An Integrated System Solution for Supply Chain Optimization in the Chemical Process Industry

Guido Berning, Marcus Brandenburg, Korhan Gürsoy, Jürgen S. Kussi, Vipul Mehta, and Franz–Josef Tölle
Bayer AG, Bayer Technology Services
51368 Leverkusen, Germany

Abstract

We consider a complex scheduling problem in the chemical process industry involving batch production. The application described comprises a network of production plants with interdependent production schedules, multi–stage production at multi–purpose facilities, and chain production. The paper addresses three distinct aspects: (*i*) a scheduling solution obtained from a genetic algorithm based optimizer, (*ii*) a mechanism for collaborative planning among the involved plants, and (*iii*) a tool for manual updates and schedule changes. The tailor made optimization algorithm simultaneously considers alternative production paths and facility selection as well as product and resource specific parameters such as batch sizes, and setup and cleanup times. The collaborative planning concept allows all the plants to work simultaneously as partners in a supply chain resulting in higher transparency, greater flexibility, and reduced response time as a whole. The user interface supports monitoring production schedules graphically and provides custom–built utilities for manual changes to the production schedule, investigation of various what–if scenarios, and marketing queries.

Keywords

Supply chain management, Collaborative planning, Genetic algorithm

Inroduction

In recent years, there is a renewed interest in Advanced Planning and Scheduling (APS) applications (Stadtler and Kilger, 2000). This is evident from a range of new generation APS tools available in the market which combine the increased computational power with state–of–the–art optimization and database management techniques. Having implemented an Enterprise Resource Planning (ERP) system (e.g. Schumann, 1997), many companies are interested in integrating their supply chain internally and externally with the aid of the internet–based methodologies (e.g. Knolmayer et al., 2000, O'Leary, 2000, Tan et al., 2000, Yang, 2000).

Although the scope of applications varies, the primary needs for implementing APS tools remain the same: To plan future activities while anticipating the market but at the same time being flexible enough to handle changes in the short term and minimizing the effects of these changes on the long term perspective. The added emphasis on meeting conflicting business objectives, such as reducing response time, meeting due dates, increasing customer satisfaction, and reducing inventory, is an additional reason for using an APS tool.

Long term rough–cut planning essentially looks ahead to plan roughly and prepare the business well in advance for the upcoming market trends while meeting the primary goals of the business in terms of improving overall business efficiency. *Short term* scheduling, on the other hand, considers decisions on production sequencing, minimizing idle time and due date violations, and keeping the inventory level low, at the same time respecting all production constraints.

In practice, however, it is often not possible to represent all the constraints in a mathematical model or to provide a realistic estimate of the specific cost effects. Due to uncertainty in cost and production parameters the methods that strive for the true optimum are condemned to fail in practice and hence are rendered less effective in handling industrial applications. Many production constraints, such as sequence dependencies, lot size restrictions or resource allocations, lead to $\mathcal{NP}$–hard optimization problems (Monma and Potts, 1989, Pinedo, 1995), for which cost optimum solutions cannot be computed within a reasonable CPU time. Moreover, automated system solutions without user interface are of little use in many practical applications. Instead, flexibility based on qualitative factors such as market

and production dynamics and multiple conflicting business objectives is important. A holistic approach is needed, which combines the computational power and the flexibility of the modern APS tools with the possibility of user–driven interactions and manual interventions.

There are many articles in the scientific literature which deal with mathematical optimization problems of planning and scheduling in the chemical industry (e.g. Burkard et al., 1998a, Garcia–Flores and Wang, 2002, Kallrath, 2002a, Westenberger and Kallrath, 1994). Other papers present case study oriented research contributions (e.g. Blömer and Günther, 1998, Grunow et al., 2002, Mendez and Cerda, 2002) or focus on specific methodological aspects (e.g. Blömer and Günther, 2000, Burkard et al., 1998b, Kallrath, 2002b, Neumann et al., 2002, Timpe, 2002). According to our practical experience at Bayer AG, a successful planning application, however, involves three major components: (*i*) A comprehensive mathematical backbone to compute an optimized solution which can then be improved either through further optimization on specific sub–problems or through manual interaction, (*ii*) a mechanism for transparency, collaboration, information sharing and conflict management, and (*iii*) a powerful interface for manual interaction and problem visualization.

This paper emphasizes various aspects of an integrated planning and scheduling problem occuring in the process industry. Problems involving short and long term scheduling and planning with product chains passing through multiple batch production plants and consisting of many intermediate production steps are highlighted in Section 1. All the three aspects mentioned above were considered in the industrial application discussed in this paper: (*i*) A *genetic algorithm* (*GA*) was adopted instead of relying on other optimization techniques such as MILP or MINLP (e.g. Kolisch and Hartmann, 1998, Kolisch and Padman, 2001). The mathematical model and the implemented algorithm are described in Section 2. (*ii*) Increased transparency and reduced response time for changes in the production schedule were achieved by implementing a *collaborative planning model* which allows distributed decision making and access to the production schedule at the various plants involved. Additional system support for conflict management was provided by introducing the concept of a so–called "chain planner" who is authorized to look at production schedules of multiple plants simultaneously. The concept and architecture are presented in Section 3. (*iii*) Using a commercial APS tool, an application specific scheduling model was implemented to allow *manual updates and changes*, as described in Section 4.

1 Problem Statement

1.1 *Application Environment*

We consider a network of multi–purpose production plants that produce a variety of finished products through multi–stage production processes. The plants are linked by supplier–customer relations, i.e. one plant may produce intermediate goods that are processed further by other plants. The customer–supplier relations depend on the material flow given by the recipes of the final products. Furthermore, each plant interacts with external suppliers and customers. To fulfill customer orders, a variety of processing steps and a number of plants have to be considered depending on the recipe of the final products.

Each plant comprises a large number of multi–purpose production *facilities* (or *equipment units*) at which production takes place in *batch* mode. An ordered set of batches produced consecutively (without any delay in between) at the same facility is called a *campaign*. In general, the size of a campaign is limited from below.

A *product* is the outcome of a unique *process*. Each process involves various types of facilities (e.g. mixers, reactors, dryers) which have to be passed in a given order. The processing of a campaign at a facility is called an *activity*. Hence, a process can be defined as an ordered set of activities. Since a campaign consists of a set of batches which have to be processed one after another, an activity can be described as an ordered set of *operations*. An operation is defined as the processing of a batch.

Due to the peculiarity of the process industry the number of feasible combinations of process-specific equipment units (*routings*) is limited. Also the type of equipment feasible for a specific product is limited. The alternative product–equipment assignments are defined by a routing structure which consists of a source and a sink node and paths indicating the alternative production flows (see Figure 1a).

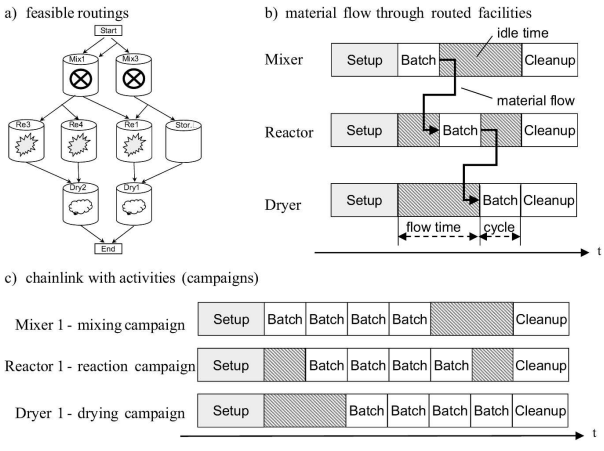

Figure 1. Routings, material flow, and campaigns

Before starting a process, the necessary equipment units have to be assembled, i.e. the various facilities have to be connected (*routed*) to each other (as required by the process) and to be cleaned. After termination of the process, the equipment is cleaned again. Therefore,

a *setup* time before a process and a *cleanup* time after a process is required.

The time between the beginning of the first activity and the beginning of the last activity of a process is defined as the *flow time*. The duration of an operation is characterized by a given *cycle time* which indicates the time span required to complete a batch at a facility (see Figure 1b). The cycle time depends on the process and the selected routing, but is identical for every operation carried out at the routed facilities.

The *material flow* between production facilities can be linear, convergent or divergent (due to coupled production). In the practical application considered here two types of demand can be distinguished, forecasts and customer orders. Each *demand* (or *requirement*) *element* is given by a required quantity of a specific product and a specific due date.

1.2 *Formal Description of the Decision Problem*

1.2.1 *Basics* The problem addressed here can be roughly stated as follows:

Given

- a set of production plants, their production equipment and corresponding capacities,
- a set of products, that can be processed in each plant, associated constraints, production parameters, routings, and bills of material (BOM),
- a set of demand elements (both forecasts and customer orders), and
- a set of penalties and cost functions.

Provide

1. a feasible schedule which simultaneously ensures that
   a. enough material at each BOM level is produced to satisfy the demands,
   b. early warnings for shortages of intermediates and raw materials are provided,
   c. all production constraints, resource availabilities, and business requirements are respected, and
   d. good solutions in a practical sense with regard to the multiple conflicting objectives are obtained;
2. a platform and a mechanism allowing
   a. every plant to operate independently, however to be able to collaborate with other plants for increased transparency and information sharing,
   b. reduction in response time to any changes in the production schedule of a plant acting as the internal supplier or customer, and
   c. conflict management and monitoring;
3. a system support and a user friendly tool for
   a. easy visualization of the production schedule,
   b. manual updates and interactive changes of production schedules,
   c. answering marketing queries,
   d. maintaining latest inventory positions, and
   e. maintaining master and transactional data.

Next, with the above explanations in mind, we give the notation used for the formulation of the optimization problem.

1.2.2 *Problem Parameters (Master Data)* Let $\mathcal{X}$ denote the set of multi–purpose, multi–product plants and $\mathcal{F}$ the set of facilities. Each plant $x \in \mathcal{X}$ involves a set $\mathcal{F}_x \subseteq \mathcal{F}$ of facilities, where $\mathcal{F} = \cup_{x\in\mathcal{X}}\mathcal{F}_x$ and for $x, x' \in \mathcal{X}$ with $x \neq x'$ we have $\mathcal{F}_x \cap \mathcal{F}_{x'} = \emptyset$. $\mathcal{P}$ denotes the set of products (raw material, intermediates or finished goods) that may be produced or consumed by processes.

The set of all processes is denoted by $\mathcal{A}$. Associated with each process $a \in \mathcal{A}$ are a set $\mathcal{P}_a^-$ of input and a set $\mathcal{P}_a^+$ of output products with $\mathcal{P}_a^- \cap \mathcal{P}_a^+ = \emptyset$. For any pair $\mathcal{P}_a^+, \mathcal{P}_{a'}^+$ with $\mathcal{P}_a^+ \neq \mathcal{P}_{a'}^+$ we have $\mathcal{P}_a^+ \cap \mathcal{P}_{a'}^+ = \emptyset$, i.e., each product $p \in \mathcal{P}$ is manufactured in a unique way. For product $p \in \mathcal{P}_a^-$ ($\mathcal{P}_a^+$), $\delta_{a,p}^-$ ($\delta_{a,p}^+$) denotes the input (output) fraction of product p for process a.

In order to manufacture a final product $p \in \mathcal{P}$ one has to produce a set $\mathcal{P}_p \subseteq \mathcal{P}$ of products consisting of p and the intermediates according to the recipe of p. Let $\mathcal{A}_p \subseteq \mathcal{A}$ denote the corresponding set of processes.

A routing $r \subseteq \mathcal{F}$ is an ordered set of facilities that have to be passed when manufacturing product p by process a_p. r_i is the i–th routed facility. The set of feasible routings associated with $a \in \mathcal{A}$ is given by $\mathcal{R}_a$.

The size b of a batch depends on process a as well as on the chosen routing $r \in \mathcal{R}_a$: $b = b(a, r)$. The same holds for the minimum campaign size $m = m(a, r)$, the cycle time $z = z(a, r)$, the flow time $f = f(a, r)$, the setup time $s = s(a, r)$, and the cleanup time $c = c(a, r)$. For the sake of simplicity, the dependence on a and r is omitted in the following if no misunderstanding can occur.

1.2.3 *Instance Parameters (Transactional Data)* The customer orders and forecast demands are reflected by a set $\mathcal{E}$ of requirement elements $e = (\pi, q, t_d)$. $\pi \in \mathcal{P}$ is the product required, q the corresponding quantity, and t_d denotes the due date. In general, to satisfy demands, a process $a \in \mathcal{A}$ has to be performed several times. In order to distinguish between these occurrences of a we use the term *chainlink* C_a, which denotes a demand–induced set of batches of process a, all performed in a single campaign using the same routing. To manufacture π, each process $a \in \mathcal{A}_\pi$, i.e. a corresponding chainlink C_a, has to be performed. According to the recipe of π, several processes $a', a'', \ldots$, i.e. chainlinks $C_{a'}, C_{a''}, \ldots$, have to be performed in a certain order. The resulting (partially) ordered set $\{C_{a'}, C_{a''}, \ldots\}$ of chainlinks is named (*product*) *chain* $\mathcal{C}_e$.

In the following, we assume that for all points in time t we have $t \geq 0$ where t is not allowed to exceed a given *time horizon*. The *inventory level* of product $p \in \mathcal{P}$ is denoted by $i(p,t)$. $i(p,0)$ is the *initial inventory level*.

The occupation of a facility $f \in \mathcal{F}$ is reflected by a *resource* (*occupation*) *object* $o_f = (t_s, t_c, n, v, w, a, r)$. t_s (t_c) denotes the start (completion) time of chainlink C_a at each facility $f \in r$ or, in other words, the start (completion) time of the activity at facility $f = r_1$ ($r_{|r|}$). n denotes the campaign size. v and w are binary variables indicating whether a setup or cleanup takes place ($v = 1$ and $w = 1$, respectively) or not ($v = 0$ and $w = 0$, respectively). (Later on, we show, how setup and cleanup can be saved.) Hence, all $f \in r$ are occupied during the time interval $[t_s - v \cdot s, t_c + w \cdot c[$. Besides the occupation of f by a chainlink, o_f might also reflect the occupation of f by a *shutdown* (e.g. for maintenance). In this case, we set $n = v = w = a = r = 0$. $\mathcal{O}_f$ denotes the set of all occupations of f and the vector $O = (\mathcal{O}_f)_{f \in \mathcal{F}}$ is called (*production*) *schedule*.

Upon generating a new schedule one has to consider that there are some chainlinks and shutdowns, reflected by resource objects $\hat{o}_f$, which have already been scheduled in the past and may not be changed. These objects form the *initial schedule* $\hat{O} = (\hat{\mathcal{O}}_f)_{f \in \mathcal{F}}$. Moreover, for some products $p \in \mathcal{P}$ a so–called *frozen zone* $[0, f_p[$, i.e. a kind of quarantine time where no scheduling of C_{a_p} is allowed, might be given.

To give a short impression of the size of real–world instances of the optimization problem, some key figures are listed in Tables 1 and 2.

Table 1. Master data

	average value	maximum value
$\lvert\mathcal{F}\rvert$	1,000 – 1,300	1,300
$\lvert\mathcal{A}\rvert$	600 – 650	700
$\lvert\mathcal{R}_a\rvert$	2 – 4	150
$\lvert r\rvert$	3 – 8	10
$\lvert\mathcal{A}_p\rvert$	3 – 6	10
$\lvert\mathcal{C}_e\rvert$	6 – 25	9,000

Table 2. Transactional data

	maximum value
$\lvert\mathcal{E}\rvert$	950
time horizon	600
objects in $\hat{O}$	1,200
inv. changes in $\hat{O}$	45,000

1.3 *Properties of a Feasible Solution*

A schedule O is called *feasible*, if the following conditions hold (with $o_f = (t_s, t_c, n, v, w, a, r)$ and $o'_f = (t'_s, t'_c, n', v', w', a', r')$):

1. O contains $\hat{O}$, i.e.:
$$\mathcal{O}_f \supseteq \hat{\mathcal{O}}_f \quad (f \in \mathcal{F}). \tag{1}$$
2. At each facility $f \in \mathcal{F}$, at most one resource object is scheduled at a time, i.e., for any pair o_f, o'_f with $t_s \leq t'_s$ we have
$$t_c + w \cdot c \leq t'_s - v' \cdot s' \quad (o_f, o'_f \in \mathcal{O}_f, f \in \mathcal{F}). \tag{2}$$
3. The frozen zones $[0, f_p[$ are obeyed, i.e.:
$$t_s - v \cdot s \geq f_p \quad (p \in \mathcal{P}; o_f \in \mathcal{O}_f \backslash \hat{\mathcal{O}}_f, f \in \mathcal{F}). \tag{3}$$
4. The minimum campaign sizes m are respected, i.e.:
$$n \geq m \quad (o_f \in \mathcal{O}_f, f \in \mathcal{F}). \tag{4}$$
5. The operations of an activity are performed one after another without preemption, i.e.:
$$t_c = t_s + f + n \cdot z \quad (o_f \in \mathcal{O}_f, f \in \mathcal{F}). \tag{5}$$
6. Each activity uses a feasible routing, i.e.:
$$r \in \mathcal{R}_a \quad (o_f \in \mathcal{O}_f, f \in \mathcal{F}). \tag{6}$$
7. The activities of a chainlink run in parallel and the setup and cleanup variables take the same values, i.e., for each pair $o_f, o_{\tilde{f}}$ with $o_{\tilde{f}} = (\tilde{t}_s, \tilde{t}_c, \tilde{n}, \tilde{v}, \tilde{w}, \tilde{a}, \tilde{r})$ and $\tilde{f} \in r$ we have
$$t_s = \tilde{t}_s, t_c = \tilde{t}_c, n = \tilde{n}, v = \tilde{v}, w = \tilde{w}. \tag{7}$$
8. If two resource objects o_f, o'_f are scheduled consecutively at the same facility f w.r.t. the same process a and routing r with $t_s \leq t'_s$, the cleanup after the activity corresponding to o_f and the setup before the activity corresponding to o'_f can be saved, i.e.:
$$r = r' \wedge a = a' \Rightarrow v = 0 \wedge w' = 0. \tag{8}$$
9. No negative inventory level may occur, i.e.:
$$i(p,t) \geq 0 \quad (p \in \mathcal{P}; t \geq 0) \tag{9}$$

1.4 *Solution Methodology*

As mentioned before, the application requirements are considered in three distinct aspects: (*i*) *optimization*, (*ii*) *collaborative planning*, and (*iii*) *manual interaction*.

1.4.1 *Optimization* Due to the size and complexity of realistic problem instances, exact solution methods are not applicable. Moreover, the problem structure does not allow to sub–divide the problem by plants, facilities or manufacturing levels, or to represent capacities, resources or the planning horizon in an aggregated way. An optimization method has to be chosen which quickly

computes initial solutions and shows a strong convergence behaviour. As *genetic algorithms* (*GA*) tend to have this property, it has been decided to use them as a basis for the implemented solution strategy. For an overview of genetic algorithms and their applications see Gen and Cheng (2000).

1.4.2 *Collaborative Planning* Using AspenMIMI© from AspenTech as a commercially available APS tool and a common server–client network structure, a platform for collaborative planning was established where local models are able to retrieve information from a central database. The data transfer and data consistency are realized with the help of remote function calls and macros.

1.4.3 *Manual Interaction* A user friendly tool was developed for easy and fast manual interaction and monitoring the production schedule. Again, AspenMIMI© from AspenTech was used, which — due to its toolbox structure with graphical user interface oriented programming — allows a high degree of freedom in customizing and development. A high level of user feedback has been considered in developing application specific utilities.

2 Solution Approach

Let $e = (\pi, q, t_d) \in \mathcal{E}$ be a requirement element and $\mathcal{R}_e = \{R^1, \ldots, R^{n_e}\}$ a set of *routing combinations* R determining for each process $a \in \mathcal{A}_\pi$ which routing $r \in \mathcal{R}_a$ to choose: $R^i \in \times_{a\in\mathcal{A}_\pi}\mathcal{R}_a$ $(i = 1, \ldots, n_e)$ where $n_e = \prod_{a\in\mathcal{A}_\pi} |\mathcal{R}_a|$. R_a denotes the routing $r \in \mathcal{R}_a$ to be chosen for all processes a in chainlink C_a. Hence, associated with each routing combination $R \in \mathcal{R}_e$ is a chain $\mathcal{C}_e$ (cf. Figure 2).

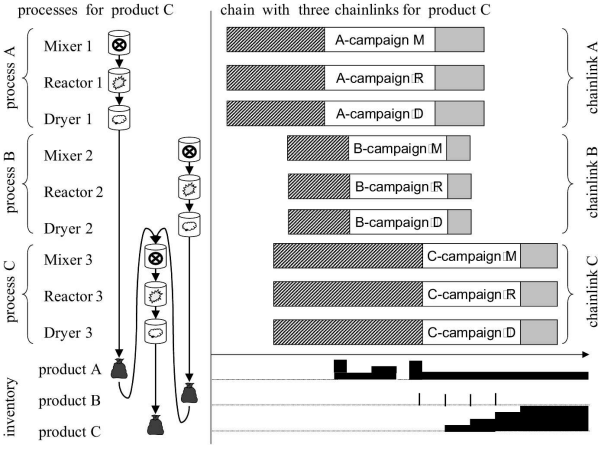

Figure 2. Chain with three chainlinks

The set $\{\mathcal{C}_e^1, \ldots, \mathcal{C}_e^{n_e}\}$ of all chains associated with $\mathcal{R}_e$ is denoted by Γ_e. $\mathcal{C}_e$ has the following properties:

- Each chainlink $C \in \mathcal{C}_e$ uses the routing given by R.
- The output of each chainlink $C \in \mathcal{C}_e$ is sufficient to satisfy the primary demand q and the secondary demands induced by q.

In the following, we show how the chains $\mathcal{C}_e \in \Gamma_e$ are generated based on the determination of $\mathcal{R}_e$ (cf. Section 2.1), computed (cf. Section 2.2), and scheduled (cf. Section 2.3), and how the obtained schedule can be improved (cf. Section 2.4). In Section 2.5, we show how all the components of GA work together. Section 2.6 is devoted to the parameter setup and in Section 2.7, the implementation issues are presented.

2.1 *Chain Generation*

Prior to the computation of $\mathcal{R}_e$ and Γ_e the sets $\mathcal{P}_\pi$ and $\mathcal{A}_\pi$ as well as the manufacturing levels l_p $(p \in \mathcal{P}_\pi)$ of product p are determined by means of Algorithm 1: Here, $\mathcal{P}'$ denotes an auxiliary set and l_e is the maximum manufacturing level associated with e.

Algorithm 1. Chain generation

$\mathcal{P}_\pi := \{\pi\}$; $\mathcal{A}_\pi := \{a_\pi\}$; $\mathcal{P}' := \{\pi\}$; $l_\pi := 0$
WHILE $\mathcal{P}' \neq \emptyset$ DO
  $\mathcal{P}' := \mathcal{P}' \backslash \{p\}$; $\mathcal{A}_\pi := \mathcal{A}_\pi \cup \{a_p\}$
  FOR $p' \in \mathcal{P}^-_{a_p}$ DO
    $\mathcal{P}' := \mathcal{P}' \cup \{p'\}$; $l_{p'} := l_p + 1$
  IF $p' \notin \mathcal{P}_\pi$ THEN $\mathcal{P}_\pi := \mathcal{P}_\pi \cup \{p'\}$,
$l_e := \max_{p\in\mathcal{P}_\pi} l_p$

Now, the set $\mathcal{R}_e$ can be easily obtained by setting $\mathcal{R}_e := \emptyset$ and adding all possible routing combinations R. A routing combination R is yielded by selecting exactly one routing $r \in \mathcal{R}_a$ for each process $a \in \mathcal{A}_\pi$.

2.2 *Chain Computation*

To compute a chain $\mathcal{C}_e$ w.r.t. routing combination R the following computations for each chainlink $C_a \in \mathcal{C}_e$ have to be performed:

First, the size n_{C_a} of the campaigns associated with C_a are determined such that the total (primary and secondary) demand q_p of each product $p \in \mathcal{P}_\pi$ is satisfied (cf. Algorithm 2, where $r := R_a$). Let $\mathcal{P}_l = \{p \in \mathcal{P}_\pi | l_p = l\} \subseteq \mathcal{P}_\pi$ be the set of all products in $\mathcal{P}_\pi$ with manufacturing level l.

Algorithm 2. Campaign sizes

$q_\pi := q$; FOR $p \in \mathcal{P}_\pi \backslash \{\pi\}$ DO $q_p := 0$
FOR $l := 0$ TO l_e DO
  FOR $p \in \mathcal{P}_l$ DO
    $n_{C_a} := \lceil \max\{q_p, \delta^+_{a_p,p} \cdot m \cdot b\}/(\delta^+_{a_p,p} \cdot b) \rceil$
    FOR $p' \in \mathcal{P}^-_{a_p}$ DO $q_{p'} := q_{p'} + \delta^-_{a_p,p'} \cdot b \cdot n_{C_a}$

Second, the set $Pred_{C_a}$ ($Succ_{C_a}$) of *predecessor* (*successor*) *chainlinks*, i.e. chainlinks $C_{a'}$ ($C_{a''}$) with $\mathcal{P}^+_{a'} \cap \mathcal{P}^-_a \neq \emptyset$ ($\mathcal{P}^+_a \cap \mathcal{P}^-_{a''} \neq \emptyset$) and the corresponding minimum time lags or *delays* $t_{C_{a'}C_a}$ ($t_{C_aC_{a''}}$) between the start time of $C_{a'}$ (C_a) and the start time of C_a ($C_{a''}$) have to be determined by means of Algorithm 3 (with $r' := R_{a'}$, $b' = b'(a', r')$, and $z' = z'(a', r')$). Here, $n_{p,i}$ denotes the minimum number of batches to be completed by C_a (with $a = a_p$) before the i–th batch of $C_{a'} \in Succ_{C_a}$ can be started w.r.t. the demand for (input) product $p \in \mathcal{P}^-_{a'}$.

Algorithm 3. Chainlinks and delays

FOR $C_a \in \mathcal{C}_e$ DO
  $Pred_{C_a} := \emptyset$; $Succ_{C_a} := \emptyset$
  FOR $C_{a'} \in \mathcal{C}_e$ DO
    IF $\mathcal{P}^+_{a'} \cap \mathcal{P}^-_a \neq \emptyset$ THEN
      $Pred_{C_a} := Pred_{C_a} \cup \{C_{a'}\}$
      $Succ_{C_{a'}} := Succ_{C_{a'}} \cup \{C_a\}$
  FOR $C_{a'} \in Succ_{C_a}$ DO
    $t_{C_aC_{a'}} := 0$
    FOR $p \in \mathcal{P}^+_a \cap \mathcal{P}^-_{a'}$ DO
      $n_{p,i} := \min\{j \in \mathbb{N} | \delta^+_{a,p} \cdot b \cdot j \geq \delta^-_{a',p} \cdot b' \cdot i\}$
      $t_{C_aC_{a'}} :=$
      $\max\{t_{C_aC_{a'}}, \max\{t_i | i \in \{1, \ldots, n_{C_a}\}\} + f\}$
      with $t_i := n_{p,i} \cdot z - (i-1) \cdot z'$

Note that each chain $\mathcal{C}_e \in \Gamma_e$ is computed independently from all other chains $\mathcal{C}_{e'}$ which have already been scheduled, i.e. without consideration of inventory levels at the start times of the chainlinks $C \in \mathcal{C}_e$. Moreover, the minimum campaign sizes might lead to overproduction, i.e. producing more than required to cover total demands. (In Section 2.4 it will be shown how overproduction can be decreased.)

2.3 *Chain Scheduling*

2.3.1 *Strategies* Having explained how chains are generated and computed, it will now be described how they are scheduled. Let a requirement element $e = (\pi, q, t_d)$ and the corresponding sets $\mathcal{R}_e$ and Γ_e be given. A chain $\mathcal{C}_e \in \Gamma_e$ is scheduled into an existing schedule O by iteration on the chainlinks $C_a \in \mathcal{C}_e$. For each chainlink C_a, a start time t_s is determined in such a way that

- each facility $f \in r := R_a$ is available within the time interval $[t_s - s, t_s + f + n \cdot z + c[$ with $n := n_{C_a}$,
- possible savings of setup and cleanup are realized,
- the delays $t_{C_{a'}C_a}$ with $C_{a'} \in Pred_{C_a}$ and $t_{C_aC_{a''}}$ with $C_{a''} \in Succ_{C_a}$ are respected,
- a violation of due date t_d is avoided (if possible),
- activities being performed consecutively at the same facility are synchronized, i.e. the time lags between their start times are kept short, and
- the obtained schedule O' is feasible (cf. Section 1.3).

After the determination of t_s a resource object o_f reflecting the occupation of f by C_a is created for each facility $f \in r$ and added to $\mathcal{O}_f$. Due to material flows between a chainlink and preceding and subsequent chainlinks, the chainlinks have to be scheduled in a certain order. We distinguish between two strategies:

- *Backward scheduling strategy*: The chainlinks C_a are ordered by increasing manufacturing levels l_p with $p \in \mathcal{P}^+_a$. Starting at time t_d, they are scheduled iteratively by this order.
- *Forward scheduling strategy*: The chainlinks C_a are ordered by decreasing manufacturing levels l_p with $p \in \mathcal{P}^+_a$. Starting at time 0, they are scheduled iteratively by this order.

As it is possible that $\mathcal{C}_e$ cannot be scheduled without exceeding t_d, the backward scheduling strategy does not lead to a feasible schedule O' in any case. In these cases, the forward scheduling strategy generates schedules with exceeded due dates. Then, both strategies are combined in a backward–and–reverse scheduling strategy that first tries to schedule a chain just–in–time using the backward scheduling strategy and, in case of failure, switches to the forward scheduling strategy.

2.3.2 *Scheduling Algorithm* A feasible schedule O is obtained from the initial schedule $\hat{O}$ by scheduling the chains $\mathcal{C}_e \in \Gamma_e$ ($e \in \mathcal{E}$) iteratively following an order given by a permutation ε of $\mathcal{E}$. If the surplus amount $i(\pi, t_d) - \sum_{e' \in \mathcal{E}: \pi' = \pi} H(t'_d, t_d) \cdot q'$ of product π being available at time t_d (with $H : \mathbb{R} \times \mathbb{R} \to \{0, 1\}, H(t, t') = 1 \Leftrightarrow t \leq t'$) is not sufficient to satisfy q, the corresponding chain $\mathcal{C}_e$ is scheduled w.r.t. a routing combination $R \in \mathcal{R}_e$. R is taken from a vector ϱ of routing combinations. (A detailed description follows in Section 2.5)

As mentioned above, we first try to schedule $\mathcal{C}_e$ according to the backward scheduling strategy. Before, the *current schedule* $\tilde{O} := O$ is saved and may be called again if the backward strategy fails and one has to schedule $\mathcal{C}_e$ according to the forward scheduling strategy.

When a chainlink C_a has been scheduled with start time t_s, the latest start time of each preceding chainlink $C_{a'} \in Pred_{C_a}$ with start time t'_s is updated as follows: First of all, the earliest precedence–feasible start time is determined:

$$t'_s := \min\{t'_s, t_s - t_{C_{a'}C_a}\}. \tag{10}$$

Afterwards, the resource constraints are taken into account by setting

$$t'_s := \max\{t \leq t'_s | [t - s', t + n' \cdot z' + f' + c'[\cap [t_s - v \cdot s, t_c + w \cdot c[= \emptyset \ (o_f \in \mathcal{O}_f, f \in r)\} \tag{11}$$

with $n' := n_{C_{a'}}$ and $r' := R_{a'}$. Provided that

$$t'_s - s' < 0 \ \text{ or } \ t'_s - s' < \max_{p' \in \mathcal{P}^+_{a'}} f_{p'} \tag{12}$$

(i.e. a frozen zone of an output product of a' is violated), the backward scheduling strategy is stopped and the forward scheduling strategy is restarted with $\tilde{O}$. Else, possible savings of setup and cleanup are realized: For each $f \in r'$, let o_f reflect the first occupation of f after t'_s. If $a = a'$ and $r = r'$ hold for each facility $f \in r'$, then we set $w := 0$, $v' := 0$, and increase the start time $t'_s := t'_s + c' + s$. (If the forward scheduling strategy has been applied, the first occupation before t'_s is examined and the time spans c and s' are reduced, if possible.)

Next, a resource object o'_f with $t'_c := t'_s + n' \cdot z' + f'$ is added to $\mathcal{O}_f$ ($f \in r'$). Moreover, the inventory levels of all input products $p \in \mathcal{P}^-_{a'}$ and output products $p \in \mathcal{P}^+_{a'}$ are updated.

2.4 *Schedule Improvement*

2.4.1 *Order Aggregation* As mentioned before, the constraints on minimum campaign sizes may cause overproduction. As the position of each chain is determined in the scheduling step, inventory levels cannot be considered in the previous chain computation step which computes each chain $\mathcal{C}_e$ independently from all others. Furthermore, scheduling chainlinks with overproduction (i.e. where the size n of the corresponding campaigns is too long) may result in a poor synchronization of the chainlinks $C_a \in \mathcal{C}_e$ because — with increasing n — it becomes more difficult to find a feasible start time t_s for C_a.

To consider these effects, chains are not generated and computed for single requirement elements $e \in \mathcal{E}$. Requirement elements for the same product are cumulated until minimum campaign sizes for all the required intermediates are reached. Basically, order aggregation applies a well–known procedure of material requirements planning (e.g. Günther and Tempelmeier, 2000), where the total demand for a product is computed based on the primary demands and the matrix of input factors.

At first, all requirement elements w.r.t. π that can be fulfilled using the initial inventory $i(\pi, 0)$ are removed from $\mathcal{E}$ and the inventory level for π is updated. Let $\mathcal{E}_\pi \subseteq \mathcal{E}$ be the set of all requirement elements for π ordered by increasing due dates. An *aggregate order* $\tilde{e} = (\tilde{\pi}, \tilde{q}, \tilde{t}_d)$ w.r.t. an (ordered) subset $\tilde{\mathcal{E}}_\pi \subseteq \mathcal{E}_\pi$ is given by

$$\tilde{\pi} := \pi, \;\; \tilde{q} := \sum_{e' \in \tilde{\mathcal{E}}_\pi} q', \text{ and } \tilde{t}_d := \min_{e' \in \tilde{\mathcal{E}}_\pi} t'_d. \tag{13}$$

Define the matrix A of (modified) input factors by $A := (a_{pp'})_{p,p' \in \mathcal{P}_\pi}$ with $a_{pp'} := \delta^-_{a,p} / \delta^+_{a,p'}$. Furthermore, let $\hat{q} = (\hat{q}_p)_{p \in \mathcal{P}_\pi}$ be the vector of *primary demands* with $\hat{q}_p = \tilde{q}$ for $p = \tilde{\pi}$ and 0, otherwise. Then, following Günther and Tempelmeier (2000), the vector $\bar{q} = (\bar{q}_p)_{p \in \mathcal{P}_\pi}$ of *total demands* can be obtained as follows:

$$\bar{q} := (I - A)^{-1} \cdot \hat{q}, \tag{14}$$

where I denotes the *identity matrix*.

To obtain aggregate orders for π, the first elements of $\mathcal{E}_\pi$ are aggregated until either

$$\bar{q}_p \geq \delta^+_{a,p} \cdot b \cdot m \;\; (p \in \mathcal{P}_\pi) \tag{15}$$

is fulfilled or all requirement elements of $\mathcal{E}_\pi$ have been aggregated. In the former case, the next aggregated order is generated by starting with the first element of $\mathcal{E}_\pi$ that has not been aggregated yet.

2.4.2 *Campaign Reduction* Neither the chain computation nor the order aggregation considers available inventory for intermediate products. Furthermore, the order by which requirement elements are aggregated does not affect the order by which the respective chains are scheduled nor the start times of the corresponding chainlinks. Therefore, it is necessary not only to check the campaign size prior to scheduling, but to adjust it after the scheduling of a chain in order to avoid overproduction.

After a chain $\mathcal{C}_e$ is scheduled, it is checked for all chainlinks $C_a \in \mathcal{C}_e$ if the size n of the campaigns associated with C_a can be reduced, starting with C_a having the lowest manufacturing level.

To check a chainlink C_a with start time t_s and completion time t_c, the surplus inventory i^+_p being available at t_c is determined for each output product $p \in \mathcal{P}^+_a$:

$$i^+_p := \min_{t \geq t_c} i(p, t) - \sum_{e \in \mathcal{E} : \pi = p} q. \tag{16}$$

The maximum number n^- by which each campaign corresponding to C_a can be reduced is given by

$$n^- := \min\{n, \min_{p \in \mathcal{P}^+_a} \{\lfloor i^+_p / \delta^+_{a,p} \cdot b \rfloor\}\}. \tag{17}$$

If $n^- = n$, then all campaigns corresponding to C_a can be saved and the resource objects o_f ($f \in r$) associated with C_a can be removed from the schedule. In addition, for each $f \in r$, we have a look at the first resource object o'_f (o''_f) before (after) t_s and adjust the setup and cleanup indicators accordingly: $w' := 1$ ($v'' := 1$). Then, we check for campaign reduction w.r.t. the chainlinks associated with o'_f (o''_f).

If $n^- < n$, then all campaigns corresponding to C_a are reduced by $\min\{n^-, n - m\}$ batches. In this case, the completion time t_c of C_a is reduced by $\min\{n^-, n - m\} \cdot z$ and the resource objects o_f ($f \in r$) associated with C_a are adjusted accordingly. Moreover, for each facility $f \in r$ the setup indicator of the first resource object o'_f after t_s is updated by setting $v' := 1$.

After C_a has been reduced, the inventory levels of all input and output products $p \in \mathcal{P}^-_a \cup \mathcal{P}^+_a$ are updated and campaign reduction is performed for all $C_{a'} \in Pred_{C_a}$. Having checked the whole chain $\mathcal{C}_e$, the delay times between its chainlinks have to be updated.

2.4.3 *Move–Algorithm* After a chain $\mathcal{C}_e$ is scheduled and all campaigns are reduced, $\mathcal{C}_e$, i.e. some of its chainlinks, might be moved backward or forward. By moving

a chainlink, idle time on the corresponding facilities is reduced and the synchronization of subsequent chainlinks is improved. To move a chainlink, first the direction of the possible move is determined. In case the due date t_d of the requirement element e is exceeded, it is tried to move all chainlinks backward, starting with a chainlink with the highest manufacturing level. In the other case, a forward move starting with a chainlink with the lowest manufacturing level is attempted.

A forward move of a chainlink $C_a \in \mathcal{C}_e$ begins with the determination of the latest possible start time $\bar{t}_s$ considering the existing schedule O, due date t_d, and delay times $t_{C_aC_{a'}}$ for all subsequent chainlinks $C_{a'} \in Succ_{C_a}$. After determining $\bar{t}_s$, it is checked if the inventory levels of all input and output products $p \in \mathcal{P}_a^- \cup \mathcal{P}_a^+$ remain nonnegative, provided that C_a is moved from t_s to $\bar{t}_s$. In that case, the move is performed, i.e., the start and completion time of each resource object o_f $(f \in r)$ associated with C_a, the inventory levels of all input and output products $p \in \mathcal{P}_a^- \cup \mathcal{P}_a^+$, and all affected setups and cleanups are modified accordingly.

2.5 *Optimization Algorithm*

2.5.1 *Basics* In the following, starting with an initial schedule $\hat{O}$, a feasible schedule O is obtained for an instance of the optimization problem using GA. Algorithm 4 shows how all components of GA work together (with $\mathcal{E}'$ denoting the initial requirement set).

O is determined w.r.t. a pair (*individual*) (ε, ϱ) consisting of a permutation $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_{|\mathcal{E}|}) \in E$ (with E denoting the set of all permutations of $\mathcal{E}$) and a vector $\varrho \in P = \{(\varrho_1, \ldots, \varrho_{|\mathcal{E}|}) | \varrho_e \in \mathcal{R}_e, e = 1, \ldots, |\mathcal{E}|\}$ of routing combinations (with ϱ_e determining the routing combination R to be used for chain $\mathcal{C}_e$ $(e = 1, \ldots, |\mathcal{E}|)$ and P denoting the set of all possible vectors of routing combinations).

For each $e \in \mathcal{E}$, we determine the routing combination $R := \varrho_e$ to be used for chain $\mathcal{C}_e$ and compute $\mathcal{C}_e$ w.r.t. R. Next, the chains $\mathcal{C}_{\varepsilon_1}, \mathcal{C}_{\varepsilon_2}, \ldots, \mathcal{C}_{\varepsilon_{|\mathcal{E}|}}$ are scheduled following the order induced by ε, and a feasible schedule $O = O_{\varepsilon,\varrho}$ is obtained. With an objective function F defined on the set of all feasible schedules, a fitness value $\tilde{F}((\varepsilon, \varrho)) := F(O_{\varepsilon,\varrho})$ can be assigned to each individual $(\varepsilon, \varrho) \in E \times P$.

Using appropriate methods of *replication* and *mutation* of individuals and *selection* to choose individuals from a *population* $\mathcal{I} \subseteq E \times P$, the strategy of genetic algorithms can easily be applied to obtain improved feasible schedules. In a preprocessing step, the sets $\mathcal{R}_e$ and P are generated. The optimization starts with a randomly generated initial population.

A fitness value $\tilde{F}((\varepsilon, \varrho))$ assigned to each individual $(\varepsilon, \varrho) \in \mathcal{I}$ is used to select a set $\mathcal{I}_s \subseteq \mathcal{I}$ of *survivors* from the population. By replication and mutation, a new population $\mathcal{I}'$ for the next iteration of the GA is generated from the survivors in $\mathcal{I}_s$. The iteration stops if a termination criterion is fulfilled, and the schedule $O_{\varepsilon,\varrho}$ corresponding to the individual (ε, ϱ) with the best fitness value $\tilde{F}((\varepsilon, \varrho))$ is returned.

Algorithm 4. GA

STEP 0 (*Initialization*)
Determine $\mathcal{E}$ by applying *order aggregation* to $\mathcal{E}'$
Determine $\mathcal{R}_e$ $(e \in \mathcal{E})$ by *chain generation*
Select $\mathcal{I} \subseteq E \times P$; $\mathcal{I}' := \emptyset$; $O := \hat{O}$; $F^\star := \infty$
STEP 1 (*Determination of new schedule*)
IF $\mathcal{I} \neq \mathcal{I}'$ THEN
  Select $(\varepsilon, \varrho) \in \mathcal{I} \backslash \mathcal{I}'$; $\mathcal{I}' := \mathcal{I}' \cup \{(\varepsilon, \varrho)\}$; $O := \hat{O}$; $i := 1$
  WHILE $i \leq |\mathcal{E}|$ DO
    Compute $\mathcal{C}_e$ w.r.t. $e := \varepsilon_i$ and $R := \varrho_{\varepsilon_i}$; $\tilde{O} := O$
    IF $\mathcal{C}_e$ can be scheduled *backward* into $\tilde{O}$ THEN $O := \tilde{O}$
    ELSE Schedule $\mathcal{C}_e$ *forward* into O
    Perform *campaign reduction* to $C_a \in \mathcal{C}_e$
    Perform *move–algorithm* to $C_a \in \mathcal{C}_e$
    $i := i + 1$
  $\tilde{F}((\varepsilon, \varrho)) := F(O_{\varepsilon,\varrho})$
  IF $\tilde{F}((\varepsilon, \varrho)) < F^\star$ THEN $F^\star := \tilde{F}((\varepsilon, \varrho))$; $O^\star := O$
  GOTO STEP 1
ELSE
  IF Termination criteria fulfilled THEN RETURN $O^\star$
  ELSE
    STEP 2 (*Determination of new population*)
    Determine $\mathcal{I}_s \subseteq \mathcal{I}$ by *selection*
    Determine $\mathcal{I}' \subseteq E \times P$ from $\mathcal{I}_s$ by *replication*
    Perform *mutation* to $\mathcal{I}'$
    $\mathcal{I} := \mathcal{I}'$; $\mathcal{I}' := \emptyset$; GOTO STEP 1

2.5.2 *Objective Function* In the industrial application considered, the planning horizon is significantly longer than the time between two consecutive planning runs. Thus, the distant part of the schedule will be rescheduled in later planning runs. Therefore, most terms of the objective function F are weighted by a time–dependent factor $e^{\omega \cdot t}$ with a degression parameter $\omega < 0$. As a consequence, the impact of a decision on the objective function value (e.g. violation of a due date) depends on the corresponding point in time t.

The first evaluation criterion for a schedule is given by the development of *inventory levels* over time. With $\Delta_t := \min_{t' \in T: t' > t} t' - t$ (where T denotes the set of all points in time t at which $i(p,t)$ changes) and $\Delta_{i(p,t)} = i(p,t) - \sum_{e \in \mathcal{E}: \pi = p} H(t_d, t) \cdot q$, the inventory level of a product $p \in \mathcal{P}$ can be quantified by

$$\sum_{t \in T} \max\{0, \Delta_{i(p,t)}\} \cdot \Delta_t \cdot e^{\omega \cdot t}. \tag{18}$$

For a requirement element $e = (\pi, q, t_d)$, let t_s (t_c) be the start (completion) time of chainlink $C_a \in \mathcal{C}_e$ with

$a = a_\pi$. The *earliness* and *tardiness* for e are measured by

$$\max\{0, t_d - t_c\} \cdot q \cdot e^{\omega \cdot t_d} \tag{19}$$

and

$$\max\{0, t_c - t_d\} \cdot q \cdot e^{\omega \cdot t_d}, \tag{20}$$

respectively. The durations of the *setups* and *cleanups* are measured by

$$v \cdot s \cdot e^{\omega \cdot (t_s - s)} \tag{21}$$

and

$$w \cdot c \cdot e^{\omega \cdot (t_c + c)} \tag{22}$$

respectively. The *makespan*

$$\max_{f \in \mathcal{F}} \max_{o_f \in \mathcal{O}_f} (t_c + w \cdot c) \tag{23}$$

of schedule O is also considered. Moreover, without going into detail, a *priority value* for each routing has been taken into account when computing $F(O)$.

With w_e, w_t, w_p, w_s, w_c, and w_m denoting weights corresponding to earliness, tardiness, (product) inventory levels, setup, cleanup, and makespan, the objective function value $F(O)$ is computed as follows:

$$\begin{aligned} F(O) \quad := \quad & \sum_{p \in \mathcal{P}} w_p \cdot (18) + w_e \cdot \sum_{e \in \mathcal{E}} (19) + w_t \cdot \sum_{e \in \mathcal{E}} (20) \\ & + w_s \cdot \sum_{f \in \mathcal{F}} \sum_{o_f \in \mathcal{O}_f} (21) + w_c \cdot \sum_{f \in \mathcal{F}} \sum_{o_f \in \mathcal{O}_f} (22) \\ & + w_m \cdot (23). \end{aligned}$$

In the following, we show how a new population $\mathcal{I}'$ can be obtained from a given population $\mathcal{I}$ with $|\mathcal{I}'| = |\mathcal{I}| = \lambda$.

2.5.3 *Selection* After determining the fitness value $\tilde{F}((\varepsilon, \varrho))$ of each individual $(\varepsilon, \varrho) \in \mathcal{I}$ as described above, a set $\mathcal{I}_s \subseteq \mathcal{I}$ of μ *survivors* is selected from the population $\mathcal{I}$ or from the populations $\mathcal{I}$ and the survivors $\mathcal{I}'_s$ of the former iteration. This selection can be performed either in a deterministic manner, choosing the first μ individuals with the best fitness value, or in a stochastic manner, choosing the individuals (ε, ϱ) randomly, each with a probability $\tilde{F}((\varepsilon, \varrho)) / \sum_{(\varepsilon', \varrho') \in \mathcal{I}} \tilde{F}((\varepsilon', \varrho'))$.

The set $\mathcal{I}_s$ is the basis for the generation of $\mathcal{I}'$. The individuals of $\mathcal{I}'$ are yielded by replication and mutation.

2.5.4 *Replication* The replication method consists of two strategies: *reduplication* and *recombination*. The former creates a new individual (ε, ϱ) by copying one randomly chosen survivor $(\varepsilon', \varrho') \in \mathcal{I}_s$: $(\varepsilon, \varrho) := (\varepsilon', \varrho')$. The latter generates a new individual (ε, ϱ) from two randomly chosen survivors $(\varepsilon', \varrho'), (\varepsilon'', \varrho'') \in \mathcal{I}_s$.

To recombine (ε', ϱ') and $(\varepsilon'', \varrho'')$, a *cross–over position* $i \in \{1, \ldots, |\mathcal{E}|\}$ is determined randomly. The permutation ε of the new individual is obtained by copying the first i elements of ε' to ε and adding all missing elements of $\{1, \ldots, |\mathcal{E}|\}$ to ε in the order given by ε''. The reduplication of ϱ is simply done by copying the first i elements of ϱ' and the last $|\mathcal{E}| - i$ elements of ϱ'' to ϱ. The ratio of the number of elements generated by recombination and the number of elements generated by reduplication is given by $\zeta \in [0, 1]$.

2.5.5 *Mutation* The mutation method modifies a replicated individual (ε, ϱ) to become a different individual (ε', ϱ'). Several *mutation parameters* are assigned to (ε, ϱ) to control the mutation: the *mutation rates* $\alpha_\varepsilon = \alpha_\varepsilon(\varepsilon, \varrho)$ and $\alpha_\varrho = \alpha_\varrho(\varepsilon, \varrho)$, which determine the number of mutations that will be performed to ε and ϱ, respectively, the *mutation step length* $\beta = \beta(\varepsilon, \varrho)$, and the *strategy selection probability* $\gamma_j = \gamma_j(\varepsilon, \varrho)$, which equals the probability for mutating ε by strategy j. Mutating an individual, these mutation parameters are mutated, too. Note that, if an individual (ε, ϱ) is obtained by recombination of two individuals (ε', ϱ') and $(\varepsilon'', \varrho'')$, its mutation parameters equal the arithmetic mean of the parameters of (ε', ϱ') and $(\varepsilon'', \varrho'')$.

To mutate ε, four different mutation strategies are applicable, and in each of the α_ε mutations, exactly one strategy j is chosen randomly with probability γ_j:

- The *exchange strategy* exchanges the elements ε_i and ε_j at two randomly chosen different positions $i, j \in \{1, \ldots, |\mathcal{E}|\}$.
- The *inversion strategy* inverts the order given by ε between two randomly chosen different positions $i \in \{1, \ldots, |\mathcal{E}|\}$, $j \in \{i-\beta, \ldots, i+\beta\} \cap \{1, \ldots, |\mathcal{E}|\}$.
- The *insert strategy* inserts the element ε_i at a randomly chosen position $i \in \{1, \ldots, |\mathcal{E}|\}$ into a randomly chosen target position $j \in \{i - \beta, \ldots, i + \beta\} \cap \{1, \ldots, |\mathcal{E}|\}$. This can be done either by *left insert* ($i > j$) or by *right insert* ($i < j$). The choice between left and right insert is done randomly with equal probability.
- The *move strategy* moves k randomly chosen subsequent elements of ε to other k randomly chosen subsequent positions. The positions $i \in \{1, \ldots, |\mathcal{E}|\}$ and $j \in \{i-\beta, \ldots, i+\beta\} \cap \{1, \ldots, |\mathcal{E}|\}$ as well as the length $k \in \{1, \ldots, |i - j|\}$ are chosen randomly with equal probability. The move strategy can be performed either to the left or to the right.

To mutate ϱ, we first set $\varrho' := \varrho$. Then, α_ϱ times, an element ϱ'_e is chosen randomly from ϱ' and changed to a randomly determined value which is taken from the set $\{1, \ldots, |\mathcal{R}_e|\}$.

The mutation parameters of an individual are mutated by considering two parameters $\phi \in [0, 100]$ and $\chi \in [0, 1]$. Each mutation parameter is either increased or decreased by ϕ per cent, and the probability for an increase or decrease equals χ. Hereafter, the mutated probabilities γ'_j are standardized so that they sum up to 1.

2.6 *Calibration of the Optimization Method*

To tune the parameters for optimization properly, two classes of them have been considered: First, the quality of the final schedule and the convergence behaviour of the algorithm can be influenced by the *GA parameters* and the way of creating the initial population. Second, the *objective function parameters* (weights) have to be set appropriately in order to reflect the overall business objectives.

2.6.1 *GA Parameters* As there is a trade–off between the computational time and the quality of the schedule obtained, the choice of appropriate termination criteria is highly important. We have experienced that evaluating not more than 200 – 500 populations and allowing a maximum computational time of 7,200 seconds leads to good results for problem instances of realistic size.

Settings of λ and μ influence the convergence behaviour of the algorithm, possible avoidance of local optima, and computational time for each iteration. A high ratio λ/μ makes it difficult to find good individuals in an early iteration, i.e. the algorithm will converge slowly. A low ratio λ/μ makes it more difficult to leave local optima. Good results have been obtained by choosing $\lambda/\mu \in [10, 20]$. Furthermore, it has been considered that the absolute values of λ and μ influence the computational time spent on each iteration of GA and therefore the time required for obtaining a first feasible solution. Numerical experiments revealed that not more than 40 individuals ought to be evaluated in each iteration.

The generation of an *initial population* can be performed by creating each individual either randomly or deterministically. Tests of both methods show that it is recommendable to create initial individuals (ε, ϱ) with ε being given by ordering the requirement elements in $\mathcal{E}$ by increasing due date t_d and with ϱ being obtained by choosing a chain using routings with highest priority. The mutation parameters for each initial individual are set deterministically, too: The mutation rates α_ε and α_ϱ are set to 1, i.e. ε and ϱ are mutated exactly once. Since the mutation of ε should consider the whole permutation, step length β is set to $|\mathcal{E}|$. Each mutation strategy should be chosen with equal probability.

For *selection*, tests have shown that a stochastic selection strategy has a negative impact on the selection pressure because individuals with a poor fitness value may be chosen to survive with a certain probability. For that reason, a deterministic strategy has been applied.

For *replication*, we have found out that better results are obtained if many individuals contain information given by two surviving individuals. Therefore, the recombination parameter ζ is set to a rather high value, i.e. $\zeta \geq 0.6$.

The *mutation* of an individual (ε, ϱ) can be performed by either mutating only ε or ϱ in one single mutation step or by mutating both. Best results are yielded by deciding randomly with equal probability if either ε or ϱ should be mutated, with the exception that both are mutated in the first iteration. For the mutation of ε, all four mutation strategies should be applied. The parameters ϕ and χ ought to be set in such a way that the probabilities for the insert and move strategy slightly increase while that of the invert and exchange strategy slightly decrease. The setting of the mutation parameter for β should ensure that it is expected to increase β by 25 per cent. Hence, even in later iterations it is probable that the whole permutation ε is considered during mutation.

2.6.2 *Objective Function Parameters* Good results are obtained with a degression parameter $\omega \in [1/60, 1/80]$. The weights w_s and w_c are taken from the interval $[1,000; 3,000]$ and the other weights are chosen from $[1, 3]$.

2.7 *Implementation Issues*

The optimization algorithm has been coded in C++ using a two–processor Intel Pentium machine with 500 MHz clock pulse and 2 GB memory under the operating system Windows NT© 4.0. The duration of setting up an optimization model for a problem instance (with a size as given by Tables 1 and 2) and of the following optimization run is in the range of 45 minutes to 120 minutes, depending on the parameter settings for the termination criteria and the size of the problem instance.

It is appealing to implement a GA on computers with parallel CPUs since each individual of a population can be treated independently from others. One way to do that (in an object–oriented manner) is to use *threads* (Hughes and Hughes, 1997). Threads are in some way independent processes which use the same memory in the RAM for the calling process. The key issue in using threads is to avoid *deadlocks*. Deadlocks can occur when more than one thread tries to access the same element in memory. This problem can be solved by encapsulating elements by *semaphores*. A thread can access an element only by taking its semaphore before and relasing it after the access. Threads that try to use the same element at the same time have to wait until the semaphore is free. The threads, which are thrown in a loop over all individuals, are scheduled by the dispatcher of the optimization system.

If there are several CPUs, the threads are scheduled in parallel at the same time on different CPUs. All terminated threads are caught in another loop. The next population can be computed after all threads have been terminated and the new population has been generated.

3 Collaborative Planning

3.1 *The Approach*

The optimization model comprises an integrated scheduling model and is run in a nightly batch job. The solution provided by the optimization run, however, is subject to manual changes and updates (cf. Section 4). Due to the integrated structure of the production system and interdependencies of plant schedules, changes

in one plant affect multiple other plants, and the ripple effects can increase the magnitude of changes in downstream plants. In the industrial application, 11 plants are integrated, and the plant schedules are highly interdependent. More than 50 per cent of the end products are produced in chains involving more than 4 plants. For this reason, it is important to consider the latest schedule situation in other plants while making changes in a given plant. Due to performance and flexibility issues these changes are carried out locally by the plant planner. However, a plant planner is allowed to change only part of the schedule. For this purpose, the integrated model containing the results of the optimization run is broken down into plant–specific models. Individual plant planners can use the reduced models for manual changes to the solution proposed by the system. Each sub–model is a copy of the integrated model with only the plant–specific resources.

From a plant scheduler's point of view, the studies on the integrated model brought out the need for a model architecture, where the individual plant scheduler should

- be able to work on a smaller model with the facilities the planner is allowed to schedule but at the same time be able to share and view information with and from others plants,
- be able to see the schedule changes to the relevant production steps in other plants,
- make other plants aware of the schedule changes, and
- with the help of a chain planner, reduce conflicts and find a mutually agreeable solution for product chains running through multiple plants.

The model that was implemented in this context is called *collaborative planning model*. From the business point of view, the collaborative planning model serves the following goals:

- transparency of production schedules and shared information among all the plants,
- reduction in response time for reacting to any changes in the production schedule of upstream or downstream plants,
- minimization of conflicts among involved plants by simultaneously considering the production schedule of a specific subset of plants, and
- effective use of the global inventory maintained in the individual plants.

The objectives of shared information can be served by having one centralized model which can be accessed by the different plant planners, who are allowed to operate only on the authorized part of the schedule. However, this approach was found unacceptable due to the poor system performance arising from the projected size and complexity of the integrated model, selected APS system, and links with the optimizer. Hence, an alternative approach was adopted. The approach considers each plant as a separate entity or partner in a supply chain which needs to be integrated, and all partners work in collaboration. The collaboration is achieved by sharing the relevant part of the production schedule from each plant by every other plant. This architecture allows breaking up of the integrated production schedule model involving all plants into separate plant specific models. As the separate model for each plant is a reduced model, the approach guarantees a good system response time for manual changes.

The adopted approach required an architecture and a mechanism to gather information about the latest production schedule in all other partner plants and to publish production schedule changes to the partner plants. This transfer must maintain the consistency of data and schedule. In addition, taking advantage of the fact that all the plants are part of the same organization, a mechanism was also established to prepare a virtual plant involving multiple plants to consider decisions for chain production.

3.2 *System Architecture*

The system architecture is based on a client–server-concept. It involves plant specific models on the local client communicating with an integrated model on a server (see Figure 3).

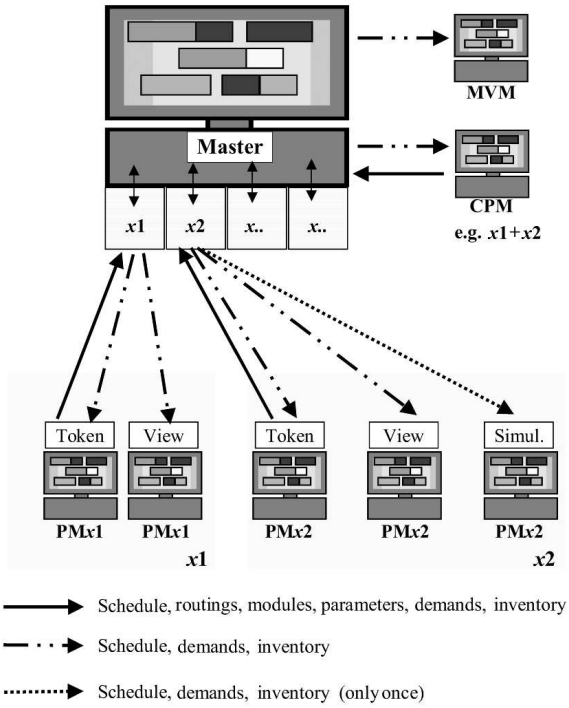

Figure 3. Collaborative planning system architecture

The architecture consists of the following modules, which are briefly explained in the following:

- *master production model* (*MM*),
- *plant production models* (*PMx*),

- *master view model* (*MVM*) for monitoring,
- *chain planner models* (*CPM*s),
- *communication mechanism* among these models, and
- *login and authorization concept* for data consistency and updates.

3.2.1 *Master Production Model* The *master production model* (*MM*) contains information related to all the master data and the current integrated schedule from all the plants called *master schedule* (*MS*). This model is updated during the night for the latest data related to (*i*) products, (*ii*) bills of material (BOMs), (*iii*) demands, (*iv*) material receipts, and (*v*) inventory positions from the ERP system.

MM is used for the nightly optimization run and for rolling the production schedule forward each night. Then, copies of plant specific models are created from it for manual changes. This ensures that the models are synchronized at the beginning of each day. During the day, *MM* serves as a focal point to receive and send latest changes in the schedule and master data in each of the individual plants. This model is used for communication from and between the plant and chain scheduler(s). Any changes made within the plant models or in the chain planner models are broadcast back to *MM*. By collecting all the changes during the day, it provides the latest information on the production schedule and the master data for all the involved plants. It thus acts as a central data repository which always holds the most up–to–date information. By receiving the information from *MM*, the plant model remains synchronized and the data consistency among the models is maintained. The latest information on the production schedule in the other plants and other changes in the dynamic data are also received from *MM*.

3.2.2 *Plant Production Model* The *plant* (*production*) *model* for plant x (*PMx*) contains information related to the schedule of plant x and the relevant master data. Apart from the detailed information on the production schedule for plant x, it also contains production schedule information published by all the other plants $x' \neq x$. The *PMx* are generated daily from the *MM* after re–optimizing *MS*. During the day the plant planner updates the *PMx* on the local client machines.

Any plant schedule considers the resources of the current plant x. The production schedules from other plants are shared using a set $\mathcal{F}_v$ of *virtual facilities*. For each product p produced in plant x, a virtual facility $f_p \in \mathcal{F}_v$ is introduced in plant x. f_p depicts the production schedule information for product p in plant x. Using the virtual facilities, the production schedule is shared without revealing the exact information about the real facilities used within the plant.

Any schedule or master data relevant changes within a *PMx* are broadcast to *MM*. Other plant planners from the same plant work with the updated *PMx* containing the latest master data. This way it is avoided that for every change in the plant master data a completely new *PMx* is created from *MM*. This improves the system performance without loosing data consistency.

3.2.3 *Master View Model* The *master view model* (*MVM*) is a reduced model for "view only" purposes allowing a simplistic overview of the integrated production schedule. It involves only the virtual facilities, and it is used by the chain scheduler to monitor the production schedule spanning across all the plants. During the day, the model is constantly updated by receiving the changes in the production schedule from *MM*. No changes are broadcast back to *MM*. *MVM* is created at the end of a nightly batch job along with the *PMx*.

3.2.4 *Chain Planner Model* The *chain planner model* (*CPM*) is a model for a virtual plant created from the current *MM* involving a selected subset of plants. The model involves all the master data and the production schedules for the selected plants with all the resources in those plants. *CPM* receives the most current schedule from *MM*, and any schedule changes within *CPM* can be published back to *MM*. There can be more than one *CPM* with different subsets of plants involved. The *CPM* supports schedule modifications involving chain products and the affected plants and thereby helps to resolve conflicts.

3.2.5 *Communications Mechanism* The communication mechanism among *MM*, the *PMx* and the *CPM*s involves a series of complex macros using remote function calls to send and receive the information. The type of data that are exchanged between the models can be broadly classified into *master data*, *schedule relevant* or *dynamic data*, and *authorization and data consistency relevant data*.

3.2.6 *Login and Authorization Concept* The architecture allows more than one scheduler in a plant to monitor and analyze the production schedule. Even a single user can work with multiple copies of the same model. To maintain data consistency and to avoid conflicts, the concept of *token* was introduced. A token is assigned to each plant in the integrated *MM*. The process that is currently allowed to broadcast information is the one which has the token. This concept ensures that at any given point in time only one process per plant is authorized to broadcast.

4 Manual Planning Interaction

4.1 *Basics*

System–generated schedules need frequent manual changes. There are many reasons for such manual changes, such as the breakdown of an equipment, the acceptance of an unexpected customer order, or the necessity to rework some production output. To handle such changes and to analyze their consequences, an APS system has to allow manual interaction in the plant. In the industrial application considered, the commercially available APS tool AspenMIMI© from

AspenTech was used to represent the schedule and to enable manual interaction. It allows a high degree of freedom in customizing and development due to its toolbox structure with application specific programming. The tool provides a customizable planning board in form of a Gantt chart and built–in scheduling functions to support the development of an application–specific planning and scheduling system.

The built–in support functions provided by the APS tool are used both to analyze the schedule and to enable manual interaction functions. The most important built–in functions are

- the planning board with a high flexibility for customizing,
- forward and backward tracing of production activities, which allows the user to follow the production sequences and material flow, and
- multi–level BOM explosion, which enables viewing the material consumption even between the plants.

With the support of these scheduling functions, an APS tool for manual user interaction was developed. Several utilities provided by this tool are briefly described below.

4.2 *Utilities*

4.2.1 *ATP and CTP* The *ATP* and *CTP utilities* allow the user to inquire about the availability of a product. The user can check the unallocated inventory of a product around a specified date or inquire about the earliest date for the availability of a specified quantity. If the product is available at the requested date, an order can be placed to deliver the product exclusively from inventory (ATP). If the unallocated inventory is not sufficient, the user can inquire for a capacity check to produce the missing quantity (CTP). Provided that there is enough capacity, the order can be placed, and the corresponding chain can be planned. A major restriction of this utility is that a planner is allowed to use it only for products or chains which do not involve other plants. This avoids conflicts arising from inter–plant planning. The ATP/CTP utility brings benefits both as an immediate query tool and as a simulation help.

4.2.2 *Manual Update of Current Production* The automatic *roll–forward utility* updates the plan overnight and rolls the date forward by one day. However, this is based on standard production parameters. The planner revises the status of the current chainlinks using the *manual update utility* which allows the planner to enter the actual production situation into the schedule. This update of the production is an essential part of the daily manual planning in the plant and also allows a better inspection of the problems in the integrated schedule and of the conflicts between the plants.

4.2.3 *Drag and Drop of Activities* Perhaps one of the most important features of a manual planning tool is the so–called *drag and drop* of production activities in the Gantt chart. However, such a simple mouse action has its consequences on the capacity utilization and material availability. Therefore, the developed *post–procedures* of moving an activity on the planning board from one facility to another or from one time slice to another check the availability of capacity at the target facility and target time slice. In addition, available routings, possible parameter changes and requirements for setup and cleanup are considered during a drag and drop action to ensure the feasibility of the resulting schedule.

4.2.4 *Routing Data Maintenance* The *routing data maintenance utility* provides a set of user friendly matrix–based interfaces which help the planner to create or modify routings, the facilities involved, and the related production parameters. The aim is to keep this vital information as compact and simple as possible for processing and easily accessible for the planner.

4.2.5 *Batch Identification* Each batch of a campaign gets a number which will stay with it from the actual production until it leaves the shop floor. Batch numbers assigned to a campaign are considered when modifying production schedule with other utilities such as drag and drop to ensure that batches still have ascending numbers. *Batch identification* is essential to tracking batches from production to shop floor.

Conclusions

The paper has demonstrated a practical way of approaching a complex scheduling problem in the chemical process industry involving batch production. The successful implementation has lead to both quantitative and qualitative benefits. It has increased overall customer satisfaction and stability without sacrificing flexibility in production schedule changes. Moreover, it has reduced conflicts, and improved communication among different business functions.

A tailored optimization algorithm has been developed to handle a network of production plants with interdependent production schedules, multi–stage production with multi–purpose facilities and chain production. The solution procedure tends to (*i*) improve the utilization of available resources, (*ii*) reduce projected inventories, and (*iii*) minimize due date violations. The algorithm considers production schedules of all plants involved simultaneously and thus provides a better overall solution than one obtained by individually optimizing production schedules.

A collaborative planning model allows (*i*) transparency, (*ii*) shared information on production schedule, (*iii*) reduction in response time for reacting to any changes upstream or downstream, (*iv*) system support for conflict management, (*v*) monitoring and analyzing the latest production schedule of the integrated model,

and (*vi*) better use of globally available inventory. The custom–built tool for manual interaction provides (*i*) user friendly utilities for changes in the production schedules, (*ii*) a useful visualization tool for monitoring and analyzing plant specific production schedules, (*iii*) a support for generating multiple what–if scenarios, (*iv*) answers to marketing queries, and (*v*) a system representing the current situation in the production plant.

It has been observed that all three aspects, namely (*i*) a high quality solution obtained from the *optimizer*, (*ii*) a mechanism for *collaborative planning*, and (*iii*) a user friendly tool for *manual interaction*, were needed within an integrated framework to realize all the above mentioned benefits and to ensure user acceptance of the APS tool. Concentrating on any one of these aspects and achieving the best result in it but ignoring the others would have lead to a failure in realizing the full potential for improvements.

Acknowledgements

The authors would like to thank Hans–Otto Günther and Roland Heilmann for helpful comments on draft versions of this paper.

References

Blömer, F., and H. O. Günther (1998). Scheduling of a multi–product batch process in the chemical industry. *Computers in Industry*, **36**, 245-259.

Blömer, F., and H. O. Günther (2000). LP–based heuristics for scheduling chemical batch processes. *International Journal of Production Research*, **38**, 1039-1051.

Burkard, R. E., M. Hujter, B. Klinz, R. Rudolf, and M. Wenning (1998a). A process scheduling problem arising from chemical production planning. *Optimization Methods & Software*, **10**, 175-196.

Burkard, R. E., M. Kocher, and R. Rudolf (1998b). Rounding strategies for mixed integer programs arising from chemical production planning. *Yugoslav Journal of Operations Research*, **8**, 9-23.

Garcia–Flores, R., and X. Z. Wang (2002). A multi–agent system for chemical supply chain simulation and management support. *OR Spectrum*, **24**, 343-370.

Gen, M., and R. Cheng (2000). *Genetic Algorithm and Engineering Optimization*. Wiley, New York et al.

Grunow, M., H. O. Günther, and M. Lehmann (2002). Campaign planning for multi–stage batch processes in the chemical industry. *OR Spectrum*, **24**, 281-314.

Günther, H. O., and H. Tempelmeier (2000). *Produktion und Logistik*. Springer, Berlin et al. (in German)

Hughes, C., and T. Hughes (1997). *Object–oriented multithreading using C++*. Wiley Computer Publishing, New York et al.

Kallrath, J. (2002a). Planning and scheduling in the process industry. *OR Spectrum*, **24**, 219-250.

Kallrath, J. (2002b). Combined strategic and operational planning — an MILP success story in chemical industry. *OR Spectrum*, **24**, 315-341.

Knolmayer, G., P. Mertens, and A. Zeier (2000). *Supply Chain Management auf Basis von SAP–Systemen*. Springer, Berlin et al. (in German)

Kolisch, R., and S. Hartmann (1998). Heuristic algorithms for solving the resource–constrained project scheduling problem — classification and computational analysis. In J. Weglarz (Ed.), *Handbook on Recent Advances in Project Scheduling*. Kluwer, Dordrecht. pp. 147-178.

Kolisch, R., and R. Padman (2001). An integrated survey of deterministic project scheduling. *Omega - The International Journal of Management Science*, **29**, 249-272.

Mendez, C. A., and J. Cerda (2002). A MILP–based approach to the short–term scheduling of make–and–pack continuous production plants. *OR Spectrum*, **24**, to appear.

Monma, C., and C. Potts (1989). On the complexity of scheduling with batch setup times. *Operations Research*, **37**, 798-804.

Neumann, K., C. Schwindt, and N. Trautmann (2002). Advanced production scheduling for batch plants in process industries. *OR Spectrum*, **24**, 251-279.

O'Leary, D. E. (2000). Supply chain processes and relationships for electronic commerce. In M. Shaw, R. Blanning, T. Strader, and A. Whinston (Eds.), *Handbook on Electronic Commerce*. Springer, Berlin et al. pp. 431-444.

Pinedo, M. (1995). *Scheduling - Theory, Algorithms, and Systems*. Prentice Hall, Englewood Cliffs, NJ.

Schumann, A. (1997). SAP–R/3 in process industries: expectations, experiences and outlooks. *ISA Transactions*, **36**, 161-166.

Stadtler, H., and C. Kilger (Eds.) (2000). *Supply Chain Management and Advanced Planning*. Springer, Berlin et al.

Tan, G. W., M. J. Shaw, and W. Fulkerson (2000). Web–based global supply chain management. In M. Shaw, R. Blanning, T. Strader, and A. Whinston (Eds.), *Handbook on Electronic Commerce*. Springer, Berlin et al. pp. 457-478.

Timpe, C. (2002). Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, **24**, to appear.

Westenberger, H., and J. Kallrath (1994). Formulation of a job shop problem in process industry. *Unpublished Working Paper*. Bayer AG, Leverkusen, and BASF AG, Ludwigshafen.

Yang, R. (2000). Supply chain management: developing visible design rules across organizations. In M. Shaw, R. Blanning, T. Strader, and A. Whinston (Eds.), *Handbook on Electronic Commerce*. Springer, Berlin et al. pp. 445-456.