the latest pack batch $i'' \in N_i^P$ that consumes material from make batch

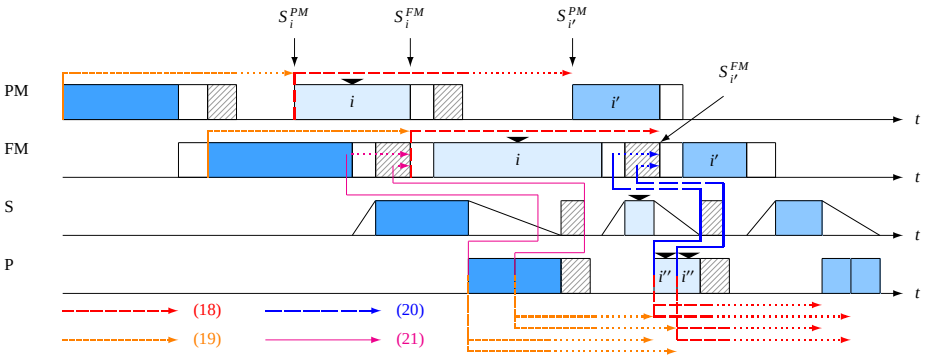

Figure 8: Illustration of constraints (18)–(21)

i. Constraint (20) is only active if pack batch i'' is allocated to storage tank $j \in J_{i'}^S$ and make batch i supplies pack batch i''.

$$S_{i'}^{FM} + \alpha_{i'}^{FM} - \rho_{i'} \geq S_{i''}^{P} + \sum_{j' \in J_{i''}^P} \alpha_{i''j'}^{P} Y_{i''j'}^{P} + \omega_{ii'j}^{S} - M(1 - X_{ii'}^{FM}) - M(1 - Y_{i''j}^{S}) - M(1 - U_{ii''}) \quad (i \in N^{FM};\ i' \in I^{FM};\ i'' \in N_i^P;\ j \in J_{i'}^S) \quad (20)$$

Constraint (21) is enforced if make batch i' is stored before make batch i ($X_{ii'}^{FM} = 0$).

$$S_{i}^{FM} + \alpha_{i}^{FM} - \rho_{i} \geq S_{i''}^{P} + \alpha_{i''j'}^{P} + \omega_{i'ij}^{S} - M X_{ii'}^{FM} - M(1 - Y_{ij}^{S}) \quad (i \in N^{FM};\ i' \in I^{FM};\ i'' \in I_{i'}^P;\ j' \in J_{i''}^P;\ j \in J_{i''}^S) \quad (21)$$

Figure 8 provides a visualisation of constraints of type (18)–(21) in a situation in which a group of one new make batch i and two new pack batches i'' are scheduled between previously scheduled batches.

5.3. Constraints on scheduled batches

The third group of constraints models the timing relations among scheduled batches. Constraints (22) ensure that the processing of the immediate successor $i' \in SUC_i^k$ of batch i cannot begin before the processing of batch i and subsequent changeover are complete. These precedence relationships are established for the premix, final-mix and packing tasks.

$$S_{i'}^{k} \geq S_{i}^{k} + \alpha_{ij}^{k} + \omega_{ii'j}^{k} \quad (k \in K \setminus \{S\};\ i \in I^k;\ j \in J_i^k;\ i' \in SUC_i^k) \quad (22)$$

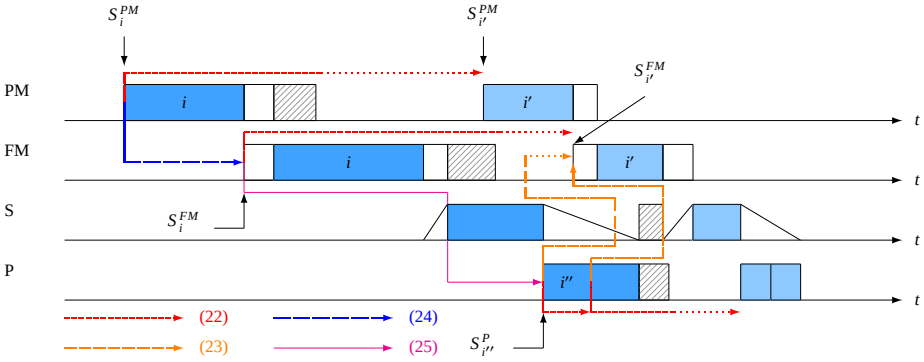

Figure 9: Illustration of constraints (22)–(25)

The precedence relationships between storage tasks are expressed by constraints (23). If make batch i' is stored immediately after make batch i in tank $j \in J^S_{i'}$, constraint (23) establishes a precedence relationship between each pack batch i'' that consumes material from make batch i through storage tank $j \in J^S_{i'}$ and make batch i'.

$$S^{FM}_{i'} + \alpha^{FM}_{i'} - \rho_{i'} \geq S^P_{i''} + \alpha^P_{i''j'} + \omega^S_{ii'j} \\ (i \in I^{FM} : \; i' \in SUC^S_i; \; i'' \in I^P_i; \; j \in J^S_{i'} \cap J^S_{i''}; \; j' \in J^P_{i''}) \quad (23)$$

Constraints (24) represent the no-wait restrictions between the premix and final-mix tasks of each scheduled make batch $i \in I^{PM}$ that requires a premix task.

$$S^{FM}_i = S^{PM}_i + \alpha^{PM}_i - \tau_i \qquad (i \in I^{PM}) \quad (24)$$

For each scheduled make batch $i \in I^M$, constraints (25) guarantee that each consuming pack batch $i' \in I^P_i$ does not start before make batch i has been stored for at least its quarantine time δ_i.

$$S^P_{i'} \geq S^{FM}_i + \alpha^{FM}_i + \delta_i \qquad (i \in I^M; \; i' \in I^P_i) \quad (25)$$

Figure 9 illustrates all active constraints of type (22)–(25) that involve make batch i and its two corresponding pack batches, i''.

5.4. Modified objective function

Two groups of constraints are used to compute the makespan C of the production schedule. Constraints (26) ensure that the makespan C is greater than or equal to the completion time of every new pack batch $i \in N^P$.

$$C \geq S^P_i + \sum_{j \in J^P_i} \alpha^P_{ij} Y^P_{ij} \qquad (i \in N^P) \quad (26)$$

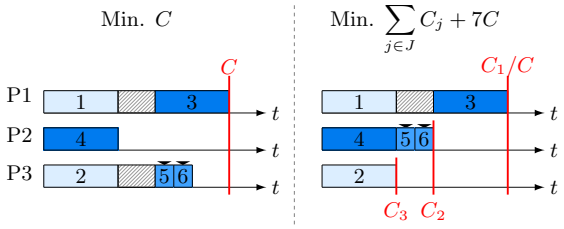

Figure 10: Illustration of modified objective function

Constraints (27) ensure that the makespan C is greater than or equal to the completion time of the last scheduled batch in the processing sequence of every packing line j.

$$C \geq S_i^P + \alpha_{ij}^P \qquad (j \in J^P;\ i \in LAST_j^P) \tag{27}$$

The minimisation of the overall makespan, C, of a partial schedule may result in adverse planning decisions. If, for example, the makespan of a partial schedule is driven by one packing line only, then new batches may be allocated to any of the less occupied packing lines, irrespective of unit-specific processing times or sequence-dependent changeovers. We therefore propose to compute a separate makespan C_j for each packing line $j \in J^P$ that satisfies constraints (28) and (29) in addition to the overall makespan, which is subject to constraint (30).

$$C_j \geq S_i^P + \alpha_{ij}^P \qquad (j \in J^P;\ i \in LAST_j^P) \tag{28}$$

$$C_j \geq S_i^P + \alpha_{ij}^P - M(1 - Y_{ij}^P) \qquad (i \in N^P;\ j \in J_i^P) \tag{29}$$

$$C \geq C_j \qquad (j \in J^P) \tag{30}$$

We then minimise the weighted sum of all makespans. Here, the weight of each separate makespan C_j is one, and the weight of the overall makespan C is seven, which corresponds to the number of packing lines in the production process described by Honkomp et al. (2000). This objective function favours schedules in which new batches do not extend the overall makespan and changeovers and long processing times on non-critical packing lines are avoided. Figure 10 shows an example, in which the extended objective function leads to a more promising partial schedule. For simplicity, only the packing lines are pictured. In this example, pack batches one to four have been scheduled in a previous iteration, and pack batches five and six must be scheduled in the current iteration. The overall makespan C is driven by packing line one. Both schedules shown in Figure 10 are optimal with respect to the overall makespan. The extended objective function favours the schedule on the right, which does not require an additional changeover.

The following two constraints are aimed at tightening the makespan constraints. During the construction phase, the overall makespan of the current iteration must be greater than or equal to the makespan obtained in the previ-

ous iteration, C^{prev}.

$$C \geq C^{prev} \tag{31}$$

The separate makespans can be tightened in a similar manner.

$$C_j \geq C_j^{prev} \qquad (j \in J^P) \tag{32}$$

5.5. Optimisation models

In this subsection, we summarise all models that are used in the proposed hybrid method.

The MILP model that is used to construct an initial schedule reads as follows.

$$\text{(C)} \begin{cases} \text{Min. } 7C + \displaystyle\sum_{j \in J^P} C_j \\ \text{s.t.} \quad (1)\text{–}(25), (28)\text{–}(32) \\ \qquad S_i^k \geq 0 \qquad (k \in K \setminus \{S\};\ i \in N^k \cup I^k) \\ \qquad U_{ii'} \in \{0,1\} \qquad (i \in N^M;\ i' \in N_i^P) \\ \qquad Y_{ij}^k \in \{0,1\} \qquad (k \in K;\ i \in N^k;\ j \in J_i^k) \\ \qquad X_{ii'}^k \in \{0,1\} \qquad (k \in \{PM, P\};\ i \in N^k; \\ \qquad\quad i' \in \{\{N^k : i < i'\} \cup I^k\} : J_i^k \cap J_{i'}^k \neq \varnothing) \\ \qquad X_{ii'}^{FM} \in \{0,1\} \qquad (i \in N^{FM}; \\ \qquad\qquad\qquad i' \in \{N^{FM} : i < i'\} \cup I^{FM}) \end{cases}$$

In the improvement phase, the earliest possible start times of all operations are computed by solving optimisation problem (E).

$$\text{(E)} \begin{cases} \text{Min. } C + \displaystyle\sum_{k \in K \setminus \{S\};\ i \in I^k \cup N^k} S_i^k \\ \text{s.t.} \quad (22)\text{–}(25), (27) \\ \qquad S_i^k \geq 0 \qquad (k \in K \setminus \{S\};\ i \in N^k \cup I^k) \end{cases}$$

The makespan $\overline{C}$ in the optimal solution to problem (E) is then used as an upper bound in computing the latest possible start times of all operations. The latest start times are computed by solving problem (L).

$$\text{(L)} \begin{cases} \text{Max.} \displaystyle\sum_{k \in K \setminus \{S\};\ i \in I^k \cup N^k} S_i^k \\ \text{s.t.} \quad (22)\text{–}(25), (27) \\ \qquad S_i^k \geq 0 \qquad (k \in K \setminus \{S\};\ i \in N^k \cup I^k) \\ \qquad C \leq \overline{C} \end{cases}$$

In each iteration of the improvement phase, we reinsert the unscheduled opera-

tions by solving problem (I).

$$
\text{(I)} \begin{cases} \text{Min. } C \\ \text{s.t. } (1)\text{–}(27) \\ \quad S_i^k \geq 0 \qquad (k \in K \setminus \{S\};\ i \in N^k \cup I^k) \\ \quad U_{ii'} \in \{0,1\} \qquad (i \in N^M;\ i' \in N_i^P) \\ \quad Y_{ij}^k \in \{0,1\} \qquad (k \in K;\ i \in N^k;\ j \in J_i^k) \\ \quad X_{ii'}^k \in \{0,1\} \qquad (k \in \{PM, P\};\ i \in N^k; \\ \qquad i' \in \{\{N^k : i < i'\} \cup I^k\} : J_i^k \cap J_{i'}^k \neq \varnothing) \\ \quad X_{ii'}^{FM} \in \{0,1\} \qquad (i \in N^{FM}; \\ \qquad\qquad i' \in \{N^{FM} : i < i'\} \cup I^{FM}) \end{cases}
$$

6. Computational results

We implemented the proposed hybrid method in Ansi-C and used the Gurobi Optimizer 5.0.2 to solve the MILP and LP models. All computations were performed on an HP Z820 workstation with two Intel Xeon CPU E5-2687W processors and 128 GB RAM. For the construction phase of the heuristic, we set a CPU time limit of 5 seconds per iteration for the Gurobi Optimizer; for the improvement phase, no CPU time limit was set. Similar to Kopanos et al. (2010), who conclude that larger group sizes do not guarantee better schedules but require more CPU time, we chose for both phases a group size of one make batch. Moreover, we applied the preprocessing methodology described in Baumann and Trautmann (2013) in both phases to exclude certain matchings between make batches and pack batches without loss of generality.

For our analysis we used two test sets, which we will refer to as set I and set II. Set I consists of the 20 small- and medium-sized instances proposed in Baumann and Trautmann (2013) with up to 234 operations to be scheduled. Set II consists of 10 large-sized instances provided by The Procter & Gamble Company (cf. Honkomp et al. 2000) with up to 1391 operations to be scheduled. In the second column of Tables 2 and 3, we list for each instance the corresponding number of operations to be scheduled.

For set I, we applied the hybrid heuristic with two different configurations. First, we used the 287 priority rules proposed by Fündeling and Trautmann (2006) to determine the assignment of batches to groups in the decomposition phase. For each of these rules, we applied the hybrid heuristic to each instance. Second, we applied the hybrid heuristic 100 times to each instance, each time using different random numbers to determine the assignment of batches to groups in the decomposition phase.

The results for both configurations are presented in Table 2. In columns seven and eight, we provide for each instance from all 287 runs with priority rules the best makespan that was obtained after the construction and after the improvement phase. In column nine, we state the total amount of CPU time required by the 287 runs. In columns ten to twelve and thirteen to fifteen, we provide the same results after 4 runs with random numbers and after 100 runs with random numbers, respectively.

We compare our results to the preliminary version of the hybrid method of Baumann and Trautmann (2011) and to the mixed-integer programming approach of Baumann and Trautmann (2013). We apply the preliminary version of Baumann and Trautmann (2011) 200 times to each instance, each time using random numbers to determine the assignment of batches to groups. The best makespan found for each instance is reported in column five of Table 2; note that the preliminary version does not include an improvement phase. Baumann and Trautmann (2013) propose 6 different model formulations; in the third and fourth column of Table 2, we list for each instance the makespan obtained by the formulation that found the largest number of feasible solutions and the corresponding CPU time requirements. The entry lim means that the MILP solver has stopped because of the prescribed time limit of one hour; the entry na means that no feasible solution was found within this time limit. In the last two columns of Table 2, we state the relative difference between the makespans obtained with the hybrid method and the analysed model of Baumann and Trautmann (2013) and the preliminary version of Fündeling and Trautmann (2006), respectively.

From Table 2 we draw the following conclusions.

- The hybrid method generates better schedules in the construction phase when random numbers are used to assign batches to groups. Although we used 287 different priority rules, we found only for instance I-12 a better schedule after the construction step compared to the 100 runs where we used random numbers for the assignment of batches to groups. After the improvement phase, the best schedules found by either configuration are equally good. However, the required CPU time per run is higher when priority rules are applied.

- The proposed hybrid heuristic (100 runs) considerably outperforms the preliminary version presented in Baumann and Trautmann (2011) (200 runs). To allow for a fair comparison, we applied the preliminary version 200 times since this version does not include an improvement phase. In general, the proposed hybrid method provides better schedules in less CPU time.

- The model of Baumann and Trautmann (2013) solves nine of the 20 instances to optimality; for 8 of these 9 instances an optimal solution has been devised using the proposed hybrid method (100 runs). Moreover, a feasible solution was obtained using the proposed hybrid method for the instance for which no feasible solution was found within the CPU time limit by the model of Baumann and Trautmann (2013). In general, the proposed hybrid method computes solutions with the same or a slightly worse makespan than the MILP model; for the medium-sized instance I-19, a considerably better solution was obtained, even when the hybrid heuristic is only applied 4 times.

For set II, we compared the results of the proposed hybrid method with the results of the heuristic presented in Fündeling and Trautmann (2006). Fündel-

Table 2: Numerical results for test set I

		BauTra13		BauTra11		Hybrid method										
				[200, Rnd]		[287, Prio]			[4, Rnd]			[100, Rnd]				
Inst	#op	C [min]	CPU [s]	C [min]	CPU [s]	C^C [min]	C^I [min]	CPU [s]	C^C [min]	C^I [min]	CPU [s]	C^C [min]	C^I [min]	CPU [s]	Δ BauTra13 [%]	Δ BauTra11 [%]
I-1	29	**489.0**	1	**489.0**	35	**489.0**	**489.0**	65	**489.0**	**489.0**	1	**489.0**	**489.0**	21	0.0%	0.0%
I-2	40	**536.5**	2	**536.5**	42	**536.5**	**536.5**	104	584.0	548.5	2	**536.5**	**536.5**	32	0.0%	0.0%
I-3	58	**664.0**	3	**664.0**	84	**664.0**	**664.0**	205	**664.0**	**664.0**	3	**664.0**	**664.0**	68	0.0%	0.0%
I-4	57	**715.5**	lim	721.0	108	**715.5**	**715.5**	263	725.0	725.0	4	**715.5**	**715.5**	93	0.0%	-0.8%
I-5	59	**616.0**	lim	658.0	107	670.0	**616.0**	306	655.5	650.5	4	**616.0**	**616.0**	84	0.0%	-6.4%
I-6	64	**684.0**	6	**684.0**	79	**684.0**	**684.0**	195	**684.0**	**684.0**	3	**684.0**	**684.0**	58	0.0%	0.0%
I-7	70	**684.0**	7	**684.0**	89	**684.0**	**684.0**	220	752.0	**684.0**	4	**684.0**	**684.0**	79	0.0%	0.0%
I-8	81	**684.0**	8	718.0	117	748.5	696.0	311	777.0	724.0	5	720.0	696.0	100	1.8%	-3.1%
I-9	92	**774.0**	lim	801.0	149	788.0	781.0	390	788.0	785.0	6	788.0	785.0	123	1.4%	-2.0%
I-10	93	**944.0**	10	**944.0**	122	**944.0**	**944.0**	336	960.5	**944.0**	5	**944.0**	**944.0**	114	0.0%	0.0%
I-11	117	**944.0**	11	**944.0**	193	952.0	**944.0**	553	993.0	979.0	8	**944.0**	**944.0**	189	0.0%	0.0%
I-12	117	882.0	lim	908.5	207	917.0	875.0	600	941.0	907.0	9	922.0	**865.0**	190	-1.9%	-4.8%
I-13	116	**1'036.0**	13	**1'036.0**	185	**1'036.0**	**1'036.0**	494	**1'036.0**	**1'036.0**	8	**1'036.0**	**1'036.0**	153	0.0%	0.0%
I-14	116	**879.0**	lim	916.0	205	947.0	886.5	575	947.5	947.5	8	914.0	890.5	176	1.3%	-2.8%
I-15	115	**878.0**	lim	909.0	214	982.5	904.0	567	961.5	943.0	9	904.0	904.0	191	3.0%	-0.6%
I-16	115	865.0	lim	911.0	241	947.0	895.0	673	947.0	947.0	9	911.0	**863.5**	208	-0.2%	-5.2%
I-17	120	1'053.5	lim	1'048.5	291	1'053.5	1'048.5	875	1'053.5	1'048.5	12	**1'043.5**	**1'043.5**	255	-0.9%	-0.5%
I-18	184	**1'740.0**	lim	**1'740.0**	426	**1'740.0**	**1'740.0**	1'215	**1'740.0**	**1'740.0**	18	**1'740.0**	**1'740.0**	391	0.0%	0.0%
I-19	234	1'700.0	lim	1'590.0	603	1'650.0	**1'521.0**	2'146	1'657.5	1'575.0	31	1'565.0	**1'521.0**	649	-10.5%	-4.3%
I-20	234	na	lim	1'505.5	610	1'580.0	**1'450.0**	2'156	1'586.0	1'520.0	28	1'481.0	1'451.0	616	na	na
Avg															-0.3%	-1.7%

Table 3: Numerical results for test set II

		FueTra06		Hybrid method					
		C	CPU	C^C	C^I	CPU	Avg #	Max #	Δ FueTra06
Inst	#op	[min]	[s]	[min]	[min]	[s]	Impr Iter	Impr Iter	[%]
II-1	1'391	7'946.0	3'184	8'039.5	**7'630.0**	3'216	708.75	1'087	-4.0%
II-2	1'308	7'601.5	3'019	7'537.5	**7'210.5**	2'961	544.00	830	-5.1%
II-3	1'282	7'633.0	2'662	7'782.5	**7'439.0**	2'860	587.25	893	-2.5%
II-4	1'289	7'423.0	2'523	7'157.0	**7'081.0**	2'482	344.00	555	-4.6%
II-5	1'289	7'422.5	2'572	7'384.0	**6'978.0**	3'238	850.75	1'271	-6.0%
II-6	1'205	7'024.0	2'575	6'914.0	**6'801.0**	2'034	366.50	568	-3.2%
II-7	1'329	7'522.5	2'993	7'235.5	**7'097.5**	2'871	444.00	595	-5.6%
II-8	1'212	7'233.0	2'433	7'056.0	**6'971.0**	2'087	386.25	457	-3.6%
II-9	1'070	6'224.0	2'081	6'509.5	**6'060.0**	1'901	407.50	512	-2.6%
II-10	1'198	6'872.0	2'491	6'690.5	**6'509.5**	2'230	509.25	731	-5.3%
Avg									-4.3%

ing and Trautmann (2006) propose 287 different multi-level priority rules; we have applied the heuristic of Fündeling and Trautmann (2006) 700 times for each of these rules per instance, which required an average CPU time of approximately 45 minutes. We applied the proposed hybrid method four times per instance, which corresponds to the same average CPU time. Table 3 shows the best makespan obtained by the heuristic of Fündeling and Trautmann (2006) and the proposed hybrid method for each instance in set II. The results indicate that the hybrid method considerably outperforms the heuristic proposed by Fündeling and Trautmann (2006). In columns eight and nine of Table 3, we list the average number of improvement iterations per run and the maximum number of improvement iterations over all four runs, respectively. These numbers indicate that even for instances with a large number of operations, the stopping criteria used for the improvement phase is met after a reasonable number of improvement iterations.

In Tables 4 and 5, we compare the proposed strategies to eliminate redundant variables and constraints to the variable-fixing strategy presented in Castro et al. (2009) and Kopanos et al. (2010). In Table 4, we analyse the illustrative example presented in Subsection 4.1 and state for both strategies the size of the MILP models that are solved during the construction phase. In each iteration, the model size, i.e., the number of constraints and variables, is indicated before and after the application of the preprocessing (PP) procedure of the Gurobi solver. Table 4 shows that the preprocessing procedure of Gurobi can eliminate only part of the redundant constraints and that the relative size of this part decreases in each iteration. In Table 5, we compare the two strategies for the first 135 iterations of instance II-1. The results show that applying the proposed strategies to eliminate redundant variables and constraints greatly reduces the CPU time required to solve the MILP models.

Table 4: Comparison of the proposed strategy and variable-fixing strategy for the illustrative example

	Proposed Strategy					Variable fixing Strategy				
	#Constr		#Vars		CPU [s]	#Constr		#Vars		CPU [s]
Iter	Before PP	After PP	Before PP	After PP		Before PP	After PP	Before PP	After PP	
1	35	4	28	5	<1	35	4	28	5	<1
2	61	37	40	22	<1	92	39	62	22	<1
3	82	57	50	32	<1	165	77	102	32	<1
4	107	80	60	41	<1	254	129	148	41	<1
5	132	104	70	50	<1	359	195	200	50	<1
6	161	130	80	59	<1	482	277	258	59	<1

7. Conclusions

We have presented a novel hybrid method for the short-term scheduling of make-and-pack production processes. The method consists of the following phases: decomposition, construction, and improvement. We have presented novel strategies for integrating MILP models into the construction and improvement phases of the hybrid method to efficiently solve the scheduling problems that arise in these phases. In an experimental performance analysis with large-scale, real-world instances provided by a consumer goods company, the novel hybrid method provides considerably better results compared with an existing state-of-the-art method. Moreover, in an analysis of small-scale instances for which optimal solutions are known, the novel hybrid method generates optimal or near-optimal schedules.

A major advantage of our hybrid method is its applicability to a wide range of production processes. The flexibility originates from the MILP model which can easily be modified to account for process-specific operating conditions. For example periodical cleaning of the processing equipment can be considered by introducing a sufficient number of unit-specific cleaning operations and a set of constraints that impose a given maximum time difference between consecutive cleaning operations on each unit. In future studies, we will further develop the MILP model to cover such operating conditions and also extend the improvement phase from a local search procedure to a variable neighbourhood search procedure, in which a neighbourhood is defined by the size of the groups of operations to be scheduled. Another promising direction for future research is the analysis of alternative objective functions to be used in the improvement phase, possibly starting from the objective function proposed for the construction phase. An additional analysis of the convergence characteristics of the improvement phase may also provide further insights. Eventually, we intend to apply the MILP integration strategies developed in this paper to hybrid methods for other related types of production process.

Table 5: Comparison of the proposed strategy and variable-fixing strategy for Instance II-1

	Proposed Strategy						Variable fixing Strategy					
	#Constr		#Vars		CPU [s]	Total CPU [s]	#Constr		#Vars		CPU [s]	Total CPU [s]
Iter	Before PP	After PP	Before PP	After PP			Before PP	After PP	Before PP	After PP		
1	435	252	424	235	<1	<1	435	252	424	235	<1	<1
15	683	629	528	328	<1	1	7'576	1'409	6'555	508	1	2
30	934	860	619	405	<1	3	17'390	4'119	13'698	586	1	13
45	1'263	1'038	750	530	1	10	30'244	9'005	21'719	704	4	63
60	1'454	1'368	802	571	1	22	45'469	15'701	30'369	748	6	205
75	1'755	1'510	903	661	1	35	61'897	23'206	39'122	825	36	422
90	2'119	1'863	1'067	817	1	68	81'400	33'141	48'881	983	18	667
105	2'144	1'875	1'017	754	4	110	103'221	44'829	59'307	922	50	1'218
120	2'643	2'356	1'243	967	1	155	128'448	59'109	70'805	1'125	39	2'041
135	3'135	2'830	1'400	1'112	2	208	155'560	74'907	82'779	1'267	138	2'980

Acknowledgements

This work was supported by the Swiss National Science Foundation (Grant No. 205121-125106).

Appendix

In the appendix, we present the pseudo-code for all of the algorithms used in the hybrid method. We use the notation introduced in Section 5. Algorithms 1 and 2 are used to update the unit-specific positions of batches in the processing sequence based on the solution provided by the MILP models in the construction and improvement phases.

Algorithm 1 Update positions of batches for task $k \in \{PM, FM, P\}$

for all $i \in N^k;\ j \in J_i^k : Y_{ij}^k = 1$ **do**
  $p_i^k := 1$
  for all $i' \in I_j^k$ **do**
    $p_i^k := p_i^k + (1 - X_{ii'}^k)$
  for all $i' \in N_j^k : Y_{i'j}^k = 1, i < i'$ **do**
    $p_i^k := p_i^k + (1 - X_{ii'}^k)$
  for all $i' \in N_j^k : Y_{i'j}^k = 1, i' < i$ **do**
    $p_i^k := p_i^k + X_{ii'}^k$
for all $i' \in I^k;\ j \in J_{i'}^k$ **do**
  $p_{i'}^k := p_{i'}^{prev,k}$
  for all $i \in N_j^k : Y_{ij}^k = 1$ **do**
    $p_{i'}^k := p_{i'}^k + X_{ii'}^k$

Algorithm 2 Update positions of batches for task $k \in \{S\}$

for all $i \in N^M;\ j \in J_i^S : Y_{ij}^S = 1$ **do**
  for all $i' \in I_j^M$ **do**
    $p_{ij}^S := p_{ij}^S + (1 - X_{ii'}^{FM})$
  for all $i' \in N_j^M : Y_{i'j}^S = 1, i < i'$ **do**
    $p_{ij}^S := p_{ij}^S + (1 - X_{ii'}^{FM})$
  for all $i' \in N_j^M : Y_{i'j}^S = 1, i' < i$ **do**
    $p_{ij}^S := p_{ij}^S + X_{ii'}^{FM}$
for all $i' \in I^M;\ j \in J_i^S$ **do**
  $p_{ij}^S := p_{ij}^{prev,S}$
  for all $i \in N_j^M : Y_{ij}^S = 1$ **do**
    $p_{ij}^S := p_{ij}^S + X_{ii'}^{FM}$

Algorithms 3 and 4 are used to compute the sets of immediate successors based on the positions computed with Algorithms 1 and 2. Algorithm 5 is used

Algorithm 3 Determine immediate successors of batches for task $k \in \{PM, FM, P\}$

for all $i \in I^k;\ j \in J_i^k$ **do**
  $SUC_i^k := \varnothing$
  for all $i' \in I_j^k$ **do**
    if $p_{i'}^k = p_i^k + 1$ **then**
      $SUC_i^k := SUC_i^k \cup i'$
  for all $i' \in N_j^k : Y_{i'j}^k = 1$ **do**
    if $p_{i'}^k = p_i^k + 1$ **then**
      $SUC_i^k := SUC_i^k \cup i'$
for all $i \in N^k;\ j \in J_i^k : Y_{ij}^k = 1$ **do**
  $SUC_i^k := \varnothing$
  for all $i' \in I_j^k$ **do**
    if $p_{i'}^k = p_i^k + 1$ **then**
      $SUC_i^k := SUC_i^k \cup i'$
  for all $i' \in N_j^k : Y_{i'j}^k = 1$ **do**
    if $p_{i'}^k = p_i^k + 1$ **then**
      $SUC_i^k := SUC_i^k \cup i'$

Algorithm 4 Determine immediate successors of batches for task $k \in \{S\}$

for all $i \in I^M;\ j \in J_i^S$ **do**
  $SUC_i^S := \varnothing$
  for all $i' \in I_j^M$ **do**
    if $p_{i'j}^S = p_{ij}^S + 1$ **then**
      $SUC_i^S := SUC_i^S \cup i'$
  for all $i' \in N_j^M : Y_{i'j}^S = 1$ **do**
    if $p_{i'j}^S = p_{ij}^S + 1$ **then**
      $SUC_i^S := SUC_i^S \cup i'$
for all $i \in N^M;\ j \in J_i^S : Y_{ij}^S = 1$ **do**
  $SUC_i^S := \varnothing$
  for all $i' \in I_j^M$ **do**
    if $p_{i'j}^S = p_{ij}^S + 1$ **then**
      $SUC_i^S := SUC_i^S \cup i'$
  for all $i' \in N_j^M : Y_{ij}^S = 1$ **do**
    if $p_{i'j}^S = p_{ij}^S + 1$ **then**
      $SUC_i^S := SUC_i^S \cup i'$

to identify the last pack batch of every packing line $j \in J^P$ in the construction and improvement phases. Algorithm 6 is used to compute an iteration-specific

Algorithm 5 Determine last batch in processing sequence of packing line $j \in J^P$

for all $i \in I_j^P$ **do**
  if $p_i^P = |I_j^P| + \sum_{i' \in N_j^P} Y_{i'j}^P$ **then**
    $LAST_j^P := LAST_j^P \cup i$
for all $i \in N_j^P : Y_{ij}^P = 1$ **do**
  if $p_i^P = |I_j^P| + \sum_{i' \in N_j^P} Y_{i'j}^P$ **then**
    $LAST_j^P := LAST_j^P \cup i$

value for the parameter M in the construction phase. Finally, Algorithm 7 is

Algorithm 6 Compute Big-M

$M := C^{prev}$
for all $i \in N^M$ **do**
  $M := M + \alpha^{PM} + \alpha^{FM} - \tau_i + \delta_i$
for all $i \in N^P$ **do**
  $M := M + \max_{j \in J_{i'}^P}(\alpha_{i'j}^P + \omega_{max}^P) - \omega_{max}^P$

used to update the sets of immediate successors after unscheduling a make batch in the improvement phase.

Algorithm 7 Update immediate successors for task $k \in K$ after removing make batch i

for all $j \in J_i^k;\ i' \in I_j^k$ **do**
  if $SUC_{i'}^k = i$ **then**
    $SUC_{i'}^k := (SUC_{i'}^k \setminus i) \cup (SUC_i^k \cap I_j^k)$
    while $SUC_{i'}^k \cap I_j^k \neq \varnothing$ **do**
      if $k = S$ **then**
        $i'' \in SUC_{i'}^k \cap I_j^k$
        $p_{i''j}^k = p_{i''j}^k - 1$
      else
        $i'' \in SUC_{i'}^k$
        $p_{i''}^k = p_{i''}^k - 1$
      $i' := SUC_{i'}^k \cap I_j^k$

References

Aguirre, A. M., Méndez, C. A., Gutierrez, G., Prada, C. D., 2012. An improvement-based MILP optimization approach to complex AWS schedul-

ing. Computers & Chemical Engineering 47, 217–226.

Baumann, P., Trautmann, N., 2011. Heuristic decomposition and LP-based scheduling in make-and-pack production. In: Ng, S., Jiao, R., Xie, M. (Eds.), Proceedings IEEE International Conference on Industrial Engineering and Engineering Management. Singapore, pp. 362–366.

Baumann, P., Trautmann, N., 2013. A continuous-time MILP model to short-term scheduling of make-and-pack production processes. International Journal of Production Research 51, 1707–1727.

Belaïd, R., T'kindt, V., Esswein, C., 2010. Storage problem in a shampoo making system. In: 8th International Conference of Modeling and Simulation. Hammamet, Tunisia.

Belaïd, R., T'kindt, V., Esswein, C., 2011. Decomposition algorithms for planning the production of a real shampoo industry. In: International Conference on Industrial Engineering and Systems Management. Metz, France.

Belaïd, R., T'kindt, V., Esswein, C., 2012. Scheduling batches in flowshop with limited buffers in the shampoo industry. European Journal of Operational Research 223, 560–572.

Bilgen, B., Günther, H.-O., 2010. Integrated production and distribution planning in the fast moving consumer goods industry: a block planning application. OR Spectrum 32, 927–955.

Bongers, P. M. M., Bakker, B. H., 2006. Application of multi-stage scheduling. In: W. Marquardt, C. P. (Ed.), 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering. Elsevier, pp. 1917–1922.

Castro, P. M., Grossmann, I. E., 2005. New continuous-time MILP model for the short-term scheduling of multistage batch plants. Industrial & Engineering Chemistry Research 44, 9175–9190.

Castro, P. M., Grossmann, I. E., Novais, A. Q., 2006. Two new continuous-time models for the scheduling of multistage batch plants with sequence dependent changeovers. Industrial & Engineering Chemistry Research 45, 6210–6226.

Castro, P. M., Harjunkoski, I., Grossmann, I. E., 2009. Optimal short-term scheduling of large-scale multistage batch plants. Industrial & Engineering Chemistry Research 48, 11002–11016.

Castro, P. M., Novais, A. Q., 2009. Scheduling multistage batch plants with sequence-dependent changeovers. AIChE Journal 55, 2122–2137.

Cerdá, J., Henning, G. P., Grossmann, I. E., 1997. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. Industrial & Engineering Chemistry Research 36, 1695–1707.

Dimitriadis, A. D., Shah, N., Pantelides, C. C., 1997. RTN-based rolling horizon algorithms for medium term scheduling of multipurpose plants. Computers & Chemical Engineering 21, 1061–1066.

Elzakker, M. A. H. V., Zondervan, E., Raikar, N. B., Grossmann, I. E., Bongers, P. M. M., 2012. Scheduling in the FMCG industry: An industrial case study. Industrial & Engineering Chemistry Research 51, 7800–7815.

Erdirik-Dogan, M., Grossmann, I. E., 2008. Slot-based formulation for the short-term scheduling of multistage, multiproduct batch plants with sequence-dependent changeovers. Industrial & Engineering Chemistry Research 47, 1159–1183.

Floudas, C. A., Lin, X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers & Chemical Engineering 28, 2109–2129.

Fündeling, C.-U., Trautmann, N., 2006. Scheduling of make and pack plants: a case study. In: Marquardt, W., Pantelides, C. (Eds.), 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering. Elsevier, pp. 1551–1556.

Giménez, D., Henning, G. P., Maravelias, C. T., 2009a. A novel network-based continuous-time representation for process scheduling: Part I. Main concepts and mathematical formulation. Computers & Chemical Engineering 33, 1511–1528.

Giménez, D. M., Henning, G. P., Maravelias, C. T., 2009b. A novel network-based continuous-time representation for process scheduling: Part II. General framework. Computers & Chemical Engineering 33, 1644–1660.

Gomes, M. C., Barbosa-Póvoa, A. P., Novais, A. Q., 2010. A discrete time reactive scheduling model for new order insertion in job shop, make-to-order industries. International Journal of Production Research 48, 7395–7422.

Günther, H.-O., Grunow, M., Neuhaus, U., 2006. Realizing block planning concepts in make-and-pack production using MILP modelling and SAP APO. International Journal of Production Research 44, 3711–3726.

Gupta, S., Karimi, I. A., 2003. Scheduling a two-stage multiproduct process with limited product shelf life in intermediate storage. Industrial & Engineering Chemistry Research 42, 490–508.

Harjunkoski, I., Grossmann, I. E., 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Computers & Chemical Engineering 26, 1533–1552.

Honkomp, S. J., Lombardo, S., Rosen, O., Pekny, J. F., 2000. The curse of reality - why process scheduling optimization problems are difficult in practice. Computers & Chemical Engineering 24, 323–328.

Jain, V., Grossmann, I. E., 2000. A disjunctive model for scheduling in a manufacturing and packing factory with intermediate storage. Optimization and Engineering 1, 215–231.

Kallrath, J., 2002. Planning and scheduling in the process industry. OR Spectrum 24, 219–250.

Kondili, E., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations - I. MILP formulation. Computers & Chemical Engineering 17, 211–227.

Kopanos, G. M., Lanez, J. M., Puigjaner, L., 2009. An efficient mixed-integer linear programming scheduling framework for addressing sequence-dependent setup issues in batch plants. Industrial & Engineering Chemistry Research 48, 6346–6357.

Kopanos, G. M., Méndez, C. A., Puigjaner, L., 2010. MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. European Journal of Operational Research 207, 644–655.

Kopanos, G. M., Puigjaner, L., Georgiadis, M. C., 2011. Production scheduling in multiproduct multistage semicontinuous food processes. Industrial & Engineering Chemistry Research 50, 6316–6324.

Kopanos, G. M., Puigjaner, L., Georgiadis, M. C., 2012. Efficient mathematical frameworks for detailed production scheduling in food processing industries. Computers & Chemical Engineering 42, 206–216.

Kurz, M. E., Askin, R. G., 2004. Scheduling flexible flow lines with sequence-dependent setup times. European Journal of Operational Research 159, 66–82.

Maravelias, C. T., 2006. A decomposition framework for the scheduling of single- and multi-stage processes. Computers & Chemical Engineering 30, 407–420.

Maravelias, C. T., Grossmann, I. E., 2003. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research 42, 3056–3074.

Marchetti, P. A., Cerdá, J., 2009. An approximate mathematical framework for resource-constrained multistage batch scheduling. Chemical Engineering Science 64, 2733–2748.

Méndez, C. A., Cerdá, J., 2000. Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. Computers & Chemical Engineering 26, 369–376.

Méndez, C. A., Cerdá, J., 2002a. An efficient MILP continuous-time formulation for short-term scheduling of multiproduct continuous facilities. Computers & Chemical Engineering 26, 687–695.

Méndez, C. A., Cerdá, J., 2002b. An MILP-based approach to the short-term scheduling of make-and-pack continuous production plants. OR Spectrum 24, 403–429.

Méndez, C. A., Cerdá, J., 2003a. Dynamic scheduling in multiproduct batch plants. Computers & Chemical Engineering 27, 1247–1259.

Méndez, C. A., Cerdá, J., 2003b. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. Optimization and Engineering 4, 7–22.

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers & Chemical Engineering 30, 913–946.

Méndez, C. A., Henning, G. P., Cerdá, J., 2001. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Computers & Chemical Engineering 25, 701–711.

Pantelides, C. C., 1994. Unified frameworks for the optimal process planning and scheduling. In: Rippin, D., Hale, J. (Eds.), Proceeding of the Second Conference on Foundations of Computer Aided Operations. CACHE, Austin, pp. 253–274.

Raaymakers, W. H. M., Hoogeveen, J. A., 2000. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. European Journal of Operational Research 126, 131–151.

Ramteke, M., Srinivasan, R., 2011. Novel genetic algorithm for short-term scheduling of sequence dependent changeovers in multiproduct polymer plants. Computers & Chemical Engineering 35, 2945–2959.

Roslöf, J., Harjunkoski, I., Björkqvist, J., Karlsson, S., Westerlund, T., 2001. An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. Computers & Chemical Engineering 25, 821–828.

Roslöf, J., Harjunkoski, I., Westerlund, T., Isaksson, J., 2002. Solving a large-scale industrial scheduling problem using MILP combined with a heuristic procedure. European Journal of Operational Research 138, 29–42.

Ruiz, R., Maroto, C., 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research 169, 781–800.

Shaik, M. A., Floudas, C. A., 2008. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. Computers & Chemical Engineering 32, 260–274.

Stefansson, H., Sigmarsdottir, S., Jensson, P., Shah, N., 2011. Discrete and continuous time representations and mathematical models for large production scheduling problems: a case study from the pharmaceutical industry. European Journal of Operational Research 215, 383–392.

Sundaramoorthy, A., Karimi, I. A., 2005. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. Chemical Engineering Science 60, 2679–2702.

Sundaramoorthy, A., Maravelias, C. T., 2008a. Modeling of storage in batching and scheduling of multistage processes. Industrial & Engineering Chemistry Research 47, 6648–6660.

Sundaramoorthy, A., Maravelias, C. T., 2008b. Simultaneous batching and scheduling in multistage multiproduct processes. Industrial & Engineering Chemistry Research 47, 1546–1555.

Venditti, L., Pacciarelli, D., Meloni, C., 2010. A tabu search algorithm for scheduling pharmaceutical packaging operations. European Journal of Operational Research 202, 538–546.

Wilkinson, S. J., Shah, N., Pantelides, C. C., 1995. Aggregate modelling of multipurpose plant operation. Computers & Chemical Engineering 19, 583–588.