Computers and Chemical Engineering 106 (2017) 883–891

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Industrial scheduling solution based on flexible heuristics

Iiro Harjunkoski*, Reinhard Bauer

ABB Corporate Research, Wallstadter Str. 59, 68526 Ladenburg, Germany

ARTICLE INFO

Article history:
Received 30 September 2016
Received in revised form 27 January 2017
Accepted 9 February 2017
Available online 16 February 2017

Keywords:
Scheduling
Standards
Heuristics
Large-scale problems

ABSTRACT

This paper presents a generic heuristic-based scheduling solution. It highlights the flexibility that a simple heuristic method can offer and shows that using the ISA-95 standard it is possible to express the most relevant problem requirements. In order to illustrate the possible benefits, the paper also compares the solution quality of a smaller scale example scheduling problem to a rigorous mixed-integer linear programming (MILP) approach and shows how a heuristic approach scales towards large-size industrial problems. The paper concludes with a discussion of the advantages and disadvantages of both approaches, showing that for certain types of problems, the heuristic approach is fully sufficient, even if it cannot be expected to result in optimal solutions.

1. Introduction

Planning and scheduling of a production process and/or a production facility is often a complex task and requires high expertise to be successfully performed. First of all, the process understanding must be sufficient in order to know what exactly needs to be considered within the scheduling activity and what the true consequenses and impacts of the decisions are on the production level. The decisions that are taken at the scheduling level highly depend on the company, as well as, on the existing IT-infrastructure. Secondly, the technique to perform the scheduling task requires either a highly experienced operator trained on the job or a person with good mathematical and analytical skills – preferably both. Nevertheless, these skills do not always meet. Apart from the fact of qualification and personal skills, scheduling is often a full-day activity, especially in the modern networked world where the presence and above all the awareness of frequent disturbances or changes in the process and in the surrounding environment are more a rule than an exception. These can upset the schedule causing that it becomes practically invalid to be executed on the plant floor. In this situation correcting the schedule becomes the main effort making all optimization targets secondary. Therefore it is valuable to have access to supporting tools that ensure correct, agile and more efficient reactions to changes and that open the possibility for optimization also within a complex and frequently changing environment.

In general, scheduling optimization problems tend to be complex due to their highly combinatorial nature, caused by the large number of alternatives e.g. in selecting the best production sequence and equipment assignment options, but also because of the fact that a scheduling model – or for the most part almost any mathematical model – is always a simplification of reality. In the scope of scheduling optimization the major modeling challenges are to select the most relevant aspects that are needed for considering real-life production, the decisions that must be optimized and how to transfer the obtained results onto the plant floor to create the desired impact. From the scheduling point of view, events occuring in production are relevant only if they have an impact on the feasibility or profitability of the current schedule. The size or severity of the impact should then decide whether a new schedule must be triggered or if some local adjustments are sufficient. In other words, it would be important to close the "control loop" for scheduling. The overall challenge of seeing scheduling only as part of a larger entity is addressed by the concept Enterprise-Wide Optimization (EWO) (Grossmann, 2005) and related mathematical programming challenges are discussed in Grossmann (2014). A number of industrial case studies with some of these practical considerations have been reported. Pinto et al. (2000) focuses on planning and scheduling of refinery operations, and more recently Zhao et al. (2017) presents an integrated optimization approach coupling up-stream refinery and down-stream ethylene plant operations. Tang et al. (2001) provides a review for steel production and Janak et al. (2006) for the chemical industry, for which EWO aspects are discussed in Wassick (2009). Laínez et al. (2012) gives an overview of the EWO-opportunities in the pharmaceutical industry and O'Sullivan and Newman (2015) provides strategies to schedule

* Corresponding author.
E-mail addresses: iiro.harjunkoski@de.abb.com (I. Harjunkoski), reinhard.bauer@de.abb.com (R. Bauer).

http://dx.doi.org/10.1016/j.compchemeng.2017.02.018
0098-1354/© 2017 Elsevier Ltd. All rights reserved.

entire underground mine operations. A thorough discussion on the industrial aspects is given in Harjunkoski et al. (2014), where also various solution and modeling approaches are discussed, along the lessons learned from the industrial implemenations of scheduling. In order to be able to solve larger problem instances that are mostly triggered either by EWO-problem extensions or real case studies, a number of approaches can be found in the literature. Decomposition algorithms based on MILP (Kopanos et al., 2010), as well as in Zhao et al. (2017), hybrid approaches combining various techniques such as CP and MILP (Jain and Grossmann, 2001), MILP, heuristics and simulation (Basán and Méndez, 2016), timed-automata based approaches (Subbiah et al., 2011), sequential MILP-based algorithms building a schedule in a constructive manner (Roslöf et al., 2002), as well as greedy heuristics (Pranzo et al., 2003) are examples of these. Other drivers of related research come from the fact of linking production scheduling closer to the online processes. Good discussions and examples of these are shown in Gupta and Maravelias (2016), Gupta et al. (2016) and Kopanos and Pistikopoulos (2014), where reactive scheduling aspects enable closer analysis of the actual process states. More generic methodological steps are presented e.g. in Vegetti and Henning (2015). There exists also vast literature on algorithmics and optimization. A good overview can be found for example in Cormen et al. (2009), as well as in Manber (1989). A good discussion of algorithms and optimization in the context of manufacturing scheduling is given in Framinan et al. (2014). This work covers algorithmic complexity, exact, approximate and heuristic approach as well as the respective models. The academic literature is very broad across different research communities and a more thorough review is only possible with a more narrow and specialized scope such as about a specific industry.

In order to be able to utilize existing modeling and solution techniques efficiently, one of the best ways is to productize them as part of a larger scheduling and dispatching framework (Harjunkoski, 2016). The main challenges when productizing scheduling solutions are:

- Defining a software-landscape that can host the algorithmic environment providing both flexibility to alternate between solution approaches and sufficient computational capacity
- Finding a generic problem description that is able to express realistic problem instances and that can be configured to meet also more specialized needs
- Gathering the necessary data and communicate the results into production such that any deviations can be detected efficiently
- Providing algorithms that work efficiently for various cases and provide – if not optimal – solutions that can be used in practice
- Creating configuration environments that allow a non-expert to easily maintain and adapt the solution.

In this paper, we address the above challenges and present an approach that mainly fulfills them. Providing a relatively simple and clearly represented example allows us to highlight some of the main aspects of using heuristic approaches, as well as, to compare the resulting solution with a rigorous MILP approach. The scalability issue is shown by reporting the performance of the heuristics for much larger problem instances, showing that many intractable MILP problems can still be solved within seconds using a suitable heuristic algorithm, naturally compromising on the optimality. Thus, it will be shown that heuristic solutions are a good alternative to rigorous optimization in some cases and that an especially interesting challenge for the future is to design hybrid solution approaches for larger scheduling instances that combine robust mathematical programming methods with “non-optimal” heuristics.

This paper compares the practical performance of full-space MILP models and rather simple heuristic approaches. While those

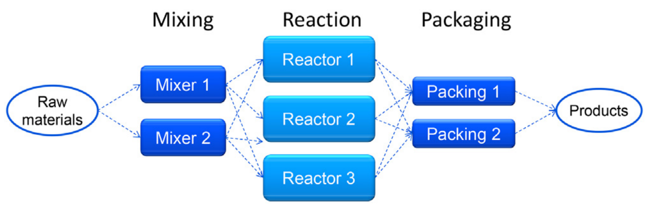


Fig. 1. Considered three-stage example production process.

two approaches follow completely different paradigms, both exist on their own right, being able to fulfill the requirements of different applications. A detailed discussion on the advantages and disadvantages of both paradigms and their corresponding scope is given later in the text. We would like to point out that math-heuristics stand in between both approaches, being the best choice for some applications that require a balance between algorithm runtime and solution quality.

2. Problem definition

For making it possible to treat the problem in a generic way and also putting the rest of this paper into context we define an example problem, which in this case is a simple chemical multi-stage batch process. The problem is an adaptation from Harjunkoski and Bauer (2014) and Harjunkoski and Bauer (2016), where we mainly have adjusted the processing times such that the bottleneck can shift depending on the solution candidate. The optimal solution is of course unique but having the variation can especially give a hint if non-optimal heuristic algorithm is driven towards a suboptimal solution and cannot thus fully use the desired flexibility. The example process comprises as before three stages: mixing, reaction and packaging. Here the reaction stage has three parallel machines and the other stages only have two alternatives to choose from. As earlier, all products cannot be processed on all machines, since the mixer 2 is incompatible for processing product B. There are also sequent-dependent changeover times, which must be considered. Fig. 1 shows a generic process overview.

Each of the stages consumes electricity and materials, however, when comparing heuristics with a mixed-integer programming formulation we exclude these in order to be able to use a standard precedence-based continuous-time approach with the main target of minimizing the make span. Nevertheless, it is straightforward to add the material constraints to the heuristic approach without large performance loss as will be shown later in this paper. Also, using a discrete-time approach (e.g. RTN in Castro et al., 2013) would be possible allowing full consideration of the material constraints but in this case, the problem size and resulting feasible grid density may not result in a global optimal solution from the makespan point-of-view, which is here of primary interest. The problem data is as follows: There are three different products that can be made (A, B, C) and we assume that a sequence-independent preparation or setup time is always needed, which is at minimum 15 min for mixers and reactors, depending on the sequence and always 60 min for the packaging machines, independent of the production sequence. This time is used e.g. for preparing and adjusting the equipment for the next task. The processing times are shown in Table 1, the field

Table 1
Processing times (min) of each product at each equipment.

Product	Mixer1	Mixer2	Reactor1	Reactor2	Reactor3	Packing1	Packing2
A	65	75	120	180	180	30	40
B	110	N/A	240	120	120	45	60
C	75	80	150	210	40	40	40

Table 2
Sequence-dependent changeover times (min) of each product at each equipment.

Mixer	A	B	C	Reactor	A	B	C
A	15	60	120	A	15	120	60
B	120	15	60	B	60	15	120
C	60	60	15	C	120	60	15

Table 3
Electricity consumption (kW) of each product at each equipment.

Product	Mixer1	Mixer2	Reactor1	Reactor2	Packing1	Packing2	Packing3
A	1200	1000	50	80	10	10	N/A
B	900	N/A	120	80	12	12	10
C	800	750	50	140	10	10	10

with N/A means that processing the product on the equipment is not allowed.

The sequence-dependent changeover times are shown in Table 2. For clarity, these also include the minimum setup-times mentioned above such that the information is complete, i.e. even if there is no product change we still require 15 min to start with the next batch. Please note that they are non-symmetric and apply to the mixer (left part of the table) and reactor (right part) stages, regardless of the equipment choice. For the packaging stage, there is only the sequence-independent changeover or setup time of 60 min.

By purpose, we have selected changeover times, where the ideal sequence on the mixer stage is not the ideal one for the reactor stage. To complement the example data we also provide some artificial material- and electricity consumption data, which however, can only be considered here in the heuristic approach.

Table 3 shows the electricity consumption of the given equipment which is invidual and varies between parallel equipment. This can be relevant if there is an upper power limit for the total simultaneous consumption (e.g. 2000 kW), a need to minimize the peak power consumption levels due to grid-capacity related fees, a wish to participate in a day-ahead electricity market or just to improve the overall demand-side management by better visibility and plannability. In Table 4, the material consumption is shown with the related stage during which the consumption takes place. It is assumed that consumed amounts must be at hand in the beginning of the batch stage and the produced amount will be available at the end. We assume that these amounts are stage-specific and independent of the selected equipment (i.e. fixed batch sizes). These constraints are mostly relevant in order to ensure raw-material availability or not to exceed for instance material silo or other storage capacity limitations.

Using the above data it is possible to create various problem instances for batch scheduling problems. The main instance in this paper that will be used for comparison comprises a number of production orders of each of the above products A, B and C, without limiting due dates.

3. Solution approach

The above example problem is here modeled and solved using a heuristic approach. In order to enable the flexibility and openness there must be a flexible data-interface for describing the scheduling problem. The ISA-95 standard (ANSI/ISA-95, 2005) has been used for importing all the above problem information. This is well motivated while the standard has been developed for more than two decades and contains a very complete description of the existing functions related to scheduling and control. In the so called Purdue Reference Model or functional model shown in Fig. 2, the wide dotted line illustrates the boundary of the enterprise-control interface and the 10 different functions are shown in the circles. The labeled lines indicate information flows of importance to manufacturing control and for example one can see that production orders from order processing are sent to the production scheduling module (2.0) and the resulting schedule then further to production control (3.0). All functions are described more in detail in the part 1 of the ISA-95 standard (ANSI/ISA-S95, 2000). Please note that this is not reflecting an organizational structure within a company, but an organizational structure of functions. Different companies will place the functions in different organizational groups. The configuration of the full scheduling problem is done using the XML-representation or implementation of the ISA-95 standard (Harjunkoski and Bauer, 2014), which allows to include all the information necessary for defining the problem in a generic and standardized way. To ensure an efficient operation for various industrial examples, the solution applies a number of generic heuristic algorithms, which are expected not to find the global optimal solution but instead in a very short time result in solutions that are close-to-optimal. For the earlier described example problem (Fig. 1), the goodness of the heuristic algorithm is evaluated by solving the smaller instances of the same problem to global optimality using an MILP model. For the comparison the only objective is the make span as it is easy to realize using both approaches. Different, also large-size use cases are shown to show the scalability of the algorithms, along the discussion on how the algorithmic performance can be affected both in terms of efficiency and quality.

One of the main ideas of using the ISA-95 standard is to be able to completely detach the problem-specific data from the algorithm. This should on one hand ensure that the tested algorithms are not affected by the problem type of instance and thus truly give an idea of how generic a methodology is, as well as where potential weaknesses are. On the other hand, the main idea of this separation is also to allow easier implementation of scheduling models or approaches. If the algorithm developer does not need to care about where to get the data from, how to modify it to fit to the problem description and which solution or channel to use for communicating the scheduling results back, he can really focus on the algorithmical work and have the benefit of being able to rapidly test the resulting solution onto various problem instances. The main principle is shown in Fig. 3, which illustrates that the scheduling function is entirely “isolated” between the data layers.

The benefit of the selected approach is also that the framework that hosts such a system can easily be complemented by new algorithms, also by those that have some problem-specific features making them especially suitable for the solution of certain classes or types of scheduling problems.

3.1. The MILP model

The MILP model used for the comparison (mainly for solution quality) is based on the continuous-time general precedence

Table 4
Material consumption (−) and production (+) amounts (kg) per stage.

Product	Raw material 1 (mixer)	Raw material 2 (mixer)	Reactor waste (reactor)	Final product (packaging)
A	−200	−200	+20	+320
B	−500	N/A	+50	+400
C	−150	−300	+15	+380

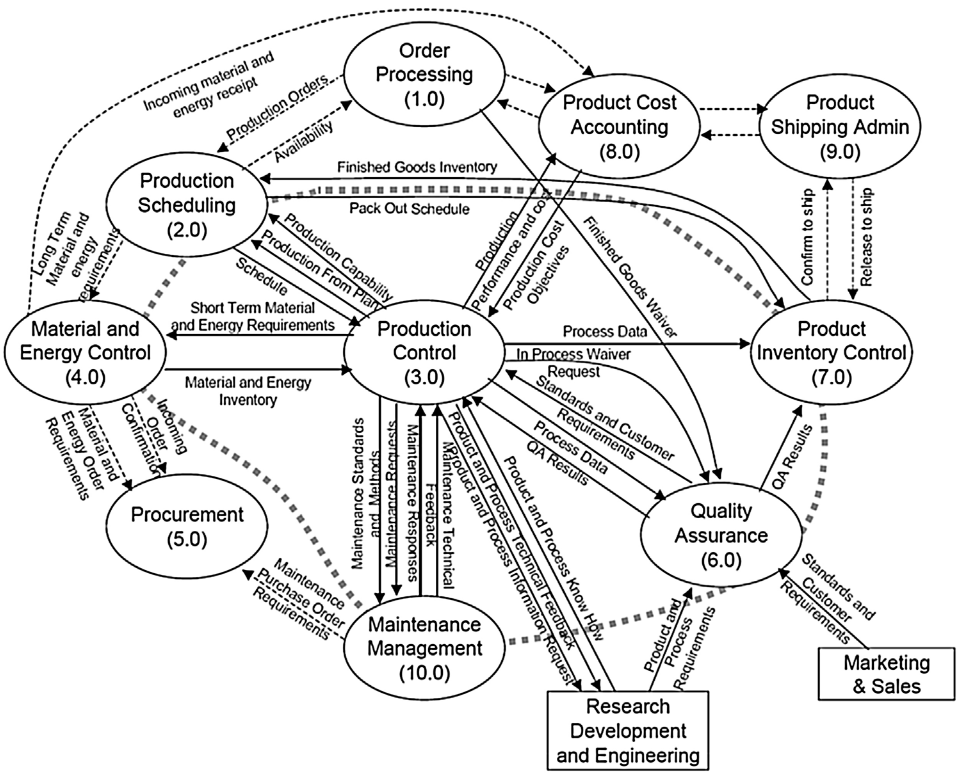

Order Processing (1.0)
Product Cost Accounting (8.0)
Product Shipping Admin (9.0)
Production Scheduling (2.0)
Production Control (3.0)
Material and Energy Control (4.0)
Procurement (5.0)
Quality Assurance (6.0)
Product Inventory Control (7.0)
Maintenance Management (10.0)
Research Development and Engineering
Marketing & Sales
Incoming material and energy receipt
Production Orders
Availability
Finished Goods Inventory
Pack Out Schedule
Long Term Material and energy requirements
Production Capability
Production From Plan
Schedule
Short Term Material and Energy Requirements
Material and Energy Inventory
Production Performance and cost
Production Cost Objectives
Finished Goods Waiver
Confirm to ship
Release to ship
Process Data
In Process Waiver Request
Standards and Customer Requirements
Process Data
QA Results
QA Results
Material and Energy Order Requirements
Incoming Order Confirmation
Maintenance Standards and Methods
Maintenance Requests
Maintenance Responses
Maintenance Technical Feedback
Product and Process Information Request
Product and Process Technical Feedback
Product and Process Know How
Maintenance Purchase Order Requirements
Product and Process Requirements
Standards and Customer Requirements

Fig. 2. Functional enterprise-control model (ISA-95).

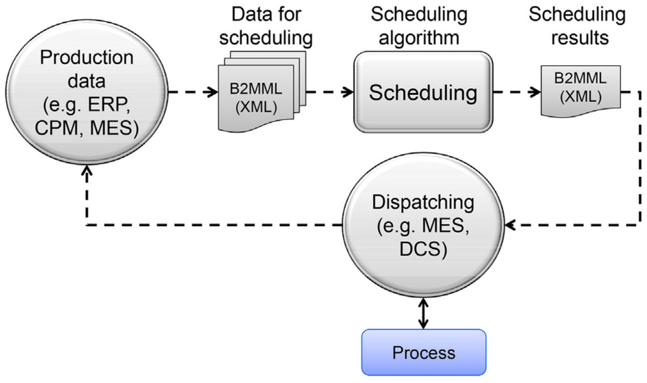


Fig. 3. Functional process flow of the scheduling.

approach (see e.g. Harjunkoski et al., 2014), where we assume that the same sequence between products is applied to all production stages. Even if the MILP model is not novel, for a fair comparison it is motivated to provide it in this context in order to also reveal its potential weaknesses or improvement needs.

3.1.1. Sets

The sets contains subsets that define equipment suitability to products and their allocation to stages

P	production orders p
S	production stages s
M	equipment m
M^s	pieces of equipment m belonging to stage s
M^p	pieces of equipment m that can process product p

3.1.2. Variables

$y_{p,m}$	assignment variable for product p on equipment m
$x_{p,q}$	sequencing variable between products p and q
$t^e_{p,s}$	ending time of stage s of product p
$t^s_{p,s}$	starting time of stage s of product p
$y^{aux}_{p,q,m}$	auxiliary variable tracking simultaneous assignment of products p and q
ms	make span

3.1.3. Parameters

$T_{p,m}$	processing time of product p on equipment m
τ^s_s	transfer time from stage s to the next stage
$\tau^c_{p,q,m}$	sequence-dependent changeover time from product p to q on equipment m
τ^{setup}_m	setup time of equipment m

3.1.4. Constraints

The assignment of product to equipment is done in Eq. (1), where exactly one suitable equipment must be assigned for each stage.

$$\sum_{M\in M^s\cap M^p} y_{p,m} = 1 \quad \forall p \in P, s \in S \tag{1}$$

Based on the selected assignment the duration of the stage, i.e. the difference between start and end times are defined in Eq. (2).

$$t^e_{p,s} = t^s_{p,s} + \sum_{M\in M^s\cap M^p} y_{p,m} \cdot T_{p,m} \quad \forall p \in P, s \in S \tag{2}$$

The next stage can only start after the earlier one has finished plus a mandatory transfer time that in this case is only stage-dependent (Eq. (3)) but could be easily extended to also be dependent on the equipment choices of the respective stages.

$$t^s_{p,s+1} \geq t^e_{p,s} + \tau^S_s \quad \forall p \in P, s \in S \tag{3}$$

The three sequencing constraints define that next product can only start after the sequence-specific changeover operations have been completed (Eq. (4)) and also after potential sequence-independent setup times on the equipment (Eq. (5)). This could also be integrated into one equation (Eq. (4)) by merging the two parameters into one. Finally, Eq. (6) defines that only one sequence is allowed.

$$t^e_{p,s} + \tau^c_{p,q,m} \leq t^s_{q,s} + M \cdot (1 - x_{p,q} + 1 - y^{aux}_{p,q,m}) \quad \forall p, q \in P, s \in S, m \in M^s \cap M^p \cap M^q \tag{4}$$

$$t^e_{p,s} + \tau^{setup}_m \leq t^s_{q,s} + M \cdot (1 - x_{p,q} + 1 - y^{aux}_{p,q,m}) \quad \forall p, q \in P, s \in S, m \in M^s \cap M^p \cap M^q \tag{5}$$

$$x_{p,q} + x_{q,p} \leq 1 \quad \forall p, q \in P | p \neq q \tag{6}$$

For the above sequencing constraints we need an auxiliary variable to ensure that it is only active if both products are assigned and processed in the same equipment. For this using a continuous auxiliary variable that takes the value one only in the case where this is true (Eqs. (7)–(9)).

$$y^{aux}_{p,q,m} \leq y_{p,m} \quad \forall p, q \in P, m \in M^p \cap M^q \tag{7}$$

$$y^{aux}_{p,q,m} \leq y_{q,m} \quad \forall p, q \in P, m \in M^p \cap M^q \tag{8}$$

$$y^{aux}_{p,q,m} \geq y_{p,m} + y_{q,m} - 1 \quad \forall p, q \in P, m \in M^p \cap M^q \tag{9}$$

Finally, we define the makespan through a constraint saying that it is always larger than the end time of the last processing step of each product.

$$ms \geq t^e_{p,s} \quad \forall p \in P, s = |S| \tag{10}$$

3.1.5. Objective function

In this case the objective is simply to minimize the make span, i.e. find the production schedule that corresponds to the minimum total production time.

$$\min ms \; y_{p,m}, x_{p,q} \in \{0, 1\} \qquad t^e_{p,s}, t^s_{p,s}, y^{aux}_{p,q,m}, \mathrm{ms} \geq 0 \tag{11}$$

The formulation can easily be expanded to cover the original more flexible sequencing, i.e. allow different sequences on different stages by adding the stage index to the sequencing variable, which then becomes $x_{p,q,s}$. Additionally, only Eq. (6) must be defined over the index s, as shown in Eq. (6b).

$$x_{p,q,s} + x_{q,p,s} \leq 1 \quad \forall s \in S, \forall p, q \in P | p \neq q \tag{6b}$$

Even if we claim that for the studied problem this will not result in an improved solution, especially in problems including several processing stages and high variations in processing times, this strategy i.e. replacing Eq. (6) by Eq. (6b) and updating the sequencing variable in all relevant constraints might improve the solution quality.

In the following we shortly describe the used heuristics.

3.2. Heuristic approach

The three heuristics tested are simple constructive heuristics that aim at locally optimizing the start times of single production steps without a full consideration of all consequences for the entire schedule of the plant. No global optimization objective is given to the algorithm. All heuristics are fully holistic and do not require any tuning or custom-tailoring to a specific instance as long as the instance is given in the already discussed ISA-95 sub-language used in the applied algorithmic framework. Because of the simplicity of the heuristics it is guaranteed that they can be applied to a large set of applications. Of course, the solution quality and – to an extend-also the algorithm runtime differs for specific instances or applications. However, both turned out to be fully sufficient for a very broad set of test instances stemming from various application areas.

To reach fast computation times, the heuristic algorithms utilize custom-tailored data structures which maintain information on resource availability over time. All resources (material, personal, equipment) are handled similarly and by the same data structures. For each resource, the following operations can be done quickly: consume or produce a resource at a given point in time, temporarily consume or produce a resource for a given time window, ask for the availability of a resource at a given time or time window, set lower and upper bounds, set or change the state of resource by also considering changeover-times. A highly efficient implementation of those datastructures is crucial for the viability of the overall approach. The technique of algorithm engineering has been applied to accomplish this requirement. Architecture-wise it is necessary to decouple the algorithmic engine from the ISA-95-based data definition part. This way, two major requirements can be enforced simultaneously: Easy data-definition through ISA-95 and custom-tailored, highly performant algorithms and datastructures that result in good algorithm performance and scalability. As a further plus of that approach, changes in data-definition do not affect the algorithmic part. As a drawback, we experienced the considerable complexity of the code that transforms the instances from one model to another. Our framework is handwritten and uses no third-party optimization libraries or datastructures. However, in principle it would be possible express our approach in terms of a constraint program. We skipped this approach as to the best of our knowledge, the data structures this would require are not yet supported by any high performant CP-solver.

The heuristic framework uses a rather generic and strong concept for handling timing and resource constraints. Each production step for each batch/product is modeled as a *task* which is unique in the entire model. No predefined stages or similar concepts have to be defined. Tasks that belong to the same production batch/product can be coupled by *dependencies.* Dependencies are of the form start/stop task B no later than x minutes after start/stop of task A. The resource constraint concept uniformly models the requirements of a production step on material, equipment and personel. For each task, the amount of a required resource, the list of resources actually being able to satisfy the requirement and the resource usage type (production, consumption, temporary production, temporary consumption) can be specified. Further, resource constraints can be annotated with *states* in order to model state-dependent setup-times.

The heuristics are called ASAP (as-soon-as-possible), BPA (basic pre-emption algorithm) and FPA (fast pre-emption algorithm). All three heuristics iteratively schedule (i.e. assign times, durations and resources) production steps, never correcting a decision. In each scheduling step (which is not to be mixed-up with the production step to be scheduled), a local optimization function is to be satisfied. The degree of freedom to improve that local target is the choice of the production step to consider next for scheduling. A scheduling step is performed in a way that all resulting “partial schedules” are feasible schedules with respect to the problem definition, though not scheduling all production steps to be considered.

The local optimization objective for a production step differs for the three tested heuristics. It is “early production step start time” for ASAP and ‘early production step end time with possible interrup-

Table 5
Main results (make span, solution times and the gap) from the common example problems with 9, 12, 15 and 60 orders.

Results	MILP1	MILP2	MILP3	ASAP	BPA	FPA
9 orders						
MS/time	760	**760**[a]	**760**[b]	845/0.003	820/0.007	820/0.005
Gap	27.0%	0.1%	0.1%	*11.2%*	*7.8%*	*7,8%*
12 orders						
MS/time	965	**935**	**935**	1130/0.01	1085/0.012	1085/0.004
Gap	52.1%	25.5%	27.3%	*20.1%*	*16.4%*	*16.4%*
15 orders						
MS/time	1170	1120	**1105**	1325/0.008	1260/0.025	1260/0.06
Gap	52.6%	39.1%	37.2%	*19.9%*	*14.0%*	*14.0%*
60 orders						
MS/time	6785	**3990**	5025	4190/0.172	4155/1.441	4155/0.14
Gap	93.7%	87.7%	89.0%	*5.0.%*	*4.1%*	*4.1%*

[a] Optimal solution reached at 2800 CPU-s.
[b] Optimal solution reached at 4225 CPU-s.

tions' for BPA. The objective function of the FPA mimics a greedy algorithm that aims to identify a feasible start time for the production steps in a given, predefined order. It is easy to adapt the heuristics to work with other local optimization objectives.

3.3. Data interface

As discussed above the complete data is provided through the ISA-95 XML-implementation called Business to Manufacturing Markup Language (B2MML). This is a structured way to put together complex data and the standard defines where to store which type of data. The details are given in the standards B2MML descriptions and can also be read in Harjunkoski and Bauer (2014). Here, we just highlight a few issues. The data is split into several XML files or structures in case the data is stored e.g. in a data base, of which each is well described by the standard. Most elements require an ID, which can be descriptive or based on system-internal naming practices and also allow to make comments in the corresponding "Description" fields. The XML file has a tree-structure, which also enables an object-oriented approach. This means that for instance when defining an equipment, all the properties and data for that equipment is stored under the specific branch. A simple example for the considered problem is shown in Fig. 4, which illustrates the systematic way of storing the data for scheduling.

4. Comparison of the methods

The scheduling problem defined in Fig. 1 with the data shown in Tables 1 and 2 is solved using the MILP model presented in Eqs. (1)–(11) and three heuristics. The results are presented in Table 5 and show 4 example problems. In the table the two first MILP models are equal but MILP1 was stopped after 100 CPU-seconds and MILP2 after 10000 CPU-seconds (roughly 2,8 h). For a comparison using a more flexible MILP model by adding the stage-index to the sequencing variable and replacing Eq. (6) with Eq. (6b), we also show the result of this model at 10000 CPU-seconds under the title MILP3. Allowing a longer solution time would in most cases not make any sense for the purpose of short-term scheduling. The three heuristics introduced above are marked as ASAP, BPA and FPA and the solution time is indicated next to the make span. On the second row, the integer gap is shown for the MILP problems and for the heuristics, these indicate the gap between the best MILP solution found – not the true optimality gap.

It is clear that a heuristic cannot guarantee optimality. Nevertheless, in many of the instances reported in Table 5, the heuristic solutions of the best algorithms lie between 4 and 20% of the best solution found by the MILP. This is of course still far away from an optimal solution but very promising in terms of the solution times. It also shows that part selecting the best type of heuristics that performs well both in terms of solution time and quality may not be such a trivial task. Here we can conclude that the FPA heuristic is the best choice and acknowledge that if the optimality is key then the heuristic cannot overperform the mathematical approaches. In MILP, the goodness can be easily measured (integer gap) but in larger solution instances it is meaningless either due to the fact that no solution has been found or the lower bound is simply still too far from the optimality. Also the results show that the added flexibility in MILP3 performs better only on the 15-order example, nevertheless the solution assumes same sequence on all processing stages and thus it is also a feasible solution for the simplified MILP2. The better result comes thus only from a more successful tree search. As we cannot measure an integrality gap for the heuristics the gap simply shows how far the solution is from the best MILP solution identified. In the 60 order case the MILP approach only found a better solution than the heuristic after 7150 CPU-s. All computations were done on a Laptop computer with an Intel I5 processor and 16Gb of memory. The MILP problems were solved on GAMS 24.5.3. using CPLEX 12.6.2.0 with standard settings.

Table 6
Computational results using the best heuristic (FPA) for 120 and 300 products.

Products	Make span	CPU-s	MILP1/MILP2/MILP3
120	7905	0.800	No solution found at 10000 CPUs
300	19215	8.188	No solution found at 10000 CPUs

Table 7
Smaller example problem data (mins).

Product	Mixer1	Mixer2	Reactor1	Transition time
A	120	180	360	0
B	180	120	240	60
C	240	180	480	120

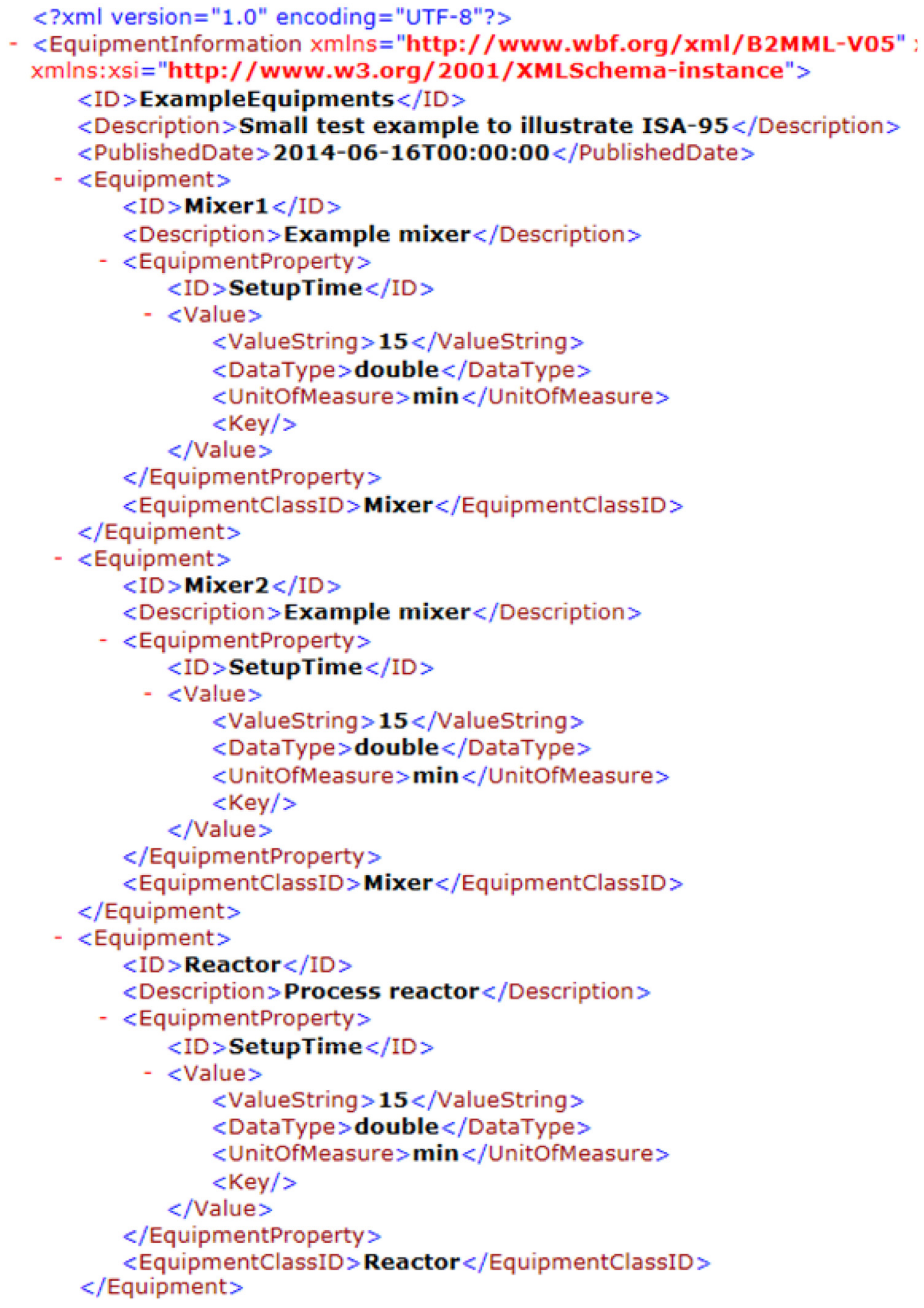
In order to prove the scalability of the best heuristics identified above (FPA), we performed two further large-scale tests (120 and 300 orders), the results of which are shown in Table 6. The MILP model did not find a solution to none of the problems within 10,000 s. Based on these problems, the solution quality is similar to the smaller problems and the scaling factor of the solution time is proportional to n^2.

An example of a Gantt Chart for the smaller test problems can be seen in Fig. 5. It shows the traditional tasks as rectangles but also tracks the utilities used in the production. This can also be an invaluable help for an operator to evaluate the goodness and suitability of a candidate solution. The Gantt chart also allows manual operations and can re-run a heuristic after a drag & drop operation, which calls for very agile and fast solution algorithms.

Another test problem was constructed with slightly different structure. We assume a problem with only two processing stages, where the first stage has two parallel machine and the second stage only one. We have three products A, B and C: The changeover times are the same as shown in Table 2. The production and transition times between stage 1 to stage 2 are shown in Table 7.

This problem basically turns out into a sequencing problem due to the strong dominance of the last processing stage with only a single equipment. The corresponding results of the problem solved by the MILP model (time limit 1000 CPUs) are shown in Table 8.

The results of a structurally different problem shows that the constructive heuristics performs qualitatively much better in a more sequence-intensive problem and in all examples the heuristic solution is within 1–2% of the best MILP solution found at 1000 CPU-seconds.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <EquipmentInformation xmlns="http://www.wbf.org/xml/B2MML-V05" :
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ID>ExampleEquipments</ID>
    <Description>Small test example to illustrate ISA-95</Description>
    <PublishedDate>2014-06-16T00:00:00</PublishedDate>
  - <Equipment>
      <ID>Mixer1</ID>
      <Description>Example mixer</Description>
    - <EquipmentProperty>
        <ID>SetupTime</ID>
      - <Value>
          <ValueString>15</ValueString>
          <DataType>double</DataType>
          <UnitOfMeasure>min</UnitOfMeasure>
          <Key/>
        </Value>
      </EquipmentProperty>
      <EquipmentClassID>Mixer</EquipmentClassID>
    </Equipment>
  - <Equipment>
      <ID>Mixer2</ID>
      <Description>Example mixer</Description>
    - <EquipmentProperty>
        <ID>SetupTime</ID>
      - <Value>
          <ValueString>15</ValueString>
          <DataType>double</DataType>
          <UnitOfMeasure>min</UnitOfMeasure>
          <Key/>
        </Value>
      </EquipmentProperty>
      <EquipmentClassID>Mixer</EquipmentClassID>
    </Equipment>
  - <Equipment>
      <ID>Reactor</ID>
      <Description>Process reactor</Description>
    - <EquipmentProperty>
        <ID>SetupTime</ID>
      - <Value>
          <ValueString>15</ValueString>
          <DataType>double</DataType>
          <UnitOfMeasure>min</UnitOfMeasure>
          <Key/>
        </Value>
      </EquipmentProperty>
      <EquipmentClassID>Reactor</EquipmentClassID>
    </Equipment>
```

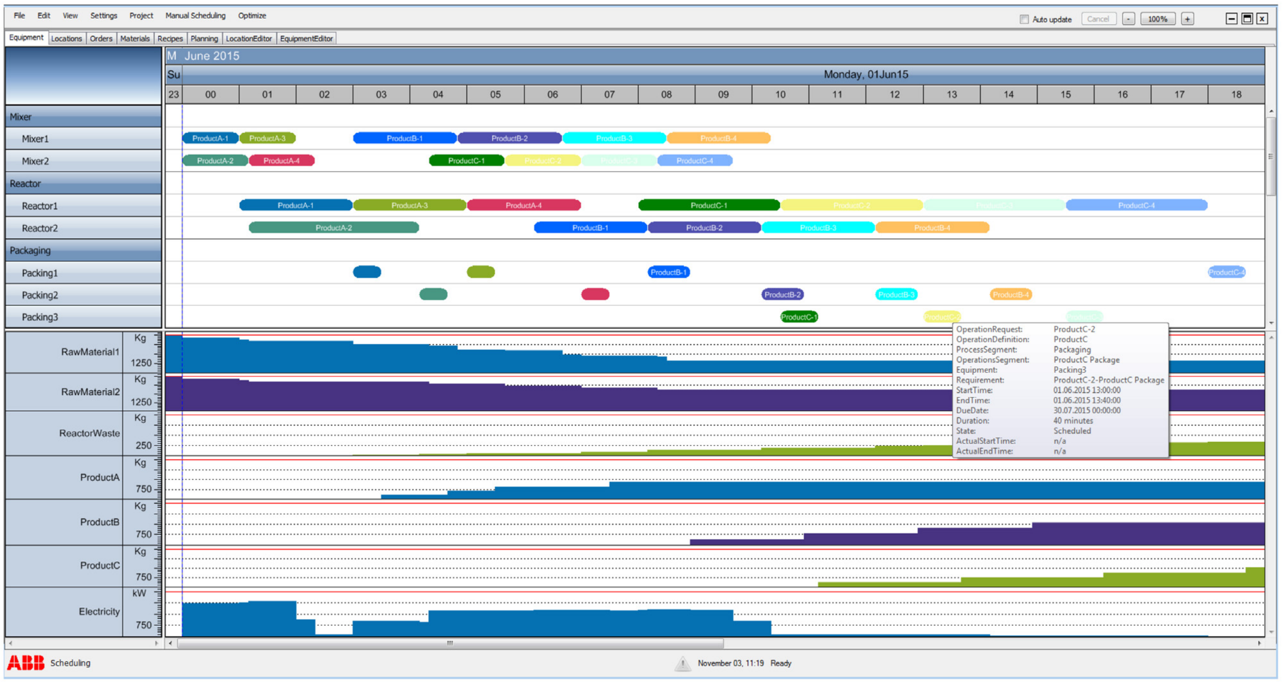
Fig. 4. Example of a B2MML file for the Equipment.

Table 8
Computational results for the smaller example problem.

Results	MILP			FPA		
	Obj.	CPUs	Gap	Obj.	CPUs	Difference
9 orders	3570	1000	10%	3630	0.001	1,7%
12 orders	4695	1000	40%	4755	0.001	1,3%
15 orders	5820	1000	55,2%	5880	0.036	1,0%
60 orders	N/A	1000	N/A	22755	0.050	N/A
120 orders	N/A	1000	N/A	45255	0.281	N/A
300 orders	N/A	1000	N/A	112755	3.516	N/A

Some of the main pro's and con's of using a heuristic algorithm vs. a robust MILP based algorithm is shown in Table 9. From there we can see that both approaches have their natural strengths and weaknesses. Heuristic approaches typically require programming, whereas the most important work is done within modeling for an MILP approach. As indicted above constructive heuristics have an interesting, additional feature: They can also be applied in an real-time environment where a guaranteed termination of the algorithm within a given time-period is required. This does not hold for MILP-based solutions which do not give you runtime guarantees. Furthermore, it is straightforward to do a theoretical complexity analysis of the applied heuristics.

The main benefit of heuristics is perhaps illustrated in a best way in Fig. 6., where the x-axis shows the CPU-s and y-axis the objective function value. The MILP objective value over time for the 60-order problem is shown with the solid gray line and the heuristic with the



OperationRequest:	ProductC-2
OperationDefinition:	ProductC
ProcessSegment:	Packaging
OperationsSegment:	ProductC Package
Equipment:	Packing3
Requirement:	ProductC-2-ProductC Package
StartTime:	01.06.2015 13:00:00
EndTime:	01.06.2015 13:40:00
DueDate:	30.07.2015 00:00:00
Duration:	40 minutes
State:	Scheduled
ActualStartTime:	n/a
ActualEndTime:	n/a

Fig. 5. Gantt chart showing the production plan and material information.

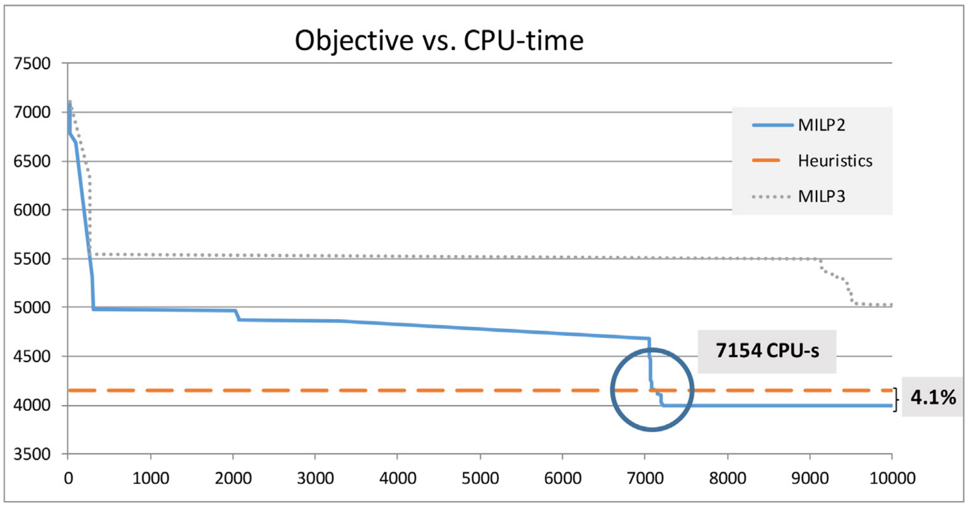


Fig. 6. MILP vs. heuristic performance over solution time (60-order problem).

partially dashed green line. One can see that the heuristics finds the solution within a fraction of a second. Also the MILP problem finds a solution within 15 s, the quality of which however is very low. Only after 5000 CPU seconds the MILP is able to overperform the heuristic and at the end of the calculation time the gap between the heuristic and MILP solution is only 11.2%. This shows how a heuristic algorithm can in a very short computational time deliver a fairly good solution but in the long run cannot compete in terms of optimality against a deterministic mathematical programming approach. Furthermore, in this particular case the only objective was to minimize the make span.

It is remarkable though that the generic MILP solver is able to find even a feasible solution within just a few seconds to such a large optimization problem. This is mainly owing to the built-in heuristics and presolver functions that most modern MILP solvers (here CPLEX) are equipped with. An interesting question is if the initial solution could be rigorously improved such that a better solution could be reported already earlier. Potential strategies could involve adding tighter cuts to the formulation (difficult to make it generic as this may require problem-specific knowledge as the solvers already apply clique cuts), start from a known heuristic solution (not a fair competition but partly the MILP solvers already apply root-node heuristics and model reduction) or solve a feasible subproblem or a set of them. However, as the problem is NP-hard, computationally these tasks may be close to as difficult as finding the optimal solution. Therefore, the combination of rigorous mathematical methods and heuristics is without question an interesting and challenging area of research.

5. Conclusions

In this paper we have shown that a generic heuristic approach can be efficiently used to solve many complex and large-size industrial scheduling problems. The solution quality of the heuristic

Table 9
Comparison of main features of the discussed approaches.

	Heuristics (ASAP, BPA, FPA)	Mathematical programming (MILP)
Pros	• Fast execution • Flexible • Very good scalability • Straightforward to model various requirements • Possible for interactive solutions	• Optimal solutions • Easy to balance various objectives • Well-proven, standardized solution engines • Modeling instead of algorithmic development
Cons	• Solution may be far from optimal • Difficult to balance objectives • More self-development needed • Constructive heuristics sensitive to the order in task list	• Long solution times for realistic problem sizes • Even a solution cannot be guaranteed for large problems • Not easy to determine which modeling approach works for which case

approach was evaluated by solving the same problems also using MILP to represent a rigorous mathematical programming approach. It is evident that both approaches have their pros and cons, which are listed in Table 9.

Even if the criterium against using a heuristic approach is that it simply often lacks the ability to navigate towards an optimum, in many cases for certain types of problems the heuristic approach is fully sufficient, in these example cases mostly at least 80% optimal. There are plenty of open questions with respect to heuristic approaches: How to reach a better support for scheduling problems from constraint programming solvers? How to enhance the proposed heuristics to iteratively optimize against arbitrary optimization functions? How to offline define a set of local heuristics that are able to cover a wide range of instance classes and optimization functions? How to online identify which heuristic fits best to a given instance? Also, an opportunity could be to consider possibilities to combine the strengths of both approaches: Using MILP to improve the quality of the solution supported by heuristics to boost up the performance. MILP solvers such as IBM CPLEX used for solving these examples have already successfully implemented various heuristics both in the presolvers as well as within the branching schemes that show their strengths and significantly improve the solution quality at the end of a branch and bound tree search. Nevertheless, it seems that larger problems are still not solvable to optimality and here various collaborative schemes could make a clear change.

References

ANSI/ISA-95.00.03-2005, 2005. Enterprise-Control System Integration. Part 3: Activity Models of Manufacturing Operations Management, ISBN 1-55617-955-3.

ANSI/ISA-S95.00.01-2000, 2000. Enterprise-Control System Integration. Part 1: Models and Terminology, ISBN: 1-55617-727-5.

Basán, N.P., Méndez, C.A., 2016. Hybrid MILP/simulation/heuristic algorithms to complex hoist scheduling problems. Comput. Aided Chem. Eng. 38, 1929–1934.

Castro, P., Sun, L., Harjunkoski, I., 2013. Resource-task network formulations for industrial demand side management of a steel plant. Ind. Eng. Chem. Res. 52 (36), 13046–13058.

Cormen, T., Leiserson, C., Rivest, R., Stein, C., 2009. Introduction to Algorithms, ISBN 978-0262033848.

Framinan, J., Leisten, R., Ruiz Garcia, R., 2014. Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools, ISBN 978-1-4471-6890-4.

Grossmann, I.E., 2005. Enterprise-wide optimization: a new frontier in process systems engineering (Conference Paper). AIChE J. 51 (7), 1846–1857.

Grossmann, I.E., 2014. Challenges in the application of mathematical programming in the enterprise-wide optimization of process industries. Theor. Found. Chem. Eng. 48 (5), 555–573.

Gupta, D., Maravelias, C., 2016. On deterministic online scheduling: major considerations, paradoxes and remedies. Comput. Chem. Eng. 94, 312–330.

Gupta, D., Maravelias, C., Wassick, J., 2016. From rescheduling to online scheduling. Chem. Eng. Res. Des. (in press).

Harjunkoski, I., Bauer, R., 2014. Sharing data for production scheduling using the ISA-95 standard. Front. Energy Res. 2/44, 1–15.

Harjunkoski, I., Bauer, R., 2016. Configurable scheduling solution using flexible heuristics. Comput. Aided Chem. Eng. 38, 2361–2366.

Harjunkoski, I., Maravelias, C.T., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014. Scope for industrial applications of production scheduling models and solution methods. Comput. Chem. Eng. 62, 161–193.

Harjunkoski, I., 2016. Deploying scheduling solutions in an industrial environment. Comput. Chem. Eng. 91, 127–135.

Jain, V., Grossmann, I.E., 2001. Algorithms for hybrid MILP/CP models for a class of optimization problems. INFORMS J. Comput. 13 (4), 258–276.

Janak, S.L., Floudas, C.A., Kallrath, J., Vormbrock, N., 2006. Production scheduling of a large-scale industrial batch plant. I. Short-term and medium-term scheduling. Ind. Eng. Chem. Res. 45, 8234–8252.

Kopanos, G.M., Pistikopoulos, E.N., 2014. Reactive scheduling by a multiparametric programming rolling horizon framework: a case of a network of combined heat and power units. Ind. Eng. Chem. Res. 53, 4366–4386.

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. Eur. J. Oper. Res. 207, 644–655.

Laínez, J.M., Schaefer, E., Reklaitis, G.V., 2012. Challenges and opportunities in enterprise-wide optimization in the pharmaceutical industry. Comput. Chem. Eng. 47, 19–28.

Manber, U., 1989. Introduction to Algorithms: A Creative Approach, ISBN978-0201120370.

O'Sullivan, D., Newman, A., 2015. Optimization-based heuristics for underground mine scheduling. Eur. J. Oper. Res. 241 (1), 248–259.

Pinto, J.M., Joly, M., Moro, L.F.L., 2000. Planning and scheduling models for refinery operations. Comput. Chem. Eng. 24 (9–10), 2259–2276.

Pranzo, M., Meloni, C., Pacciarelli, D., et al., 2003. In: Jansen, K., Jansen, K. (Eds.), A New Class of Greedy Heuristics for Job Shop Scheduling Problems. Springer LNCS 2647, pp. 223–236, WEA 2003.

Roslöf, J., Harjunkoski, I., Westerlund, T., Isaksson, J., 2002. Solving a large-scale industrial scheduling problem using MILP combined with a heuristic procedure. Eur. J. Oper. Res. 138 (1), 29–42.

Subbiah, S., Schoppmeyer, C., Engell, S., 2011. An intuitive and efficient approach to process scheduling with sequence-dependent changeovers using timed automata models. Ind. Eng. Chem. Res. 50 (9), 5131–5152.

Tang, L., Liu, J., Rong, A., Yang, Z., 2001. A review of planning and scheduling systems and methods for integrated steel production. Eur. J. Oper. Res. 133 (1), 1–20.

Vegetti, M., Henning, G.P., 2015. An ontological approach to integration of planning and scheduling activities in batch process industries. Comp. Aided Chem. Eng. 37, 995–1000.

Wassick, J.M., 2009. Enterprise-wide optimization in an integrated chemical complex. Comput. Chem. Eng. 33 (12), 1950–1963.

Zhao, H., Ierapetritou, M.G., Shah, N.K., Rong, G., 2017. Integrated model of refining and petrochemical plant for enterprise-wide optimization. Comput. Chem. Eng. 97, 194–207.