A Projection-Based Method for Production Planning of Multiproduct Facilities

Charles Sung and Christos T. Maravelias
Dept. of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, WI 53706

DOI 10.1002/aic.11845
Published online July 27, 2009 in Wiley InterScience (www.interscience.wiley.com).

An algorithm is presented for identifying the projection of a scheduling model's feasible region onto the space of production targets. The projected feasible region is expressed using one of two mixed-integer programming formulations, which can be readily used to address integrated production planning and scheduling problems that were previously intractable. Production planning is solved in combination with a surrogate model representing the region of feasible production amounts to provide optimum production targets, while a detailed scheduling is solved in a rolling-horizon manner to define feasible schedules for meeting these targets. The proposed framework provides solutions of higher quality and yields tighter bounds than previously proposed approaches. © 2009 American Institute of Chemical Engineers *AIChE J,* 55: 2614–2630, 2009
Keywords: production planning, scheduling, mixed-integer programming, projection

Introduction

Recent trends in the chemical industry toward product customization and diversification have led to multiproduct facilities that are often complex process networks (batch mixing/splitting, recycle streams) with multiple shared resources. At the same time, facilities must remain able to respond to demand fluctuations while existing assets should be utilized close to their capacity. Production planning of heavily loaded units subject to complex operational constraints and demand fluctuations is a challenging task in which production targets often need to be set close to system limits.

To solve such problems effectively, it should be determined what production targets are feasible. This is traditionally accomplished using production planning methods that incorporate scheduling.[1,2] Using basic process information, scheduling models are able to explicitly model complex interactions involved in "how" production occurs. This allows them to accurately but implicitly answer "whether" specific production amounts can be produced (i.e., are feasible). The natural goal, then, is to extract feasibility information from scheduling models in such a way that (a) maintains the accuracy of detailed scheduling, yet (b) encourages a compact formulation that adds minimal computational burden to the production planning model.[3]

Sung and Maravelias[4] proposed conveying feasibility and cost information using a surrogate model to provide a convex approximation of the scheduling feasible region and an underestimation of production cost. That work is extended here to models capable of capturing and representing nonconvex regions. This article is organized as follows: The production planning problem statement is posed, followed by a motivating example and review of previously proposed methods in which the production planning problem incorporates scheduling models. Two new mixed-integer programming (MIP) formulations are then introduced for representing nonconvex regions as either the difference or union of polytopes/polyhedra. Next, we present the proposed algorithm for the identification of the projection of the scheduling feasible region onto the production planning variables. A rolling horizon scheme follows as part of a framework in which the surrogate model may be used to hierarchically decompose integrated production planning and scheduling problems. We conclude with an example problem in which integration of production planning with surrogate models generated by our algorithm is shown to be both accurate and computationally efficient.

Correspondence concerning this article should be addressed to C. T. Maravelias at maravelias@wisc.edu

Background

Production planning problem statement

The goal in production planning is to meet customer demand at minimum total cost (production + holding + unmet/backlog) over a medium-term horizon. Given are:

- A fixed planning horizon divided into planning periods $t \in \{1, 2, \ldots, T\}$ of uniform duration.
- A set of products $k \in K$.
- Demand $d_{k,t}$ for product k at the end of time period t.
- Holding cost h_k and penalty u_k for unmet demand for product k.
- Process capacities.
- Production costs.

Decision variables include the following:

- Production amount $P_{k,t}$ of product k in period t.
- Inventory level $I_{k,t}$ of product k at the end of period t.
- Unmet demand $U_{k,t}$ for product k in period t.

A general formulation for production planning is given by Eqs. 1–5. Resource constraints, which determine whether specific production amounts are feasible, are modeled by generic function $f(P_{k,t})$ in Eq. 2, while total production cost Cp_t is modeled by generic function $g(P_{k,t})$ in Eq. 3. Inventory balance for product k at the end of period t is maintained in Eq. 4.

$$\min_{Cp_t, I_{k,t}, P_{k,t}, U_{k,t}} \quad \sum_t Cp_t + \sum_{k \in K, t} \left(h_k I_{k,t} + u_k U_{k,t} \right) \tag{1}$$

$$f(P_{k,t}) \leq 0 \quad \forall t \tag{2}$$

$$Cp_t = g(P_{k,t}) \quad \forall t \tag{3}$$

$$I_{k,t} = I_{k,t-1} + P_{k,t} - d_{k,t} + U_{k,t} \quad \forall k, t \tag{4}$$

$$I_{k,t}, P_{k,t}, U_{k,t} \geq 0 \quad \forall k, t \tag{5}$$

To solve production planning problems optimally, it is necessary to account accurately for resource constraints and production costs, which can be achieved via the integration of scheduling formulations in Eqs. 2 and 3. Our goal is to develop computationally tractable and accurate approximations of functions $f(P_{k,t})$ and $g(P_{k,t})$ that can replace computationally expensive scheduling models.

Motivating example

Suppose we would like Eq. 2 to model the process network shown in Figure 1a. Raw materials are converted by two parallel units into products A and B. The two units can operate at up to 1 kg/h ("slower" unit) and 3 kg/h ("faster" unit), respectively. For both units, 1 h of setup is required to produce A and 1 h of setup is required to produce B. Over a period of 7 h, the slower unit can produce up to 6 kg of product A only, 6 kg of product B only, or 5 kg of both products. Figure 1b visualizes this as the slower unit's "feasible region," a representation of achievable production amount combinations of A (variable P_A) and B (variable P_B). Figures 1c,d show the feasible region for the faster unit only and for both units, respectively. To emphasize, even basic combinations of simple processes are enough to bring about feasible regions of nonobvious shape, which can be accurately represented using complex MIP scheduling models which use additional "nonplanning" variables. Thus, Figure 1d can also be described as the "feasible region of the scheduling model projected onto the space of production amounts." Starting from a convex over-approximation that contains every true feasible point (Figure 1e), our goal is to identify the infeasible points that lie along the top-right boundary of Figure 1e and express them in a compact form.

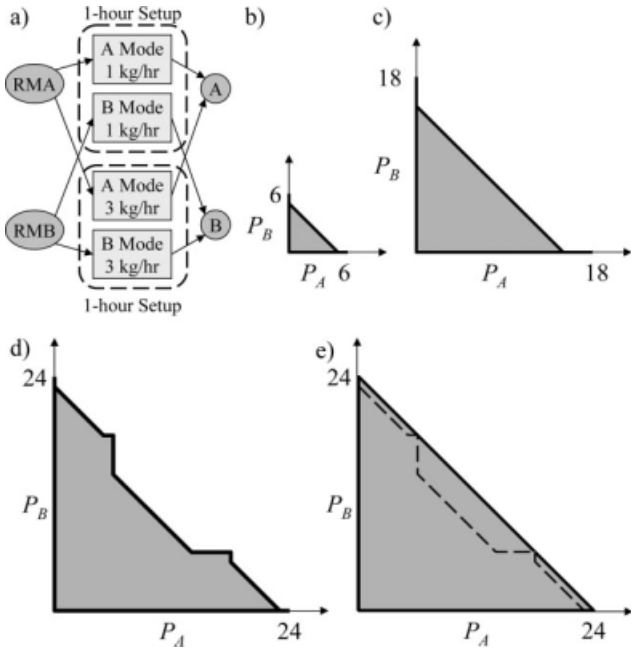

Figure 1. (a) Process network of the motivating example. Feasible region using: (b) the slower unit only; (c) the faster unit only; (d) both units. (e) Tightest possible convex over-approximation (dark lines) of the feasible region using both units.

Integration of production planning and scheduling

Scheduling models can be classified as: (i) network-based formulations for general processes or (ii) sequence-based formulations for multi- and single-stage processes. Network-based formulations can be further classified into discrete-time formulations[5] where the time horizon is divided a priori, possibly into equal periods; continuous-time formulations[6–10] where the time horizon is partitioned as part of the optimization; and mixed-time formulations[11] where the time grid is fixed but the durations of the tasks are variable. Sequence-based formulations can be further classified into slot-based,[12] unit-specific grid,[13] global-precedence,[14] and immediate-precedence[15,16] formulations for standalone scheduling, and simultaneous batching-scheduling approaches.[17,18] In theory, using detailed scheduling models allows optimal production targets to be identified. However, this results in very large models that are impractical to solve to optimality.

The most popular alternative approach is to replace a detailed scheduling model with a simplified model that is obtained through removing subsets of constraints (relaxation) or through combining variables (aggregation). A common relaxation scheme is to keep job assignment constraints and

variables but to discard sequencing constraints and variables.[19–21] Wilkinson et al.[22] aggregated time periods in a scheduling model, and Vancza et al.[23] aggregated project activities. Birewar and Grossmann[24] proposed using an aggregate traveling salesman problem for sequencing batches within a multiperiod planning model to estimate time requirements for production planning of parallel units in a flowshop plant. Wellons and Reklaitis[25] proposed identifying cyclic campaigns with dominant production rates.

Another approach is to replace a detailed scheduling model with a surrogate model which is created through online or offline[26,27] analysis. Sung and Maravelias[4] proposed using the convex hull of a scheduling model's feasible region projected onto production amount. To determine this region which they called the production attainable region[48,49] (PAR), they presented an offline algorithm based on the direct investigation of the scheduling model's feasible region. A convex over-approximation and a convex under-approximation, both polytopes, were set up and iteratively converged. If the two approximations did not exactly converge, then the over-approximation is declared the convex approximation of projection (CAP). The over-approximation can be used to solve the production planning problem, while the under-approximation can serve as a repository of known-to-be-feasible solutions. In this article, we extend that work to nonconvex regions.

Methods for integrating production planning with scheduling can also be classified by their solution strategy into: (a) hierarchical methods, (b) iterative methods, and (c) full space methods. The first two types of methods decompose the integrated problem into a higher-level problem and a lower-level subproblem that are solved separately, while the third type attempts to solve the integrated problem as a single model.

In hierarchical methods, the higher-level model is solved to make key decisions such as setting of production targets, selection of tasks, and assignment of tasks to processing units.[28] The lower-level model is then solved based off of these decisions. Usually, the higher-level model is provided a substitute for the lower-level model.[29] Hierarchical methods are straightforward and tractable relative to full space methods. However, the quality of a hierarchical solution is contingent on the accuracy of the substitute used.

In iterative methods, the higher-level model is provided informational feedback in addition to any substitute that might have been used by hierarchical methods. Iterative methods are set up such that each solution of the lower-level model improves the quality of the substitute model, for example by adding a linear inequality,[30] an integer cut,[21,31] or a generated column.[32] A common scheme is to ensure that the substitute is an over-approximation that is tightened in every iteration; this allows solutions of the higher-level model to act as a monotonically improving bound.

In full space methods, the integrated production planning and scheduling problem is treated as a single model to be solved using standard mathematical programming methods,[33,34] heuristic methods such as simulated annealing[35] or genetic algorithms,[36] or decomposition methods that exploit the structure of the model. Examples of full space decomposition approaches include decomposition by production planning item, resource, or time period. Finally, rolling-horizon methods have also been proposed as a means to address larger instances.[37,38,50]

Basic Concepts

Before presenting our algorithm, we briefly overview some concepts involving polytopes/polyhedra and difference/union thereof. We will be using combinations of polytopes/polyhedra to represent nonconvex regions, which will be ultimately expressed as a surrogate model using one of two MILP formulations presented in this section.

Polyhedra, polytopes, and facets

The solution of a linear inequality is a half space. The solution of a set of linear inequalities, i.e., the intersection of a set of half spaces, is a polyhedron. A bounded polyhedron may be called a polytope. A polytope can be defined using H-representation by a set of linear inequalities $l \in L$. Each linear inequality l may be expressed in the form $\sum_{k\in K} \omega_k^l P_k \le \Phi^l$. Where practical, $\boldsymbol{\omega}^{\boldsymbol{l}} = [\omega_A^l, \omega_B^l, ...]^{\mathrm{T}}$ is normalized to length one. A point $P = [P_A, P_B, \ldots]$ is within the polytope if and only if it satisfies every constraint $l \in L$.

H-representation:

$$\sum_{k\in K} \omega_k^l P_k \le \Phi^l \quad \forall l \in L$$

A polytope can also be defined using V-representation by a set of vertices $v \in V$ with coordinates P_k^v. A point is within the polytope if and only if it can be expressed as an affine combination of its vertices with weights between 0 and 1.

V-representation:

$$P_k = \sum_{v\in V} P_k^v \lambda^v \quad \forall k \in K$$
$$\sum_{v\in V} \lambda^v = 1$$
$$0 \le \lambda^v \le 1 \quad \forall v \in V$$

A number of software packages[39,40] can convert between H- and V-representation. In addition, a polytope in n dimensions may be defined by its $n-1$ dimensional "sides" called facets defined by exactly n points. Given the n points defining a facet we can determine the corresponding inequality (see Appendix A).

Difference and union of polytopes

Using only continuous variables, linear inequalities may be combined to formulate their intersection, which will be a convex polyhedron. Once binary variables are introduced, other set theoretic functions may be formulated such as union and difference, which can be nonconvex. Recall that the CAP is a polytope, so every feasible point must satisfy every CAP constraint $l \in L^{\mathrm{CAP}}$.

CAP:

$$\sum_{k\in K} \omega_k^l P_k \le \Phi^l \quad \forall l \in L^{\mathrm{CAP}}$$

The first formulation approximates the projection of the feasible region as a difference, of the CAP and the union of

known-to-be-infeasible polyhedrons $ip \in IP$. Hence, point P is formulated as feasible if it is in the CAP (satisfies every constraint $l \in L^{\mathrm{CAP}}$) but not in any infeasible polyhedron $ip \in IP$ (violates at least one constraint l in every set L^{ip}). Binary variable Z^l is introduced such that if it is active (=1), then violation will occur: $\sum_{k\in K} \omega_k^l P_k \geq \Phi^l$. Point P is prevented from being in infeasible polyhedron ip by requiring at least one binary variable Z^l in $l \in L^{ip}$ to be active.

NCP1:

$$\sum_{k\in K} \omega_k^l P_k \leq \Phi^l \quad \forall l \in L^{\mathrm{CAP}}$$
$$\sum_{k\in K} \omega_k^l P_k \geq \Phi^l - M^l(1 - Z^l) \quad \forall ip \in IP,\ l \in L^{ip}$$
$$\sum_{l\in L^{ip}} Z^l = 1 \quad \forall ip \in IP$$
$$Z^l \in \{0,1\} \quad \forall ip \in IP,\ l \in L^{ip}$$

The second formulation re-expresses the nonconvex projection as the union of feasible polytopes $fp \in FP$. Each feasible polytope fp is expressed in V-representation in terms of vertices $v \in V^{fp}$ whose coordinates are P_k^v. Point P is in feasible polytope fp if binary variable Z^{fp} is active (=1) and P can be expressed as a convex combination of fp's vertices.

NCP2:

$$P_k = \sum_{fp\in FP} \sum_{v\in V^{fp}} P_k^v \lambda^v \quad \forall k \in K$$
$$\sum_{v\in V^{fp}} \lambda^v = Z^{fp} \quad \forall fp \in FP$$
$$\sum_{fp\in FP} Z^{fp} = 1$$
$$0 \leq \lambda^v \leq 1 \quad \forall fp,\ \in FP,\ v \in V^{fp}$$
$$Z^{fp} \in \{0,1\} \quad \forall\, fp \in FP$$

Although both models are meant to formulate the same region, both are developed because one may be superior to the other based on context, application, or shape of the feasible region.

Merging routine

Suppose a feasible region is modeled as the union of feasible polytopes in H-representation. Geyer[41] describes this in terms of Boolean logic: Point P is feasible if it is in one of several feasible polytopes $fp \in FP$ (OR); it is in a specific polytope fp if it satisfies every one of that polytope's constraints $l \in L^{fp}$ (AND). Hence, strategies associated with Boolean logic-minimization may be adapted to minimizing the number of feasible polytopes necessary to express a nonconvex feasible region. All constraints (logic conditions) used to define input polytopes are pooled together; they are then either redistributed among output polytopes or else discarded. In other words, constraints (logic conditions) are rearranged or discarded, but not created. Output polytopes are allowed to overlap each other. This is implemented in the Multi-Parametric Toolbox[40] that we will refer to as "the merging routine." Note that this subsection has been presented in terms of merging *feasible* polytopes; however, we will be using it for merging *infeasible* polytopes as well.

Related to this work is also the work of Goyal and Ierapetritou[42] who proposed an algorithm for approximating the nonconvex feasible region of NLP models, as well as methods for the identification and approximation of a function's nonconvex surface ("implicit surface," "iso-surface"). Ning and Bloomenthal[43] evaluated surface tiling methods, specifically sampling of a regular lattice followed by cubical cell polygonization. Van Overveld and Wyvill[44] proposed the "Shrinkwrap" algorithm, which generates and tightens a triangular mesh. Finally, disjunctive programming methods[45,46] can also be used to represent nonconvex regions.

Proposed Algorithm

A five-phase algorithm is presented for the identification of the projection of the feasible region F^{SM} of a scheduling model onto the space of production amounts and the expression of this projection in the form of model (NCP1) or model (NCP2). The proposed algorithm is independent of how the scheduling model is formulated. The algorithm is presented here for $n = |K| = 2$; however, it may be generalized to higher dimensions. To overview: Given the facets of the CAP, Phase I identifies a series of infeasible triangles that surround but do not overlap the CAP. Phase II splits and expands individual infeasible triangles so that as a group they encroach onto the CAP. Phase III merges all infeasible triangles into fewer infeasible polytopes. Phase IV converts these into infeasible polyhedrons usable by model (NCP1), and Phase V in turn converts these into feasible polytopes usable by model (NCP2).

Phase I

Phase I uses the algorithm presented in Sung and Maravelias[4] to provide the CAP, an over-approximation of the feasible region, which is also used as a first approximation of the feasible region's boundary. Each facet of the CAP is defined by n vertices (points). Furthermore, each facet is assigned an "external point" outside the CAP such that together they form an "infeasible triangle" defined by $n + 1$ points. This is shown in Figure 2 for a generic facet whose $n = 2$ points are labeled A and B. The external point is labeled X, and the resulting infeasible triangle is denoted by AB-X. The interior of AB-X is meant to be infeasible, as is point X, segment AX, and segment BX. The feasibility of segment AB is

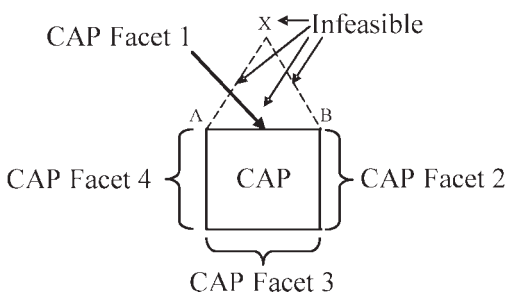

Figure 2. A CAP is shown with four facets.

CAP Facet 1 is used to create infeasible triangle AB-X.

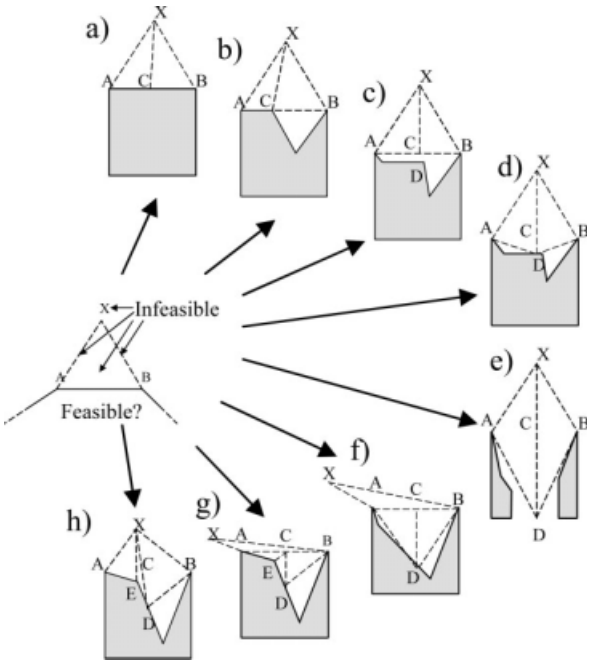

Figure 3. Some ways to split/expand an infeasible triangle.

ambiguous pending future investigation by the algorithm. CAP facets are used to seed a list of infeasible triangles to be investigated. Phase I ends with a series of infeasible triangles that surround but do not overlap the CAP.

Phase II

The purpose of Phase II is to "expand" the area covered by all infeasible triangles. This is accomplished by iteratively choosing a single infeasible triangle (generically referred to as "*AB-X*") and investigating its facet side (segment *AB*). Figure 3 shows some ways in which *AB-X* can be modified, depending on what is revealed by investigation of the feasible region. Phase II can be broken down into the steps shown in Figure 4.

(1–2) We begin each iteration by choosing the largest-sized facet on the list. The algorithm terminates if: an iteration limit (not shown in Figure 4) has been reached; the list is empty; or the largest facet size on the list is less than tolerance δ_1.

(3) Model (T1) is solved to search for a feasible point C between A and B, where C is required to be in the feasible region of the scheduling model $(P_1^C, P_2^C) \in F^{\mathrm{SM}}$.

Model (T1):

$$\max_{P_k^C, \lambda^A, \lambda^B} \mathrm{obj}$$
$$P_k^C = P_k^A \lambda^A + P_k^B \lambda^B \quad \forall k \in K$$
$$\lambda^A + \lambda^B = 1$$
$$\delta_2 \le \mathrm{obj} \le \lambda^A \le 1$$
$$\delta_2 \le \mathrm{obj} \le \lambda^B \le 1$$
$$(P_1^C, P_2^C) \in F^{\mathrm{SM}}$$

If model (T1) is feasible, then a feasible point has been found and it can be labeled C. The preferred choice is the midpoint between A and B, where obj $= 1/n = [1/2]$ (see Figure 3a). If this is not feasible, then the next closest feasible point is selected (see Figure 3b). If a feasible point is found with obj $\ge \delta_2 > 0$, then *AB-X* is replaced with *AC-X* and *BC-X* and the algorithm returns to Step 1. A positive tolerance is needed to avoid selection of A (obj $= 0$) or B (obj $= 0$).

If, however, no feasible point can be found with obj $\ge \delta_2$, then the facet is declared empty. The midpoint between A and B is labeled C, despite being infeasible, and model (T2) is solved to identify the nearest feasible point D on a perpendicular line originating from C (defined by vector $\omega^{AB\text{-}X}$).

Model (T2):

$$\min_{P_k^D} \mathrm{obj}$$
$$P_k^D = P_k^C + \omega_k^{AB-X} \mathrm{obj} \quad \forall k \in K$$
$$\mathrm{obj} \ge 0$$
$$(P_1^D, P_2^D) \in F^{\mathrm{SM}}$$

Solving model (T2) leads to one of the following cases: model (T2) is feasible and obj is less than positive tolerance δ_3; model (T2) is feasible and obj $\ge \delta_3$; model (T2) is infeasible. For the first case, $0 \le \mathrm{obj} < \delta_3$ indicates that feasible point D is very close to infeasible point C, thus *AB-X* is replaced with *AC-X* and *BC-X* (see Figure 3c), and the algorithm returns to Step 1. For the other two cases, the algorithm continues to Step 4.

(4) For the second case, where model (T2) is feasible and obj $\ge \delta_3$, feasible point D will be used in a later step to expand *AB-X* (see Figure 3d). For the third case, where model (T2) is infeasible, no feasible point can be found on the perpendicular line originating from C. Consequently, model (T3) is solved to locate whatever point is on the far side of the CAP (see Figure 3e). That point, even though it is infeasible, is labeled point D.

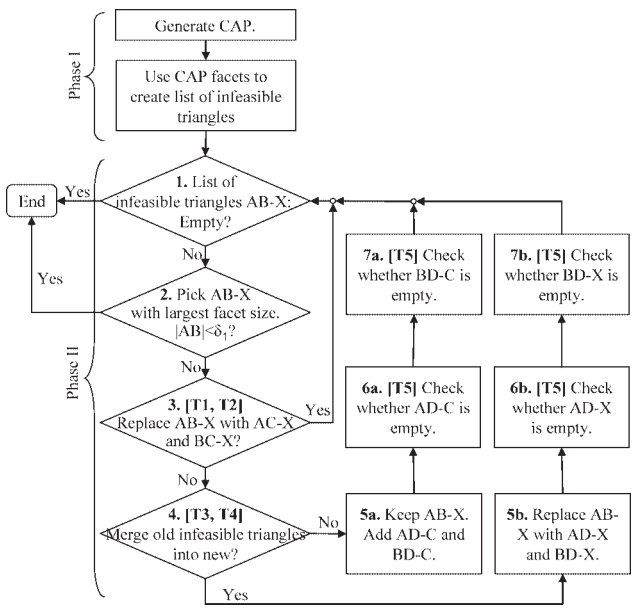

Figure 4. Phase I and Phase II flowsheet.

Model (T3):

$$\max_{P_k^D} \text{obj}$$
$$P_k^D = P_k^C + \omega_k^{AB-X}\text{obj} \quad \forall k \in K$$
$$\text{obj} \geq 0$$
$$\sum_{k\in K} \omega_k^l P_k^D \leq \Phi^l \quad \forall l \in L^{\text{CAP}}$$

Two new infeasible triangles are created: *AD-C* and *BD-C*. To slow growth in total number of infeasible triangles, it may be desirable to merge these two new triangles with *AB-X* to produce *AD-X* and *BD-X*. First, however, model (T4) should be solved to indirectly check whether the union of "*AB-X*, *AD-C*, and *BD-C*" is the same as the union of "*AD-X* and *BD-X*". Model (T4) is a system of equations that can be solved to express D in terms of P^A, P^B, and P^X. Note that λ^X will be negative due to D and X being on opposite sides of segment *AB*.

Model (T4):

$$P_k^D = P_k^A\lambda^A + P_k^B\lambda^B + P_k^X\lambda^X \quad \forall k \in K$$
$$\lambda^A + \lambda^B + \lambda^X = 1$$

If λ^A is negative (D is "outside" of *BX*) or λ^B is negative (D is "outside" of *AX*, as in Figure 3f), then the algorithm proceeds to Step 5a. If λ^A and λ^B are both positive (e.g., Figures 3d,e), then the algorithm proceeds to Step 5b.

(5a) *AB-X* is kept, however it is removed from the list of infeasible triangles to be investigated. Infeasible triangles *AD-C* and *BD-C* are added, and the algorithm proceeds to Step 6a.

(5b) *AB-X* is deleted. Infeasible triangles *AD-X* and *BD-X* are added, and the algorithm proceeds to Step 6b.

(6a) Model (T5) is solved to check for any feasible point E that may be in *AD-C*.

Model (T5):

$$\max_{P_k^E,\lambda^A,\lambda^D} \lambda^C$$
$$P_k^E = P_k^A\lambda^A + P_k^D\lambda^D + P_k^C\lambda^C \quad \forall k \in K$$
$$\lambda^A + \lambda^D + \lambda^C = 1$$
$$0 \leq \lambda^A, \lambda^D, \lambda^C \leq 1$$
$$(P_1^E, P_2^E) \in F^{\text{SM}}$$

Model (T5), by maximizing weight λ^C, implicitly maximizes the distance of feasible point E from segment *AD*. This distance can be calculated a posteriori as viol $= \Phi^{AD\text{-}C} - \omega^{AD\text{-}C}E$. If viol is greater than δ_3, then *AD-C* is replaced with *AE-C* and *DE-C* (see Figure 3g), otherwise E is ignored. The algorithm proceeds to Step 7a.

(6b) Model (T5) is solved to check for any feasible point E that may be in *AD-X*. Even though model (T5) is written for *AD-C* instead of *AD-X*, we already know that the triangular region formed by *A*, *C*, and *X* is infeasible due to the interior and sides of *AB-X* being infeasible. If viol is greater than δ_3, then *AD-X* is replaced with *AE-X* and *DE-X* (see Figure 3h), otherwise E is ignored. The algorithm proceeds to Step 7b.

(7a) A variation of Step 6a is solved to check for any feasible point that may be in *BD-C*.

(7b) A variation of Step 6a is solved to check for any feasible point that may be in *BD-X*.

As presented, the algorithm assumes that infeasible triangles created by Steps 6–7 do not contain any feasible points. This may be verified by repeating Steps 6a for each newly created infeasible triangle. To recap, all parts of the CAP are initially assumed feasible. Infeasible triangles are introduced so that regions with the CAP may be identified, via overlap, as infeasible. The purpose of Phase II, then, is to "expand" the original set of infeasible triangles such that overlap of the CAP progressively increases. The nonoverlapped portion converges toward the feasible region that we are trying to identify, approximate, and formulate a surrogate model for.

Phase III

Phase III uses one round of preprocessing followed by two rounds of merging to combine infeasible triangles into fewer infeasible polytopes. As the purpose of infeasible triangles/polytopes is to indicate infeasible regions within the CAP, infeasible triangles that do not overlap the CAP may be deleted without consequence after Phase II. Model (T6) is solved for each infeasible triangle to check for overlap, where the first constraint corresponds to the inequality defining facet AB.

Model (T6, infeasible triangle):

$$\max_{P_k} \text{obj}$$
$$\sum_{k\in K} \omega_k^{AB-X} P_k \leq \Phi^{AB-X} - \text{obj}$$
$$\sum_{k\in K} \omega_k^l P_k \leq \Phi^l - \text{obj} \quad \forall l \in L^{\text{CAP}}$$

If obj $< \delta_4$ ($\delta_4 > 0$) then there is no appreciable overlap. For the first round of merging, remaining infeasible triangles are grouped by external point and merged into polytopes. For the second round of merging, output polytopes from the first round are collected and merged again, this time as a single group. The first round of merging is organized by external point based on the premise that infeasible triangles that share an external point are more likely to be able to be merged.

Phase IV

Although infeasible polytopes as output by Phase III could be directly used by model (NCP1), some polytopes and constraints may not be useful. Phase IV attempts to delete them. It consists of five steps.

(1) Model (T7) is solved for each infeasible polytope ip' to check for overlap with the CAP.

Model (T7, ip'):

$$\max_{P_k} \text{obj}$$
$$\sum_{k\in K} \omega_k^l P_k \leq \Phi^l - \text{obj} \quad \forall l \in L^{ip'}$$
$$\sum_{k\in K} \omega_k^l P_k \leq \Phi^l - \text{obj} \quad \forall l \in L^{\text{CAP}}$$

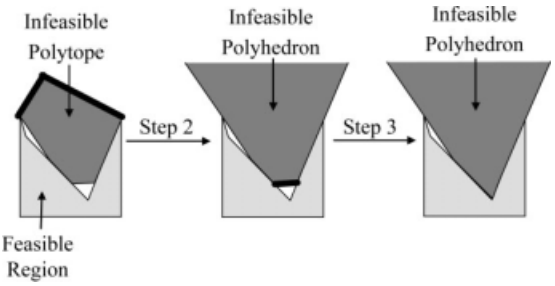

Figure 5. Removing constraints (dark lines) may expand that constraint's polytope/polyhedron.

If obj $<$ tolerance δ_5, then there is no appreciable overlap so ip' can be deleted.

(2) Certain constraints may have been useful up until now in they helped define closed infeasible regions. However, they are useless from here forward if they do not help indicate overlap with the CAP. The purpose of Step 2 is to delete such constraints. As the removal of constraints from an infeasible polytope may increase that polytope's overlap with the CAP, we solve model (T8) for each constraint $l' \in L^{ip'}$ to check whether removing l' increases overlap.

Model (T8, ip', $l' \in L^{ip'}$):

$$\max_{P_k} \text{obj}$$
$$\sum_{k\in K} \omega_k^l P_k \le \Phi^l - \delta_5 \quad \forall l \in L^{ip'} \backslash \{l'\}$$
$$\sum_{k\in K} \omega_k^{l'} P_k \ge \Phi^{l'} + \text{obj} \quad \forall l = l'$$
$$\sum_{k\in K} \omega_k^l P_k \le \Phi^l \quad \forall l \in L^{\text{CAP}}$$

The first two constraints establish the marginal area added by removing l'; the third constraint enforces overlap with CAP. If obj $\ge \delta_5$, then dropping constraint l' would enlarge polyhedron ip's overlap with CAP; therefore l' should be kept. If obj $< \delta_5$, then l' should be removed because it is redundant. In Figure 5, two of an infeasible polytope's five constraints are removed by Step 2, leaving it unbounded.

(3) Constraints not removed by Step 2 will, if deleted, increase overlap with the CAP. The purpose of Step 3 is to delete such constraints anyways if the additional overlap does not contain any feasible point. Each constraint l' is checked using model (T9).

Model (T9, ip', $l' \in L^{ip'}$):

$$\max_{P_k} \text{obj}$$
$$\sum_{k\in K} \omega_k^l P_k \le \Phi^l - \delta_5 \quad \forall l \in L^{ip'} \backslash \{l'\}$$
$$\sum_{k\in K} \omega_k^{l'} P_k \ge \Phi^{l'} + \text{obj} \quad \forall l = l'$$
$$(P_1, P_2) \in F^{\text{SM}}$$

If obj $\ge$ tolerance δ_5, then additional overlap does include feasible points, so l' should be kept. If obj $< \delta_5$, then l' should be removed. In Figure 5, one of the polyhedron's three remaining constraints is removed by Step 3.

(4) If $\omega^{l'} = -\omega^{l''}$ and $\Phi^{l'} = -\Phi^{l''}$, then constraints l' and l'' are both added to set L^{flush}. If model (T10), which defines segment $\omega^{l'.T} P = \Phi^{l'}$ is infeasible, then it may be desirable to inflate l' by small amount δ_6 such that overlap of l' and l'' is modeled as infeasible (see remarks for a more in-depth discussion).

Model (T10, $l' \in L^{\text{flush}}$):

$$\sum_{k\in K} \omega_k^l P_k \le \Phi^l - \delta_5 \quad \forall l \in L^{ip'} \backslash l'$$
$$\sum_{k\in K} \omega_k^{l'} P_k = \Phi^{l'}$$
$$\sum_{k\in K} \omega_k^l P_k \le \Phi^l \quad \forall l \in L^{\text{CAP}}$$

(5) Model (T11) is solved for each constraint l' to find an appropriate value to use for M^l in model (NCP1). Parameter M^l is used to relax constraint l when $Z^l = 0$. The optimal objective of model (T11) provides the smallest allowable value of M^l such that every point in the CAP will satisfy constraint l' when $Z^{l'} = 0$.

Model (T11, l'):

$$\max_{P_k} \text{obj}$$
$$\sum_{k\in K} \omega_k^{l'} P_k = \Phi^{l'} - \text{obj}$$
$$(P_1, P_2) \in F^{\text{CAP}}$$

Models (T6) and (T7) are similar in that they are used to check for overlap, before and after merging, respectively. Preprocessing via model (T6) is optional; however, it lessens the burden on the Phase III merging. Step 3 of Phase VI can locate corner points that converged toward but not actually found by Phase II. Models (T8) and (T9) are similar in that they are used to check for whether a constraint can be removed (is redundant with CAP) or should be removed (hides a corner), respectively.

Phase V

Model (NCP1) expresses the nonconvex region as a feasible polytope with infeasible polyhedrons removed. Phase V converts this representation into the union of multiple feasible polytopes. For model (NCP1), a point P is feasible if it satisfies every CAP constraint, $l \in L^{\text{CAP}}$, and violates at least one constraint from each infeasible polyhedron ip, $l \in L^{ip}$. In other words, every feasible point must satisfy all CAP constraints and at last one "flipped" constraint $\sum_{k\in K} \omega_k^l P_k \ge \Phi^l$, $l \in L^{ip}$ for each infeasible polyhedron ip. Thus, choosing exactly one constraint from each infeasible polyhedron leads to a "combination" of linear constraints whose intersection with CAP can potentially result in a polytope that covers part of the nonconvex region. For each combination $c' \in C$, model (T12) is solved to check whether that combination's linear constraints $l \in L^c$ define a feasible polytope.

Model (T12, c'):

$$\sum_{k\in K} \omega_k^l P_k \le \Phi^l \quad \forall l \in L^{\text{CAP}}$$
$$\sum_{k\in K} \omega_k^l P_k \ge \Phi^l \quad \forall l \in L^{c'}$$

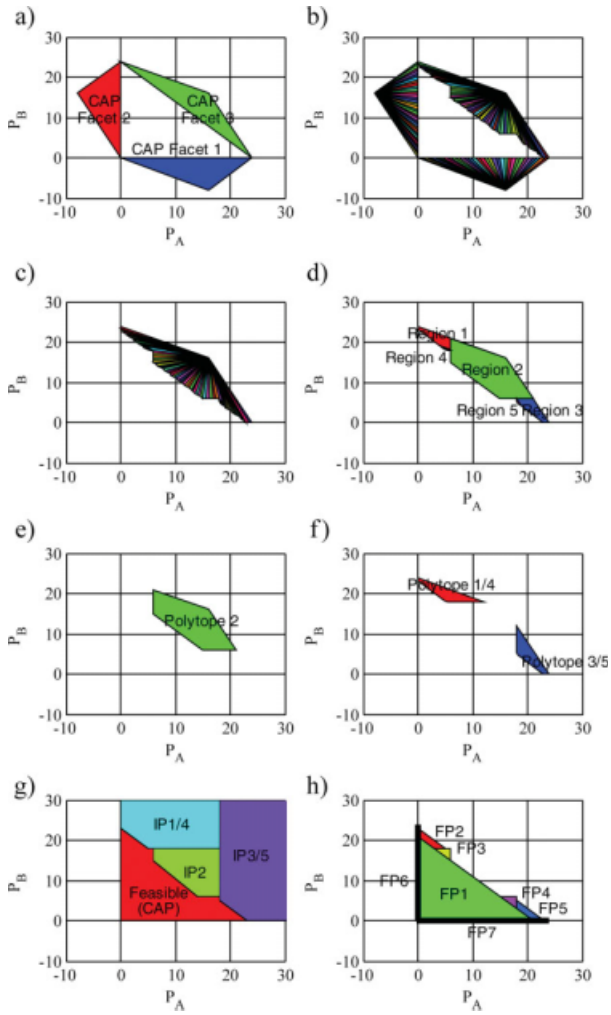

Figure 6. Motivating example: (a) Phase I, (b) Phase II, (c) preprocessing in Phase III, (d) first round of merging in Phase III, (e/f) second round of merging in Phase III, (g) Phase IV, (h) Phase V.

[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Combinations for which model (T12) is infeasible are deleted. Remaining combinations are merged into fewer feasible polytopes, which are then expressed in V-representation for use by model (NCP2).

Remarks

Motivating example revisited

The motivating example's feasible region and CAP are shown in Figures 1d,e, respectively. In Phase I, the CAP is used to create initial infeasible triangles (Figure 6a). In Phase II, many new infeasible triangles are created and expanded to push against the boundary of the feasible region (Figure 6b). In Phase III preprocessing, infeasible triangles that do not overlap the CAP are removed, leaving only infeasible triangles derived from CAP Facet 3 (Figure 6c). In the first round of merging infeasible triangles with external point (16,16) are merged into Regions 1, 2, and 3 (note that these regions overlap each other), while those with external point (18.479, 5.021) or (5.021, 18.479) are merged into Regions 6 and 7, respectively (Figure 6d). A second round of merging combines Regions 1–5 into three polytopes: 1/4, 2, 3/5 (Figures 6e,f). Phase IV identifies missing corner points for all three polytopes as well as converts them into polyhedrons. These polyhedrons are shown in Figure 6g as removing area from the CAP. As model (NCP1) contains three sets of three constraints, there are 27 enumerated combinations, of which 12 are full-dimensional regions that can be merged into feasible polytopes FP1-5 (Figure 6h) using the merging routine. The remaining 15 combinations are feasible lines on the *y*-axis (six combinations), feasible lines on the *x*-axis (six combinations), or points at the origin (three combinations). These combinations are manually merged into FP6 (vertical line in Figure 6h) and FP7 (horizontal line in Figure 6h).

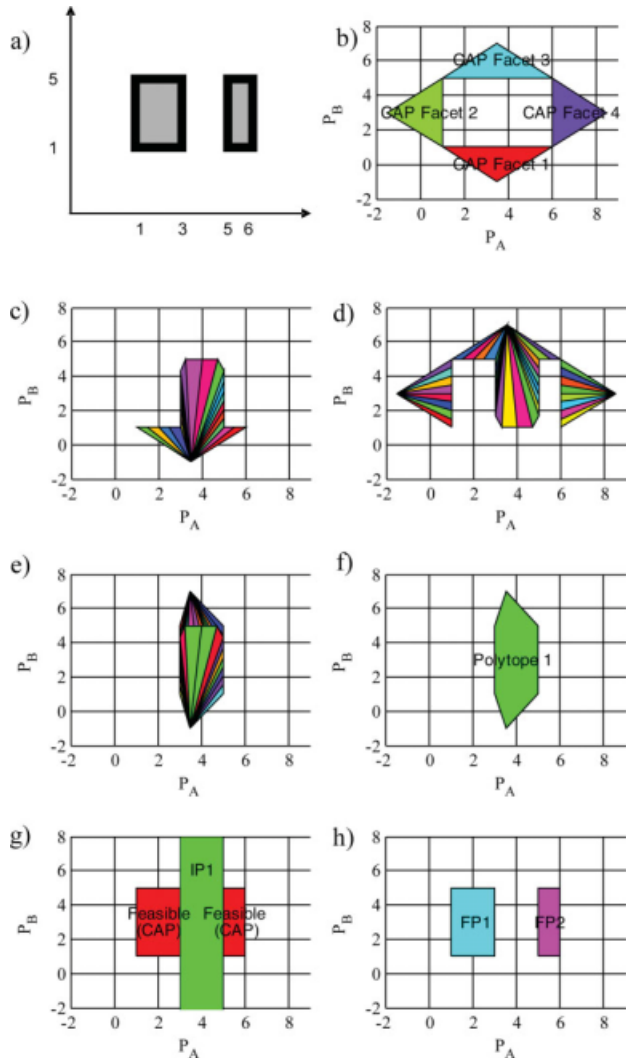

Figure 7. Second example: (a) feasible region, (b) Phase I, (c/d) Phase II, (e) preprocessing in Phase III, (f) two rounds of merging in Phase III, (g) Phase IV, (h) Phase V.

[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Disjoint feasible region

A second example is presented to illustrate the ability of the algorithm to handle disjoint feasible regions such as the one in Figure 7a. The CAP of the region is a rectangle with four facets. Phase I in Figure 7b assigns each CAP facet an external point so that each forms an infeasible triangle. Phase II in Figure 7c (CAP Facet 1 only) and Figure 7d (CAP Facets 2–4) increases the number and coverage of infeasible triangles. Note that some infeasible triangles derived from CAP Facet 1 overlap some infeasible triangles derived from CAP Facet 3 (see Figure 7e). Preprocessing removes infeasible triangles that do not overlap the CAP. Infeasible triangles that overlap each other will always be a subset of infeasible triangles that overlap the CAP; in this example, the two sets are the same. Two rounds of merging by Phase III combine remaining infeasible triangles into Polytope 1 in Figure 7f. Phase IV removes all constraints from Polytope 1 except $3 < x_1$ and $x_1 < 5$ (see Figure 7g). Model (NCP1) requires for feasibility that one of these constraints be violated, i.e., $x_1 \leq 3$ or else $5 \leq x_1$. Phase V converts this pair of constraints along with the CAP into two feasible polytopes (rectangles in Figure 7h).

Higher dimensions

The proposed algorithm can be used to identify and express nonconvex regions of higher dimensions. Although the required computational effort increases in the number of products, this is not a limitation of our method because the NCP of a process network needs to be generated only once and furthermore this can be carried out offline. In addition to this, our algorithm is more useful in higher dimensions or the presence of higher complexity because it uses scheduling formulations, in models (T1), (T2), (T5), (T7), and (T8), for a single planning period, whereas full-space methods contain multiple scheduling ''copies,'' one for each period.

Another advantage of this method is that the growth in the number of binary variables necessary to represent NCP is slower than the increase in the number of binary variables of the corresponding scheduling formulations. To illustrate this point, we present a third example which is modified from Papageorgiou and Pantelides.[33] We consider the production over a horizon of 12 h, using discrete-time resource-task network (RTN) formulation[47] (see Appendices B and C) and the proposed method. The detailed scheduling model contains 783 equations, 523 variables, and 130 binary variables. The CAP, which is an LP model, contains five equations, three variables (P_A, P_B, P_C), whereas NCP1 is modeled as the CAP with three infeasible polyhedrons, each with three constraints, removed. It contains 17 constraints, 12 variables, and 9 binary variables.

Classification of feasibility

In rare instances, merging of infeasible triangles in Phase III may wrongly remove or create lower-dimensional feasible regions. The root cause of this problem is that input infeasible triangles contain an infeasible interior, two infeasible sides *AX* and *BX*, and an *ambiguous* side *AB*. Hence, before merging some sides contain feasible points. During merging (Phase III), everything is assumed to be infeasible and redistributed. After merging (Phase IV), model (NCP1) treats all interiors as infeasible and all sides as feasible. A possible change to the algorithm would be to, right before merging, remove from each infeasible triangle's facet (segment *AB*) a δ_6-wide strip, such that all interiors and sides are infeasible before, during, and after merging.

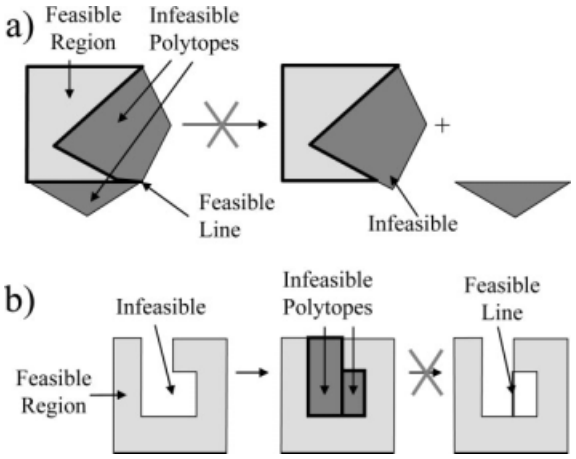

Figure 8. Merging may redistribute infeasible area such that: (a) a feasible line is wrongly made infeasible, (b) an infeasible line is wrongly made feasible.

Currently, the merging routine may misclassify a feasible line as infeasible if that line is surrounded on all sides by different infeasible triangles' ambiguously feasible segment *AB*. In some cases, such as the one shown Figure 8a, preprocessing before merging in Phase III is enough to sidestep this problem. It is also a possible issue that the merging routing may wrongly misclassify an infeasible line as feasible, if such a line is the shared boundary between multiple polytopes generated during merging. In some cases, such as the one shown Figure 8b, this can be remedied through inflation of neighboring constraints as described in Step 4 of Phase IV.

Implementation issues

A number of tolerances have been introduced to suppress possible numerical issues. The first tolerance δ_1 is actually a termination criterion. The algorithm continues until either all relevant facet sizes fall below δ_1 or else an iteration limit is hit. The second tolerance δ_2 in model (T1) is used to enforce a strict inequality. The third tolerance δ_3 in model (T2) ensures that point *D* is sufficiently far from segment *AB*. It is also a threshold for accepting new point *E* after solving model (T5). The fourth tolerance δ_4 in model (T6) ensures that only infeasible triangles that overlap the CAP are input into merging. The fifth tolerance δ_5 in model (T7) removes merging outputs that do not overlap the CAP enough. It is also used in models (T8) and (T9) to determine whether constraints should be kept. The sixth tolerance δ_6 is meant to convert specific inequalities, as identified by model (T10), into strict inequalities. The most important parameters in terms of solution quality appear to be δ_1 and iteration limit because they control Phase II where new information is

gathered by our algorithm. The other important parameter is δ_5; increasing this parameter decreases the number of infeasible polytopes passed to model (NCP1), in turn strongly decreasing the number of combinations enumerated in Phase V for model (NCP2).

Rolling Horizon Approach

In rolling horizon strategies for production planning, early periods of the planning horizon are successively fixed to feasible solutions. We propose an approach where the production planning problem in Eqs. 1–5 is solved with Eq. 2 in all time periods replaced by NCP1 or NCP2. This establishes production $P_{k,t}$, inventory $I_{k,t}$, and unmet demand $U_{k,t}$ for all time periods. Production amounts for the first period are labeled as production target P_k^*, and model (T13) is solved to find a feasible schedule P_k for $t = 1$. Scalar weight γ emphasizes that overproduction is preferable to underproduction based on the premise that excess inventory is preferable to unplanned failure to meet demand.

Model (T13):

$$\min_{P_k, P_k^+, P_k^-} \text{dev} = \sum_k \left(P_k^+ + \gamma P_k^-\right)$$
$$P_k = P_k^* + P_k^+ - P_k^- \quad \forall k \in K$$
$$P_k^+, P_k^- \geq 0 \quad \forall k \in K$$
$$P \in F^{\text{SM}}$$

Ideally, we obtain dev $< \delta_7$ meaning that model (T13) has found a feasible schedule P_k that is the same as or very similar to production target P_k^*. If so, the rolling horizon scheme skips to the next period. If optimal dev $\geq \delta_7$, then model (T13) has instead returned a feasible solution P_k that is not similar enough to P_k^*. Production amount in the first time period is replaced with P_k and fixed. The production planning problem is then re-solved to update production amounts for $t \in \{2, 4, \ldots T\}$.

Next, P_k^* is reset as the second time period's production target, and model (T13) is solved to find a feasible schedule P_k for $t = 2$. This continues for $t \in \{3, 4, \ldots, T\}$ until $P_{k,t}$ is fixed to feasible schedules for every period. Optionally, Sung and Maravelias[4] proposed saving feasible solutions found during CAP (and NCP) investigation as well as during rolling horizon. Model (T13) may start off by checking against these saved solutions.

Integrated Production Planning and Scheduling

Two products, A and B, are to be produced and delivered over 40 one-day time periods. Each one-day time period consists of 20 available hours plus 4 downtime hours during which unit setups and inventory of intermediates are reset to their original state. Demand for each product at the end of each time period is presented in Appendix D. Inventory is penalized at h_k = \$20/kg-time period. Unmet demand is penalized at u_k = \$400/kg-time period. Raw materials RM1, RM2, and RM3 are converted into Products A and B via tasks H, R1, R2, R3, and S which occur on units U1, U2, U3, and U4. Relationships between tasks, states, and units are shown below (modified from Kondili et al.[5] see Appendix E):

(1) Tasks:

- Task H: Process RM1 for 1 h to form HOT.
- Task R1: Mix 50% RM2 and 50% RM3 for 2 h to form INT1.
- Task R2: Mix 40% HOT and 60% INT1 for 2 h to form 40% Product A and 60% INT2.
- Task R3: Mix 20% RM3 and 80% INT2 for 1 h to form IMP.
- Task S: Process IMP for 2 h to form 90% Product B and 10% INT2.

(2) Units:

- Unit U1: Capacity 50 kg, suitable for Task H.
- Unit U2: Capacity 40 kg, suitable for Tasks R1, R2, and R3. However, 1 h setup is needed for each.
- Unit U3: Capacity 25 kg, suitable for Tasks R1, R2, and R3. However, 1 h setup is needed for each.
- Unit U4: Capacity 100 kg, suitable for Task S.

The detailed scheduling formulation that we use here is the discrete-time RTN formulation.[47] However, our approach can be used with other scheduling formulations (even non-MILP formulations) instead. Each scheduling time period $p \in \{1, 2, \ldots, 20\}$ is 1-h long. Figure 9 shows a detailed schedule as would be obtained by model (RTN) in Appendix B.

Implementation

The CAP of scheduling model (RTN) can be expressed in the form $\omega_A{}^l P_A + \omega_B{}^l P_B \leq \Phi^l$ as shown in Table F1 in Appendix F. This can be found by applying the surrogate generation algorithm of Sung and Maravelias[4] to model (RTN). The first nonconvex projection, as formulated using model (NCP1), expresses solutions as feasible if they are within the CAP but not within eight infeasible polyhedra. The constraints defining these polyhedra are listed in Table F2. The representation using model (NCP2) uses the union of nine feasible polytopes. The coordinates P_k^v for the vertices of these polytopes are shown in Table F3. For comparison purposes, we will also be using the linear relaxation (LR) of

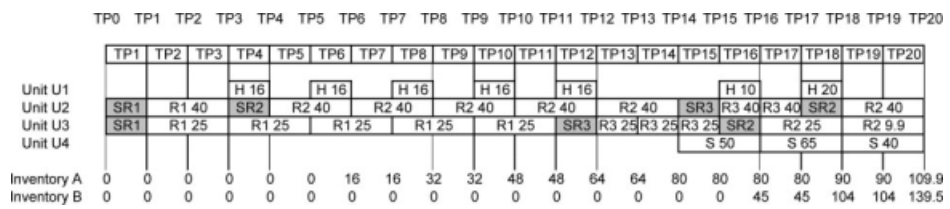

Figure 9. A detailed schedule that produces 109.5 kg Product *A* and 139.5 kg Product *B*.

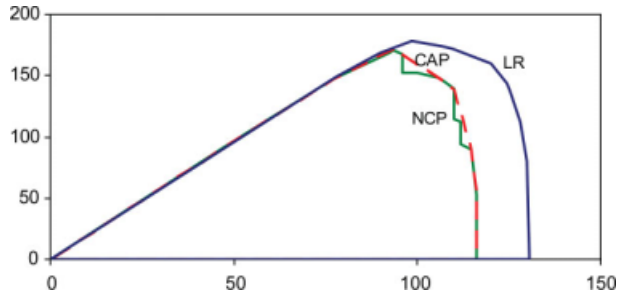

Figure 10. Feasible regions for one time period according to linear relaxation LR, CAP, and NCP.

[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

model (RTN). The area covered by LR may be identified and expressed by again using the convex surrogate generation algorithm of Sung and Maravelias[4] (Table F4).

Figure 10 compares the different feasible regions defined by linear relaxation LR, CAP, and NCP (models (NCP1) and (NCP2) are based on the same region). The feasible region of LR is the largest meaning that, compared to CAP and NCP, it provides the most optimistic assessment of what (P_A, P_B) combinations are feasible. The area covered by NCP is slightly smaller than CAP due to its ability to identify and formulate nonconvex small indentations in the CAP. Although this may appear graphically insignificant, it will be demonstrated later that this significantly improves the quality of bounds and feasible solutions found.

In the production planning problem, we will be using Tables F1–F3 to replace Eq. 2. Also, we will be directly using the linear relaxation of model (RTN), ignoring Table F4, because the LP relaxation already results in a linear program.

Solution

The production planning problem in Eqs. 1–5 was solved with Eq. 2 replaced by model (NCP2). The new problem was solved in 1.10 CPU s with objective value \$112,860. As model (NCP2) is an over-approximation, this objective value may be used as a bound on the optimal objective value (lower bound since total cost is being minimized). Figure 11 shows production amounts predicted using model (NCP2). Note that most though not all production amounts are predicted to be on the boundary of the (NCP2) feasible region.

Using the described rolling horizon scheme, model (T13) was solved to optimality 40 times in 2,411 CPU s. For 36 of 40 time periods, an exact match (dev $< \delta_7$) was found. For the remaining four time periods, inexact matches were found with average dev of 0.262. Total time to re-solve the production planning problem was 1.87 CPU s. Completion of the rolling horizon scheme yielded a feasible solution of \$113,155.

The final production planning solution is shown in Figures 12a and 13a. Demand fulfillment is shown in Figures 12b and 13b. Production of *A* operates near full capacity in every period due to demand spikes. Production and inventory of *B* are able to fulfill all demand, though very small amounts are unmet in several periods. Overall, 143.697 kg *A* and 0.360 kg *B* are unmet at \$57,623. Also, 2,372.8 kg-time period of *A* and 403.8 kg-time period of *B* are held in inventory at \$55,533 (units of kg-time period are reported here since some amount is held over multiple time periods).

Model (NCP2) in combination with the rolling horizon scheme was able to bound the optimal objective value of the combined production planning and RTN problem between \$112,860 and \$113,155, leaving a gap of only \$295 (0.26%). Just as important, production amounts for every time period are backed by RTN-formulated schedules. For example, 109.9 kg of Product *A* and 139.5 kg of Product *B* were chosen to be produced in time periods $t \in \{3,5,10,15,25,30–34\}$. A detailed schedule for producing this combination of production amounts was shown in Figure 9.

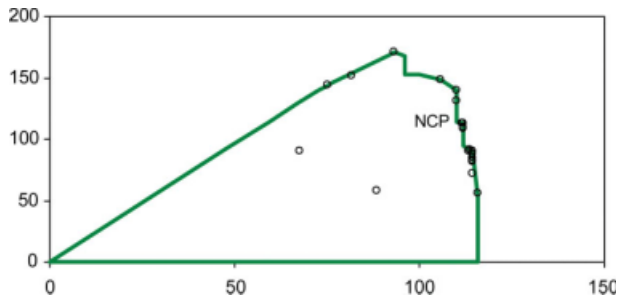

Figure 11. Predicted production amounts (i.e., production targets) using model (NCP2).

Forty production targets are shown (there is some overlap), one production target per time period. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

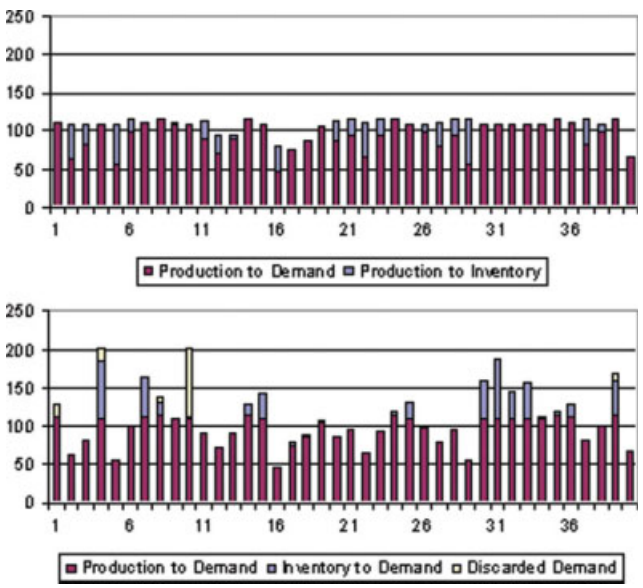

Figure 12. Production of *A* (top) and demand for *A* (bottom) in NCP2 solution.

[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

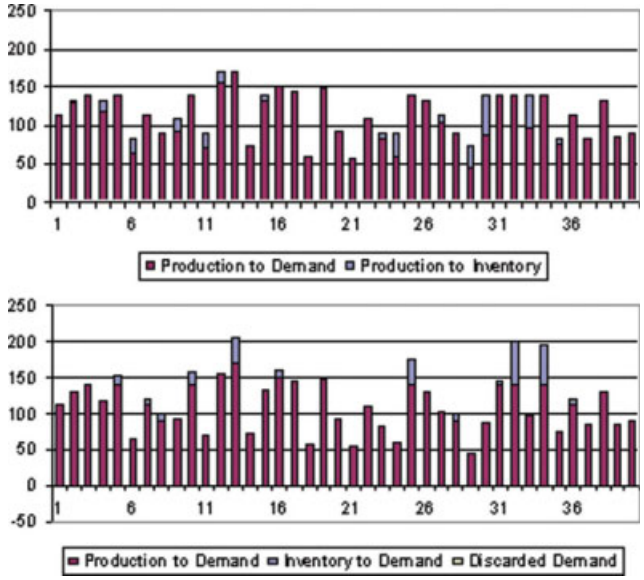

Figure 13. Production of *B* (top) and demand for *B* (bottom) in NCP2 solution.

[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Discussion

We solved the same instance using four other methods: (a) rolling horizon using linear relaxation (LR) of detailed scheduling model (RTN), (b) rolling horizon using the convex approximation CAP, (c) rolling horizon using (NCP1), and (d) full space (solved only once) using model (RTN). Solving the production planning problem using (a), (b), and (c) yielded minimum total cost: \$28,511 by linear relaxation (4.53 CPU s), \$109,409 by CAP (0.062 CPU s), and \$112,874 by NCP1 (2.86 CPU s). Figure 14 shows predicted production amounts. As in Figure 11, many production targets were set near the boundary of each surrogate model's feasible region. Note that models (NCP1) and (NCP2) predicted nearly identical production targets since both are based on the same feasible region.

The rolling horizon scheme was used to find the following feasible solutions: \$182,841 by linear relaxation, \$115,181 by CAP, and \$113,325 by NCP2. The objective function "degraded" during the rolling horizon scheme whenever model (T13) was unable to exactly match a time period's production target P_k^* with a feasible schedule P_k. Models (NCP1) and (NCP2) were able to find exact matches in 78% and 90% of time periods, respectively, so degradation was minimal. The two models produced slightly different targets (Figure 14c vs. Figure 11) that led to slightly different feasible solutions; this may have been due to numerical rounding when converting from one representation to the other and the existence of production planning solutions with the same objective function value but different production targets that are likely to lead to discrepancies when the targets cannot be met exactly. The targets predicted using CAP were matched exactly in 43% of time periods due to it choosing many production amounts within indentations of the NCP feasible region (see Figure 14b). However, feasible production amounts were always located nearby so degradation was limited though apparent. Targets based on LR were matched in only 25% of time periods. Furthermore, LR production targets were revised by model (T13) significantly downwards in 63% of time periods leading to insufficient buildup of inventory, unplanned failure to meet demand, and sizeable degradation of the objective function.

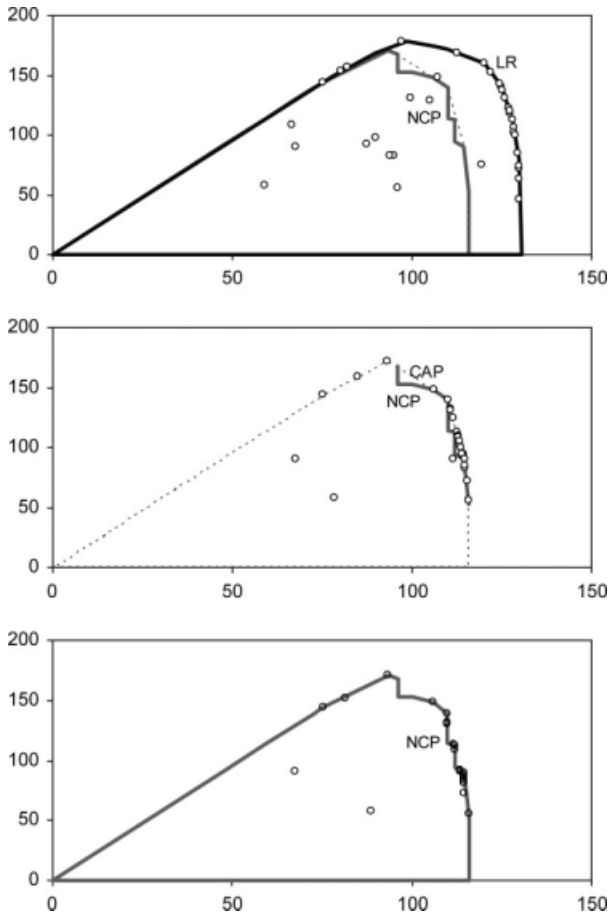

Figure 14. Predicted production amounts, before rolling horizon scheme: using linear relaxation (top), CAP (middle), and NCP1 (bottom).

Forty production targets are shown for each. For reference, the NCP feasible region is shown in (a) and (b) as well.

For reference, the combined production planning and scheduling model was solved for 10,000 CPU s as a single model ("full space") using CPLEX 10.0. A bound of \$28,511 (in this case the same as the bound found using linear relaxation) was found after 9.17 CPU s. This was eventually improved up to \$30,799. A feasible solution with objective value \$234,808 was found at around 900 CPU s; it

Table 1. Comparison of Bounds and Feasible Solutions

		Total Time	Bound	Feasible Solution	Gap
Full space	RTN	881.89	30,722	234,808	204,086
		10,000.00	30,799	182,899	152,101
Rolling horizon	LR	450.34	28,511	182,841	154,330
	CAP	2,170.47	109,409	115,181	5,771
	NCP1	2,415.25	112,874	113,325	451
	NCP2	2,414.37	112,860	113,155	295

Table 2. Model Sizes

Production Planning +	Constraints	Variables	Binary Variables
–	81	241	0
RTN	65,681	49,801	16,800
LR	65,681	49,801	0
CAP	641	241	0
NCP1	1,681	961	720
NCP2	521	2,161	360

remained unimproved until near the last 1000 CPU s when CPLEX "polishing" identified several better solutions, the best having objective value \$182,899, which is more than 60% suboptimal. This small example illustrates clearly that full-space methods, although theoretically more accurate, cannot be used to address production planning models: they not only provide solutions of low quality, but also provide very poor bounds.

Table 1 shows the bound and feasible solution found by each method. For the rolling horizon scheme, regardless of which surrogate model was used, almost all time went to solving model (T13) 40 times. Re-solving the production planning problem took negligible time, which illustrates the effectiveness of formulations NCP1 and NCP2 as surrogate models. They lead to very accurate production planning formulations while being extremely accurate. Model sizes are shown in Table 2.

All models were implemented in GAMS 22.1 and solved using CPLEX 10.0 with default options on a Pentium 4 at 2.8 GHz running Windows XP. The following computation times are offline: CAP was generated (Phase I) in 90 CPU s. We wanted to emphasize the potential accuracy of this method, so NCP1 was generated (Phase II–IV) in 24,000 CPU s, almost all spent in Phase II. NCP2 was generated (Phase V) in an additional 65 CPU s. For comparison (shown here only), we also followed through the algorithm using fewer Phase II iterations: NCP1 was generated in 5000 CPU s and NCP2 was generated in an additional 15 CPU s. This resulted in a bound and feasible solution of \$112,709 and \$114,898, respectively, which is consistent with the more accurate values of \$112,860 and \$113,155.

Conclusions

A framework was presented for the solution of hard production planning and scheduling problems. An accurate yet compact surrogate model was developed for the representation of the projection of the scheduling feasible region onto the space of production amounts. The framework can represent nonconvex regions, thus allowing to be addressed all classes of process networks, including those with large changeovers. We proposed two MIP formulations: (a) one that conveys difference, a feasible polytope with infeasible polyhedrons removed, and (b) one posed as the union of feasible polytopes. An algorithm was also presented for the investigation and the generation of polytopes and polyhedrons for these two formulations. The proposed method was able to provide tighter bounds than previously proposed methods. Just as important, it yields more accurate predictions that can be combined with schemes such as rolling horizon to identify feasible solutions of extremely high quality in reasonable computational time. For the presented integrated example, the production planning formulations based on models (NCP1) and (NCP2) led to gaps of 0.4% and 0.3%, respectively. Finally, it is important to note that the proposed framework may be applied to any model or application in which there is a hierarchical relationship—not just production planning and scheduling. The ability to generate, on demand, accurate yet compact surrogate models allows a generic higher-level model to respect—with minimal increase in model size—additional information as could be offered by a lower-level model.

Acknowledgments

The authors would like to acknowledge financial support from the National Science Foundation under Grant CTS-0547443.

Notation

Production Planning

Indices/sets

k = products $k \in K$
t = production planning time periods t

Parameters

$d_{k,t}$ = demand for k due at the end of period t
h_k = holding cost for product k for one period
u_k = penalty for not meeting demand for product k

Variables

Cp_t = production cost in period t
CT = total cost (objective function)
$f(P_{k,t})$ = generic function for production amount feasibility
$g(P_{k,t})$ = generic function for production cost
$I_{k,t}$ = inventory of product k at the end of period t
$P_{k,t}$ = production amount of product k in period t
$U_{k,t}$ = amount of unmet demand for product k in period t

Algorithm

Indices/sets

c, c' = combination of constraints: one constraint from each polyhedron ip
fp = feasible polytopes, $fp \in FP$
ip, ip' = infeasible polytopes/polyhedrons, $ip \in IP$
l, l' = constraints ($\omega_i^l P_1 + \omega_2^l P_2 + \ldots \leq \Phi^l$), $l \in L$
v = vertices, $v \in V$

Subsets

L^c = constraints for combination c
L^{CAP} = constraints for polytope CAP
L^{fp} = constraints for polytope fp
L^{ip} = constraints for polyhedron ip
V^{fp} = set of vertices for feasible polytope fp

Parameters

M^l = big-M relaxation for constraint l
n = dimensions, $|K|$
P_k^v = product k coordinate for vertex v
P_k^* = target production amount
γ = increased penalty for producing below, rather than above, production target, 5
δ_1 = minimum facet size, 1
δ_2 = tolerance for checking whether a facet is infeasible, 0.1
δ_3 = tolerance for accepting a boundary feature, 0.005
δ_4 = tolerance for accepting infeasible triangle overlap with CAP, 0.005
δ_5 = tolerance for accepting infeasible polytope overlap with CAP, 0.005 or 0.2

δ_6 = positive number used to enforce strict inequalities, 0.001
δ_7 = tolerance for accepting P_k as exactly matching P_k*, 0.001
Φ^l = right hand side of constraint l
ω^l = coefficient vector of constraint l

Variables

dev = weighted mismatch from P_k^* (objective function)
obj = generic objective function
P_k^+ = amount above P_k^*
P_k^- = amount below P_k^*
Z^{fp} = binary = 1 if feasible polytope fp is satisfied
Z^l = binary = 1 if constraint l is not satisfied
λ^v = weight on vertex v, between 0 and 1. Specific vertices have specially labeled weights: λ^A, λ^B, λ^C, λ^D, λ^E, λ^X

Literature Cited

1. Bassett MH, Pekny JF, Reklaitis GV. Perspectives on model-based integration of process operation. *Comput Chem Eng*. 1996;42:821–844.
2. Grossmann IE, Van den Heever SA, Harjunkoski I. Discrete optimization methods and their role in the integration of planning and scheduling. *AIChE Symp Ser*. 2002;98:150–168.
3. Maravelias CT, Sung C. Integration of production planning and scheduling: Overview, Challenges and Opportunities. Computers and Chemical Engineering, in press, available on-line (DOI: 10.1016/j.compchemeng.2009.06.007).
4. Sung C, Maravelias CT. An attainable region approach for production planning of multiproduct processes. *AIChE J*. 2007;53:1298–1315.
5. Kondili E, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch operations. I. MILP formulation. *Comput Chem Eng*. 1993;17:211–227.
6. Schilling G, Pantelides CC. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Comput Chem Eng*. 1996;20:S1221–S1226.
7. Zhang X, Sargent RWH. The optimal operation of mixed production facilities—General formulation and some approaches for the solution. *Comput Chem Eng*. 1996;20:897–904.
8. Ierapetritou MG, Floudas CA. Effective continuous-time formulation for short-term scheduling. I. Multipurpose batch processes. *Ind Eng Chem Res*. 1998;37:4341–4359.
9. Castro P, Barbosa-Povoa APFD, Matos H. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Ind Eng Chem Res*. 2001;40:2059–2068.
10. Maravelias CT, Grossmann IE. New continuous-time state task network formulation for the scheduling of multipurpose batch plants. *Ind Eng Chem Res*. 2003;42:3056–3074.
11. Maravelias CT. Mixed-time representation for state-task network models. *Ind Eng Chem Res*. 2005;44:9129–9145.
12. Pinto JM, Grossmann IE. A continuous time mixed integer linear programming model for short term scheduling of multi-stage batch plants. *Ind Eng Chem Res*. 1995;34:3037–3051.
13. Castro PM, Grossmann IE, Novais AQ. Two new continuous-time models for the scheduling of multi-stage batch plants with sequence dependent changeovers. *Ind Eng Chem Res*. 2006;45:6210–6226.
14. Mendez CA, Henning GP, Cerda J. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. *Comput Chem Eng*. 2001;25:701–711.
15. Cerda J, Henning GP, Grossmann IE. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel pines. *Ind Eng Chem Res*. 1997;36:1695–1707.
16. Gupta S, Karimi IA. An improved MILP formulation for scheduling multi-product, multi-stage batch plants. *Ind Eng Chem Res*. 2003;42:2365–2380.
17. Prasad P, Maravelias CT. Batch selection, assignment and sequencing in multi-stage multi-product processes. *Comput Chem Eng*. 2008;32:1106–1119.
18. Sundaramoorthy A, Maravelias CT. Modeling of storage in batching and scheduling of multistage processes. *Ind Eng Chem Res*. 2008;47:6648–6660.
19. Hooker JN, Ottosson G, Thorsteinsson ES, Kim HJ. On integrating constraint propagation and linear programming for combinatorial optimization. *In: Proceedings of 16th National Conference on Artificial Intelligence (AAAI-99)*. Menlo Park, CA: The AAAI Press/MIT Press, 1999:136–141.
20. Jain V, Grossmann IE. Algorithms for hybrid MIP/CP model for a class of optimization problems. *INFORMS J Comput*. 2001;13:258–276.
21. Maravelias CT. A decomposition framework for the scheduling of single- and multi-stage processes. *Comput Chem Eng*. 2006;30:407–420.
22. Wilkinson SJ, Shah N, Pantelides CC. Aggregate modelling of multipurpose plant operation. *Comput Chem Eng*. 1995;19:S583–S588.
23. Vancza J, Kis T, Kovacs A. Aggregation—the key to integrating production planning and scheduling. *CIRP Ann Manuf Technol*. 2004;53:377–380.
24. Birewar DB, Grossmann IE. Simultaneous production planning and scheduling in multiproduct batch plants. *Ind Eng Chem Res*. 1990;29:570–580.
25. Wellons MC, Reklaitis GV. Scheduling of multipurpose batch chemical plants, Part 2: Multi-product campaign formulation and production planning. *Ind Eng Chem Res*. 1991;30:688–705.
26. Wan X, Pekny JF, Reklaitis GV. Simulation-based optimization with surrogate models—application to supply chain management. *Comput Chem Eng*. 2005;29:1317–1328.
27. Li Z, Ierapetritou MG. Process scheduling under uncertainty using multiparametric programming. *AIChE J*. 2007;53:3183–3203.
28. McKay KN, Safayeni FR, Buzacott JA. A review of hierarchical production planning and its applicability for modern manufacturing. *Prod Planning Control*. 1995;6:384–394.
29. Subrahmanyam S, Pekny JF, Reklaitis GV. Decomposition approaches to batch plant design and planning. *Ind Eng Chem Res*. 1996;35:1866–1876.
30. Chu Y, Xia Q. Generating benders cuts for a general class of integer programming problems. *Lecture Notes Comput Sci*. 2004;3011:127–141.
31. Erdirik-Dogan M, Grossmann IE. A decomposition method for the simultaneous planning and scheduling of single-stage continuous multiproduct plants. *Ind Eng Chem Res*. 2006;45:299–315.
32. Ghosh S, Gaimon C. Production scheduling in a flexible manufacturing system with setups. *IIE Trans*. 1993;24:21–35.
33. Papageorgiou LG, Pantelides CC. Optimal campaign planning/scheduling of multipurpose batch/semicontinuous plants. I. Mathematical formulation. *Ind Chem Eng Res*. 1996;35:488–509.
34. Kallrath J. Planning and scheduling in the process industry. *OR Spectrum*. 2002;24:219–250.
35. Reklaitis GV. Overview of planning and scheduling technologies. *Latin Am Appl Res*. 2000;30:285–293.
36. Berning G, Brandenburg M, Gürsoy K, Kussi JS, Mehta V, Tölle F-J. An integrated system solution for supply chain optimization in the chemical process industry. *OR Spectrum*. 2002;24:371–401.
37. Lin X, Floudas CA, Modi S, Juhasz NM. Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant. *Ind Eng Chem Res*. 2002;41:3884.
38. Van den Heever SA, Grossmann IE. A strategy for the integration of production planning and reactive scheduling in the optimization of a hydrogen supply network. *Comput Chem Eng*. 2003;27:1813.
39. Barber CB, Dobkin DP, Huhdanpaa H. The Quickhull algorithm for convex hulls. *ACM Trans Math Software*. 1996;22:469–483.
40. Kvasnica M, Grieder P, Baoti M. *Multi-Parametric Toolbox*. Available at: http://control.ee.ethz.ch/~mpt/ (accessed on March 31, 2008).
41. Geyer T. Low Complexity Model Predictive Control in Power Electronics and Power Systems. Ph.D. Thesis. Switzerland: ETH Zurich, 2005:1–311.
42. Goyal V, Ierapetritou MG. Framework for evaluating the feasibility of nonconvex processes. *AIChE J*. 2003;49:1233–1240.
43. Ning P, Bloomenthal J. An evaluation of implicit surface tillers. *IEEE Comput Graph Appl*. 1993;13:33–41.
44. Van Overveld K, Wyvill B. Shrinkwrap: an efficient adaptive algorithm for triangulating an iso-surface. *Visual Comput*. 2004;20:362–379.

45. Balas E. Disjunctive programming and hierarchy of relaxations for discrete optimization problems. *SIAM J Algebr Discrete Methods*. 1985;35:466–486.
46. Türkay M, Grossmann IE. Disjunctive programming techniques for the optimization of process systems with discontinuous investment costs-multiple size regions. *Ind Eng Chem Res*. 1996;35: 2611–2623.
47. Pantelides CC. Unified frameworks for optimal process planning and scheduling. In: *Proceedings of Second Conference Foundations of Computer-Aided Process Operations (FOCAPO)*. Austin, TX: CACHE, 1994:253–274.
48. Davis BJ, Taylor LA, Manousiouthakis VI. Identification of the attainable region for batch reactor networks. *Ind Eng Chem Res*. 2008;47:3388–3400.
49. Glassier D, Hildebrandt D, Crowe C. A geometric approach to steady flow reactors: the attainable region and optimization in concentration space. *Ind. Eng. Chem. Res*. 1987;26:1803–1810.
50. Verderame PA, Floudas CA. Integrated operational planning and medium-term scheduling for large-scale industrial batch plants. *Ind Eng Chem Res*. 2008;47:4845–4860.

Appendix A: From Points to Facet-Defining Inequalities

Each side of an n-polytope is an $n-1$ dimensional "facet" defined by exactly n points. The linear inequality associated with a facet can be calculated by recognizing that $\boldsymbol{\omega}^T P = \Phi$ for each of its n points. This is shown below for $k \in \{1,2\}$ with the facet's $n = |K| = 2$ points labeled A and B with $P^A = (P_1^A, P_2^A)$ and $P^B = (P_1^B, P_2^B)$.

$$\boldsymbol{\omega}^T \mathbf{M} = [\,\omega_1 \quad \omega_2\,] \bullet \begin{bmatrix} P_1^A & P_1^B \\ P_2^A & P_2^B \end{bmatrix} = \Phi \bullet [\,1 \quad 1\,]$$

This can be solved as $\boldsymbol{\omega}^T\mathbf{M} = [1\ 1]\cdot\mathrm{adj}(\mathbf{M}) = [P_2^B - P_2^A, -P_1^B + P_1^A]$ and $\Phi = \det(\mathbf{M}) = P_1^A P_2^B - P_1^B P_2^A$. The signs of ω and Φ are then assigned based on some reference point known to be inside ($\boldsymbol{\omega}^T \mathrm{P} < \Phi$) or outside ($\boldsymbol{\omega}^T \mathrm{P} > \Phi$) the polytope (see Quickhull Algorithm[39]).

Appendix B: Scheduling (RTN) Formulation

A single production planning time period t is divided into equal-length scheduling time periods $p \in \{1, 2, \ldots, \mathrm{TP}\}$. RTN formulation requires tasks $i \in I$ for the conversion of resources $r \in R$, and a network for connecting tasks and resources. Equation B1 links production amounts $P_{k,t}$ of final product $k \in K \subset \mathrm{R}$ with the final inventory of the corresponding resource. Binary variable $N_{i,p}$ is active (=1) if task k starts at time point p. Task size $\xi_{i,p}$ is linked to $N_{i,p}$ by Eq. B2. Tasks may only start and end at time points, so resource levels are only updated and bounded at discrete time points, Eqs. B3 and B4, respectively.

$$P_k = R_{k,\mathrm{TP}} \quad \forall k \in K \tag{B1}$$

$$V_i^{\min} N_{i,p} \le \xi_{i,p} \le V_i^{\max} N_{i,p} \quad \forall i,p \tag{B2}$$

$$R_{r,p} = R_{r,p-1} + \sum_i \sum_{\theta=0}^{\tau_i} \left(\mu_{i,r,\theta} N_{i,p-\theta} + \upsilon_{i,r,\theta} \xi_{i,p-\theta}\right) \quad \forall r,p \tag{B3}$$

$$0 \le R_{r,p} \le R_r^{\max} \quad \forall r,p \tag{B4}$$

$$\xi_{i,p} \ge 0 \quad \forall i,p, \quad N_{i,p} \in \{0,1\} \ge 0 \quad \forall i,p \tag{B5}$$

Indices

i = tasks
p = scheduling time periods $p \in \{1, 2, \ldots, \mathrm{TP}\}$
θ = processing time index $\theta \in \{0, 1, \ldots, \max_i \tau_i\}$
r = resources: raw materials, intermediates, final products, utilities, processing units

Parameters

R_r^0 = initial amount of resource r
$R_r^{\max}$ = maximum amount of resource r
$V_i^{\min}$ = minimum batch size for task i
$V_i^{\max}$ = maximum batch size for task i
$\rho_{i,r}$ = if negative (positive), amount/proportion of input (output) for task i from resource r
$\mu_{i,r,\theta}$ = if negative (positive), amount of input (output) based on $N_{i,p}$
$\upsilon_{i,r,\theta}$ = if negative (positive), proportion of input (output) based on $\xi_{i,p}$
τ_i = processing time of task i, in units of period Δp

Variables

$N_{i,p}$ = binary = 1 if task i starting in period p
$R_{r,p}$ = amount of resource r at the end of period p
$\xi_{i,p}$ = batch size for task i starting in period p

Appendix C: Scheduling (RTN) Data for Third Example

This process network of this example, a variation of Papageorgiou and Pantelides,[33] is shown in Figure C1. The scheduling horizon is 12 h, divided in 1 h periods.

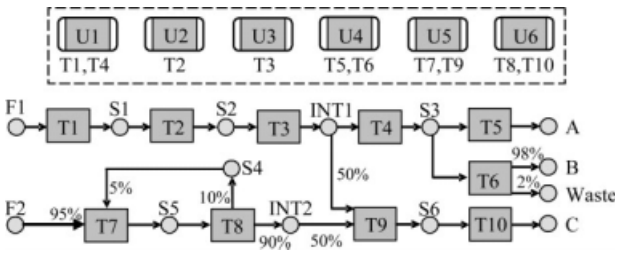

Figure C1. Process network for Third Example.

Table C1. Scheduling Parameters for Third Example: Discrete Change due to Task Binary $N_{i,p}$

		τ_i	U1	U2	U3	U4	U5	U6
	R_r^0		1	1	1	1	1	1
	$R_r^{\max}$		1	1	1	1	1	1
$\mu_{i,r,0}$ (<0),	T1	2	−1, 1					
$\mu_{i,r,\tau i}$ (>0)	T2	1		−1, 1				
	T3	1			−1, 1			
	T4	2	−1, 1					
	T5	2				−1, 1		
	T6	2				−1, 1		
	T7	4					−1, 1	
	T8	2						−1, 1
	T9	2					−1, 1	
	T10	3						−1, 1

Initial and maximum inventory are also shown in the table.

Table C2. Scheduling Parameters for Third Example: Proportional Change due to Task Size $\xi_{i,p}$

		τ_i	V_i^{max}/V_i^{min}	F1	F2	S1	S2	INT1	S3	S4	S5	INT2	S6	A	B	WS	C
kg	R_r^0			∞	∞	0	0	0	0	50	0	0	0	0	0	0	0
	R_r^{max}			∞	∞	∞	100	100	100	100	100	100	100	∞	∞	∞	∞
$v_{i,r,0}$ (<0), $v_{i,r,\tau i}$ (>0)	T1	1	50/0	−1		1											
	T2	2	80/0			−1	1										
	T3		60/0				−1	1									
	T4		50/0					−1	1								
	T5	2	80/0						−1					1			
	T6	2	80/0						−1						0.98	0.02	
	T7	2	30/0		−0.95					−0.05	1						
	T8	1	40/0							0.1	−1	0.9					
	T9	1	30/0					−0.5				−0.5	1				
	T10	2	40/0										−1				1

Initial and maximum inventory, minimum and maximum task size are also shown in the table.

Appendix D: Production Planning Data

Table D1. Production Planning Demand $d_{k,t}$ for Products A and B

t	$k = A$	$k = B$	t	$k = A$	$k = B$
1	130.130	112.859	21	96.195	55.954
2	63.496	129.085	22	66.708	109.096
3	82.048	141.326	23	93.338	82.943
4	200.798	117.877	24	120.655	60.372
5	55.726	152.748	25	130.931	174.097
6	99.292	63.856	26	98.249	130.995
7	165.067	120.604	27	80.093	102.601
8	139.117	99.651	28	95.987	99.959
9	110.274	91.699	29	56.433	45.957
10	202.107	157.270	30	160.331	87.741
11	89.790	70.507	31	188.294	144.705
12	70.490	156.116	32	146.233	200.096
13	89.852	206.443	33	156.481	97.525
14	129.452	72.450	34	112.655	193.912
15	144.578	132.384	35	118.509	75.263
16	46.638	159.066	36	128.213	120.492
17	79.383	144.145	37	82.466	83.324
18	88.693	57.828	38	99.265	130.941
19	107.246	148.151	39	167.540	85.306
20	87.241	92.157	40	67.539	90.798

Appendix E: Scheduling (RTN) Data for Integrated Example

The production recipes described earlier, based on Kondili et al.,[5] may be modeled as the process network shown in Figure E1. The scheduling horizon is 20 h, divided in 1 h periods.

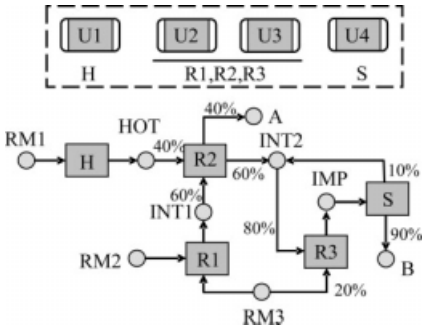

Figure E1. Process network for Integrated Example.

Note that Figure E1 only shows four units (U1, U2, U3, U4) and five tasks (H, R1, R2, R3, S). To model setup, unit U2 is split into four "units": U2_0, U2_R1, U2_R2, and U2_R3. When switching from one task to another, an immediate setup returns the unit to U2_0 (task U2_U_R1, U2_U_R2, or U2_UR3) followed by a 1-h setup to one of the three other "units" (task U2_S_R1, U2_S_R2, or U2_S_R3). Similarly, unit U3 is split into four "units": U3_0, U3_R1, U3_R2, and U3_R3. Process data is shown in Tables E1 and E2.

Table E1. Scheduling Parameters for Integrated Example: Discrete Change due to Task Binary $N_{i,p}$

		τ_i	U1	U2_0	U2_R1	U2_R2	U2_R3	U4
	R_r^0		1	1	0	0	0	1
	R_r^{max}		1	1	1	1	1	1
$\mu_{i,r,0}$ (<0),	H	1	−1, 1					
$\mu_{i,r,\tau i}$ (>0)	U2_S_R1	1		−1	1			
	U2_P_R1	2			−1, 1			
	U2_U_R1	0		1	−1			
	U2_S_R2	1		−1		1		
	U2_P_R2	2				−1, 1		
	U2_U_R2	0		1		−1		
	U2_S_R3	1		−1			1	
	U2_P_R3	1					−1, 1	
	U2_U_R3	0		1			−1	
	S	2						−1, 1

Unit U3 (U3_0, U3_R1, U3_R2, U3_R3) is not shown because it is similar to unit U2. Initial and maximum inventory are also shown in the table.

Table E2. Scheduling Parameters for Integrated Example: Proportional Change due to Task Size $\xi_{i,p}$

		τ_i	V_i^{max}/V_i^{min}	RM1	RM2	RM3	HOT	INT1	INT2	IMP	A	B
kg	R_r^0			∞	∞	∞	0	0	0	0	0	0
	R_r^{max}			∞	∞	∞	100	80	80	100	∞	∞
$v_{i,r,0}$ (<0),	H	1	50/0	−1			1					
$v_{i,r,\tau i}$ (>0)	U2_P_R1	2	40/0		−0.5	−0.5		1				
	U3_P_R1	2	25/0		−0.5	−0.5		1				
	U2_P_R2	2	40/0				−0.4	−0.6	0.6		0.4	
	U3_P_R2	2	25/0				−0.4	−0.6	0.6		0.4	
	U2_P_R3	1	40/0		−0.2				−0.8	1		
	U3_P_R3	1	25/0		−0.2				−0.8	1		
	S	2	100/0						0.1	−1		0.9

Initial and maximum inventory, minimum and maximum task size are shown in the table.

Appendix F: Scheduling Feasible Regions: LR, CAP, NC1, and NCP2

Table F1. Convex Approximation of Projection (CAP, as expressed by $l \in L^{CAP}$), from Phase I

l	$\omega_A{}^l$	$\omega_B{}^l$	Φ^l
1	−1	0	0
2	−0.8865	0.4628	0.254
3	−0.8784	0.4779	2.755
4	−0.8370	0.5473	16.065
5	−0.8317	0.5553	17.648
6	0	1	171.563
7	0.8280	0.5608	173.483
8	0.8765	0.4814	164.839
9	0.8873	0.4612	162.544
10	0.9138	0.4061	157.176
11	0.9956	0.0939	122.608
12	0.9991	0.0430	118.428
13	0.9999	0.0148	116.787
14	1	0	116.000
15	0	−1	0

Table F2. Infeasible Polyhedrons $ip \in IP$ (as expressed by constraints $l \in L^{ip}$) for Model (NCP1), After Phase IV

ip	l	$\omega_A{}^l$	$\omega_B{}^l$	Φ^l	M^l
1	1	−0.998	−0.069	−119.388	1.230
	2	−1.000	0.000	−119.672	1.330
2	3	−0.857	−0.515	−110.000	6.000
	4	−0.997	−0.074	−139.500	32.063
	5	−1.000	0.000	−100.000	16.000
3	6	−0.712	−0.702	−118.739	19.190
	7	−1.000	0.000	−114.667	1.333
4	8	0.000	−1.000	−112.000	4.000
	9	−1.000	0.000	−153.000	18.563
5	10	0.000	−1.000	−106.000	10.000
	11	−1.000	0.000	−191.281	2.666
6	12	0.000	−1.000	−11.255	1.285
	13	−1.000	0.000	−144.646	23.995
7	14	−0.371	−0.929	−158.738	28.149
	15	−1.000	0.000	−96.000	20.000
8	16	0.284	−0.959	−148.500	23.063
	17	0.848	−0.529	−93.335	22.665
	18	0.860	−0.510	−7.169	1.637

Table F3. Feasible Polytopes $fp \in FP$ (as expressed by vertices $v \in V^{fp}$) for Model (NCP2), After Phase V

fp	v	P_A^v	P_B^v
1	1	116.000	0.000
	2	114.667	0.000
	3	114.667	72.218
	4	116.000	52.994
2	5	114.667	0.000
	6	112.000	0.000
	7	112.000	107.998
	8	114.667	71.993
3	9	100.000	0.000
	10	93.335	0.000
	11	93.334	153.000
	12	100.000	153.000

Table F3. (Continued)

fp	v	P_A^v	P_B^v
4	13	96.000	167.625
	14	96.000	0.000
	15	0.000	0.000
	16	0.000	0.549
	17	66.410	127.749
	18	96.000	167.625
	19	92.222	169.135
	20	96.000	167.626
5	21	112.000	0.000
	22	93.335	0.000
	23	93.334	131.449
	24	112.000	112.500
6	25	93.334	148.500
	26	93.335	0.000
	27	106.000	0.000
	28	106.000	148.500
7	29	76.666	146.601
	30	114.667	0.000
	31	67.561	129.955
	32	0.000	0.549
	33	0.000	0.000
	34	76.374	146.154
	35	114.667	89.991
	36	114.667	90.001
	37	80.040	147.602
8	38	93.335	0.000
	39	0.000	0.000
	40	0.000	0.549
	41	59.299	114.130
	42	93.333	171.563
9	43	72.754	139.500
	44	67.561	129.955
	45	0.000	0.549
	46	0.000	0.000
	47	110.000	0.000
	48	110.000	139.500

Table F4. Convex Approximation of RTN Linear Relaxation's Projection (LR)

l	ω_A^l	ω_B^l	Φ^l
1	−1	0	0
2	−0.8865	0.4627	0.273
3	−0.8755	0.4832	3.918
4	−0.8173	0.5763	24.097
5	−0.7335	0.6797	50.088
6	−0.4567	0.8896	114.288
7	0	1.0000	178.642
8	0.4668	0.8844	204.539
9	0.6539	0.7565	201.672
10	0.7290	0.6845	197.455
11	0.8070	0.5905	191.457
12	0.9686	0.2487	156.202
13	0.9845	0.1754	147.673
14	0.9934	0.1150	140.330
15	0.9966	0.0821	136.858
16	0.9991	0.0413	132.883
17	1	0	130.403
18	0	−1	0

Manuscript received Sept. 20, 2008, and revision received Dec. 28, 2008.