Minimization of Makespan with Discrete-Time State-Task Network Formulation

Christos T. Maravelias, Ignacio E. Grossmann[*]
Department of Chemical Engineering, Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract
A novel algorithm is proposed for the minimization of makespan of multipurpose batch plants using the State Task Network formulation. The algorithm involves the solution of successive feasibility problems, where the time horizon is extended until a feasible schedule for the given demand is found. To exploit the tight LP relaxation of the STN formulation, a production maximization problem (subject to the given demand) is solved at each iteration and the algorithm terminates when the first feasible solution is found. The algorithm is guaranteed to find an optimal solution, and if integer cuts are used it can provide many alternative solutions. The algorithm appears to be computationally efficient for many classes of problems and process networks of medium complexity.

1. Introduction

The State Task Network (STN) formulation was proposed by Kondili et al. (1993) to address the problem of short-term scheduling of multipurpose batch plants. The STN, and its equivalent Resource Task Network (RTN) (Pantelides, 1994) formulation, accounts for complex process network configurations (batch splitting/mixing and recycle streams), variable batch sizes, utility requirements, and various storage policies (NIS/FIS/UIS). The objective function is the maximization of profit over a fixed time horizon. In the discrete-time formulation proposed by Kondili et al. (1993) and refined by Shah et al. (1993) the processing times of tasks are assumed to be constant and the fixed time horizon is divided into time intervals of known duration equal to the greatest common factor of the processing times of all tasks. The assumption of constant processing times is not always realistic, while the length of the intervals may be so small that it either leads to a prohibitive number of intervals rendering the resulting model unsolvable, or else it requires approximations which may compromise the feasibility and optimality of the solution. This has led several authors to develop continuous-time STN/RTN formulations where the time horizon is divided into time intervals of unequal and unknown duration (Schilling and Pantelides, 1996; Zhang and Sargent, 1995; Lee et al., 2001; Maravelias and Grossmann, 2003a). While more realistic, continuous-time formulations are hard to solve mainly due to their poor LP relaxation, which is due to the big-M time matching constraints.

In terms of the objective functions, both discrete and continuous-time STN formulations behave moderately well when the objective function is the maximization of profit over a fixed time horizon. In short-term scheduling, however, the demand is usually fixed and more meaningful objectives are the minimization of makespan for a fixed demand, or the minimization of the production cost or the inventory for fixed demands with due dates. For the minimization of makespan, specifically, discrete STN formulations have never been used and continuous STN formulations behave poorly as shown in Maravelias and Grossmann (2003a and 2003b).

In this paper we propose a novel algorithm, based on the discrete-time STN formulation, for the minimization of makespan of multipurpose batch plants. To our knowledge, this is the first discrete-time approach for the minimization of makespan. While the known assumptions (constant processing times,

[*] To whom all correspondence should be addressed. E-mail: grossmann@cmu.edu

sequence independent setup times, etc.) and limitations (large number of binary variables, need for approximations of processing times) of discrete-time STN models are still present, the proposed model appears to be significantly faster than continuous-time STN models for many classes of problems.

2. Problem Statement

We assume that we are given the following items for a batch process:
(i) the available units and storage tanks, and their capacities
(ii) the available utilities and their upper limits
(iii) the production recipe (mass balance coefficients, utility requirements)
(iv) the processing time data
(v) the amounts of available raw materials and their delivery times
(vi) the minimum demand of each of the final products
The goal is then to find:
(i) the sequence and the timing of tasks taking place in each unit
(ii) the batch size of tasks
(iii) the allocated resources
(iv) the amount of raw materials purchased and the amount of final products produced
in order to minimize the makespan or completion time of the corresponding schedule.

3. Proposed Algorithm

3.1. MILP Model of STN

The proposed algorithm is based on the discrete STN formulation of Shah et al. (1993), where the binary W_{ijt} is equal to 1 if task $i \in I$ starts on unit $j \in J$ at time $t \in T$, B_{ijt} is the batch size of task i that starts on unit j at time t, S_{st} is the inventory level of state $s \in S$ at time t, R_{st}/D_{st} are the purchases/sales of state s at time t and U_{ut} is the level of consumption of utility $u \in U$ at time t.

$$\sum_{i \in I(j)} \sum_{t'=t}^{t-p_i+1} W_{ijt'} \leq 1 \quad \forall j, \forall t \tag{1}$$

$$V_j^{MIN} W_{ijt} \leq B_{ijt} \leq V_j^{MAX} W_{ijt} \quad \forall j, \forall i \in I(j), \forall t \tag{2}$$

$$S_{st} = S_{st-1} + \sum_j \sum_{i \in I(j) \cap SO(s)} \rho_{is}^O B_{ijt-p_i} - \sum_j \sum_{i \in I(j) \cap SI(s)} \rho_{is}^I B_{ijt} + R_{st} - D_{st} \quad \forall s, \forall t \tag{3}$$

$$0 \leq S_{st} \leq C_s \quad \forall s, \forall t \tag{4}$$

$$R_{st} \leq r_{st} \quad \forall s, \forall t \tag{5}$$

$$D_{st} \geq d_{st} \quad \forall s, \forall t \tag{6}$$

$$U_{ut} = \sum_j \sum_{i \in I(j)} \sum_{\theta=1}^{p_i-1} \left(\alpha_{ui} W_{ij(t-\theta)} + \beta_{ui} B_{ij(t-\theta)} \right) \quad \forall u, \forall t \tag{7}$$

$$0 \leq U_{ut} \leq U_u^{MAX} \quad \forall u, \forall t \tag{8}$$

$$\max Z = \sum_t \sum_s \zeta_s D_{st} \tag{9}$$

$$W_{ijt} \in \{0,1\},\ B_{ijt},\ S_{st},\ D_{st},\ R_{st},\ U_{ut} \geq 0 \tag{10}$$

Equation (1) is the assignment constraint that ensures that at most one task is assigned to an equipment j at any time. Constraint (2) bounds the batch size of a task within the lower V_j^{MIN} and upper V_j^{MAX} bounds of the corresponding unit. Equation (3) is the mass balance equation for state s at time t, where ρ^I_{is}/ρ^O_{is} are mass fractions for the consumption/production of state s by task i. Constraints (4) – (6) impose bounds on S_{st}, R_{st} and D_{st}, where C_s is the capacity of the storage tank for state s, r_{st} is the availability of state s at time t and d_{st} is the demand of state s at time t. The level of consumption of utility u at time t is calculated through equation (7) where α_{ui} and β_{ui} are the fixed and variable requirements of task i for utility u. The consumption of utility u at time t is bounded by the maximum availability U_u^{MAX} of utility u by constraint (8). The objective function in (9) is the maximization of income from sales, where ζ_s is the price of state s. The discrete-time STN model (M) consists of equations (1) – (10) and has a very tight LP relaxation.

3.2. Minimization of Makespan with STN formulations

In discrete-time STN models the number of time periods is fixed (i.e. set T has a fixed cardinality) and the last time point corresponds to the (fixed) time horizon of the problem. Thus it is not clear how to minimize the makespan. A simple solution is to define a new variable *MS* (makespan) that will be the objective function to be minimized, and for every binary W_{ijt} enforce the following condition:

$$W_{ijt}(t - p_i) \leq MS \quad \forall i, \forall j, \forall t$$

The resulting model, however, does not behave well.

In continuous-time MILP formulations, where each time point is not fixed, we can assign the last time point to a variable *MS* (makespan) and minimize variable *MS*. The computational performance of this model, however, is very poor (see Maravelias and Grossmann, 2003a and 2003b), and can only be handled in a reasonable way with hybrid Constraint Programming / Mixed Integer Programming methods.

3.3. Iterative Scheme

In this paper we propose an iterative scheme based on the discrete STN formulation. At the beginning we estimate a lower bound $CARDt=|T|$ of *MS* (i.e. a time horizon), and we solve a production maximization problem subject to meeting the fixed demands. We successively solve this problem by adding one additional time point at each iteration until a feasible schedule is found. The discrete STN model (M) that we use consists of equations (11)-(21):

$$\sum_{i \in I(j)} \sum_{t'=t}^{t-p_i+1} W_{ijt'} \leq 1 \quad \forall j, \forall t \leq CARDt \tag{11}$$

$$V_j^{MIN} W_{ijt} \leq B_{ijt} \leq V_j^{MAX} W_{ijt} \quad \forall j, \forall i \in I(j), \forall t \leq CARDt \tag{12}$$

$$S_{st} = S_{st-1} + \sum_j \sum_{i \in I(j) \cap SO(s)} \rho^O_{is} B_{ijt-p_i} - \sum_j \sum_{i \in I(j) \cap SI(s)} \rho^I_{is} B_{ijt} + R_{st} \quad \forall s, \forall t \leq CARDt \tag{13}$$

$$0 \leq S_{st} \leq C_s \quad \forall s, \forall t \leq CARDt \tag{14}$$

$$R_{st} \leq r_{st} \quad \forall s, \forall t \leq CARDt \tag{15}$$

$$U_{ut} = \sum_{j} \sum_{i \in I(j)} \sum_{\theta=1}^{p_i - 1} \left(\alpha_{ui} W_{ij(t-\theta)} + \beta_{ui} B_{ij(t-\theta)} \right) \quad \forall u, \forall t \le CARDt \tag{16}$$

$$0 \le U_{ut} \le U_u^{MAX} \quad \forall u, \forall t \le CARDt \tag{17}$$

$$S_{s,t=CARDt} \ge Dem_s \quad \forall s \in FP \tag{18}$$

$$W_{ijt} = 0 \quad \forall i, \forall j \in J(i), \forall t \le CARDt - p_i + 1 \tag{19}$$

$$\max Z = \sum_{s} S_{s,t=CARDt} \tag{20}$$

$$W_{ijt} \in \{0,1\},\ B_{ijt},\ S_{st},\ U_{ut} \ge 0 \tag{21}$$

Constraints (11)-(17) are similar to the constraints of the STN formulation of Shah et al. (1993), but in this model we assume that final products are sold only at the end of the time horizon (i.e. there are no due dates); note that the term D_{st} is not included in eq. (13) and that eq. (6) has been replaced by eq. (18) which enforces that the inventory level of each final product at the end of the horizon is greater or equal to the demand. The condition that no task is processed after the end of the scheduling horizon is enforced through constraint (19).

Given a demand for final products we solve model (M) as a production maximization problem, or equivalently as an inventory maximization problem (assuming that unlimited storage is available for final products) as in equation (20). Model (M) is solved iteratively, increasing the cardinality of set T by one, as shown in Figure 1. If model (M) is infeasible for the current scheduling horizon (i.e. for the current $CARDt=|T|$), the minimization of makespan problem is also infeasible because there is no feasible schedule for the postulated scheduling horizon that meets the given demand. Thus a longer scheduling horizon is needed and we continue by increasing the number of time points. Note that the first feasible solution found for problem (M), while probably suboptimal for (M), yields the optimal solution for the minimization of makespan problem. Hence, we do not have to prove the optimality of the first integer feasible solution and the algorithm terminates when the first feasible solution is found.

3.4. Estimation of Makespan

In order to solve few feasibility problems we propose a simple way to estimate a lower bound for the time needed to satisfy the given demand. As mentioned above the discrete STN has a very tight LP relaxation when the objective is the maximization of production. Thus, by solving the LP relaxation of model (M) for an initial (arbitrary) $CARDt^0$ we can get an estimate of the amount that can be produced over $CARDt^0$ time periods. Comparing this amount to the amount needed, we can get an estimate of the time needed to satisfy the given demand.

When solving this LP, we use constraint (22) instead of constraint (18), in order to keep the ratio of produced states close to the ratio of demands and thus get a better estimate.

$$ST_{st=CARDt^0} \ge R Dem_s \quad \forall s \in FP \tag{22}$$

where R is the ratio of the demand that can be satisfied in $CARDt^0$, and FP is the set of final products.
Furthermore, the objective is to maximize this ratio R and not the total production:

$$\max R \tag{23}$$

Thus, the model (M*) that we initially solve consists of equations (11) – (17), (19) and (21) – (23).

The procedure is then the following:

1. We solve model (M*) for $CARDt = CARDt^0$. Let R^* be the maximum ratio.
2. We calculate the time needed to meet the given demand if the throughput of the plant was completely linear: $CARDt^0/R$.
3. In order to ensure that $CARDt^1$ is always lower than the optimal makespan, we adjust the makespan calculated in step 2 by underestimating it by a factor f: $CARDt^1 = f\,(CARDt^0/R)$, where $f \in [0.7, 0.9]$.

A flow chart of the proposed algorithm is shown in Figure 1, where τ is the fixed duration of each time period.

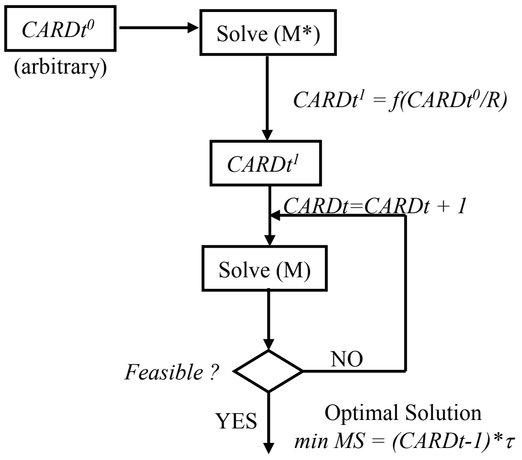

Figure 1: Proposed Algorithm

3.5. Remarks

The performance of the proposed algorithm can be enhanced in several ways. First, we can tune the parameters of the MILP solver according to the needs of our algorithm. Specifically, we can use the "feasibility" option that the latest versions of the MILP solvers offer (e.g. CPLEX, XPRESS). As explained above, the first feasible solution found in the course of the algorithm is the optimal solution for the makespan minimization problem. Thus, we are not interested in proving optimality in none of the stages of the algorithm. Instead, we are interested in quickly identifying infeasible MILP's and finding a single integer feasible solution for the first feasible MILP.
Our computational experience with both CPLEX and X-PRESS show that the option for emphasis in feasibility enhances the performance of the algorithm. Since it is important to identify infeasible models as soon as possible, we can add a constraint that explicitly enforces the total production to be more than the sum of the demands:

$$\sum_{i \in SO(s)} \sum_{j} \sum_{t} \rho_{is}^{O} B_{ijt} \geq Dem_s \quad \forall s \in FP$$

4. Examples

To illustrate the applicability and the computational effectiveness of the proposed algorithm we present three literature examples. The data for the three examples can be found online at http://pubs.acs.org as supporting information. In all examples we have used τ=*1* hour. Two instances of each example are presented. For the estimation of the makespan we have used *f = 0.8* in all examples. In examples 1 and 2 we used $CARDt^0=20$, while in example 3 we used $CARDt^0=40$ because the production network is more complicated and it takes longer to produce all products in large amounts.

The first example is from Kondili et al. (1993) (Figure 2). A minimum amount of 500 kg of product P1 and 400 kg of product P2 must be produced. The objective is to find the schedule of minimum makespan that satisfies the demand. The estimated lower bound is 27 hours and the optimal makespan of 36 hours is found in 10 iterations and 4.70 CPU seconds. The Gantt chart of the optimal solution is shown in Figure 3.

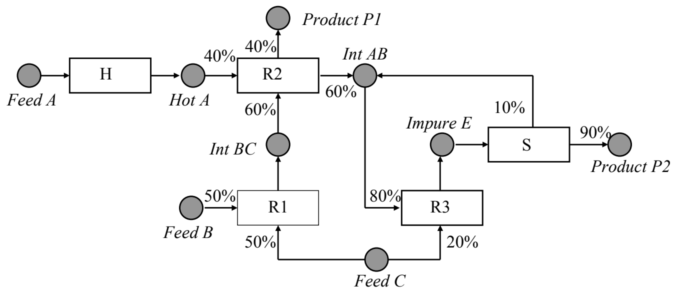

Figure 2: State Task Network of Example 1.

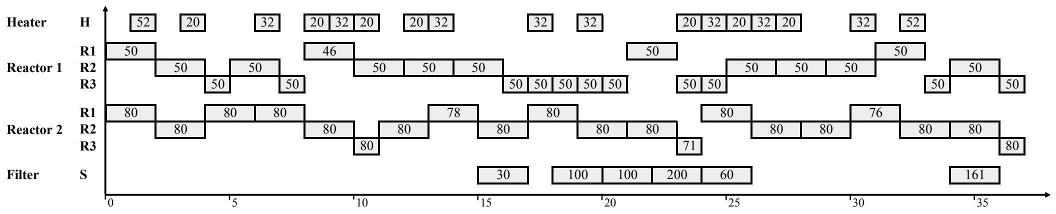

Figure 3: Gantt chart of optimal solution of Example 1.

We also solved Example 1 assuming that the demand for products P1 and P2 is 1,400 and 2,500 kg, respectively. The estimated makespan is 99 hours and the minimum one is 108 hours and was found in 11 iterations and 163.12 CPU seconds.

The process network of Figure 4 (Papageorgiou and Pantelides, 1996) is used for the production of final products P1, P2 and P3. The objective is to find the minimum makespan for the production of 10, 10 and 20 tons of products P1, P2 and P3 respectively. The optimal solution of 37 hours is found in 8 iterations and 198.24 CPU seconds. The Gantt chart of the optimal solution can be found in http://pubs.acs.org. For a demand of 40, 40 and 60 tons for products P1, P2 and P2, respectively, the minimum makespan is 100 hours, and the proposed algorithm requires 9 iterations and 9,637.57 CPU seconds.

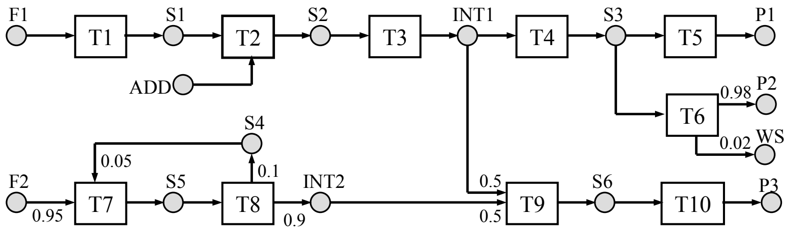

Figure 4: Process Network of Example 2.

The process network of Figure 5 is used for the production of products P1, P2, P3 and P4 (Papageorgiou and Pantelides, 1996). The objective is to find a schedule of minimum makespan that satisfies a demand of 20, 25, 20 and 15 tons for products P1, P2, P3 and P4, respectively. The optimal makespan of 44 hours is found in 9 iterations and 35.24 CPU seconds (the Gantt chart can be found in http://pubs.acs.org). The problem is also solved assuming that the demand for products P1, P2, P3 and P4 is 80, 100, 80 and 100 tons, respectively. The optimal solution with a makespan of 158 hours is found in 12 iterations and 158.16 CPU seconds.

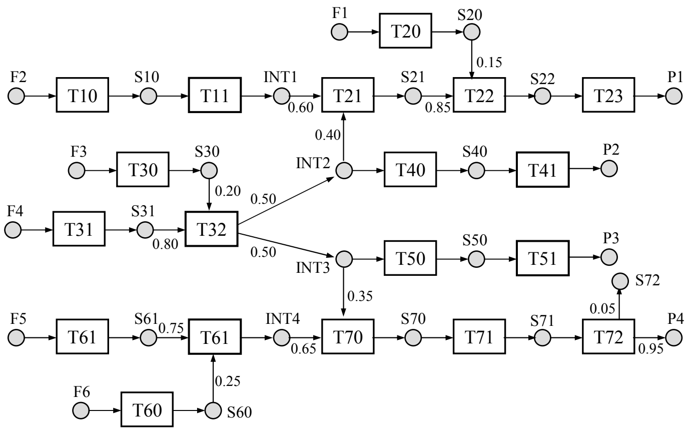

Figure 5: State Task Network of Example 3.

5. Computational Results

The computational times for the three examples are presented in Tables 1 and 2. A Pentium III PC at 1 GHz was used for all runs. All problems were modeled in GAMS and solved using CPLEX 7.5. A relative optimality criterion of 1%, a resource limit of 1,800 CPU seconds for each subproblem and the option for emphasis in feasibility has been used for all models.

Table 1: Computational statistics for the minimization of makespan.

	Example 1	Example 2	Example 3
Estimated Makespan	27	30	36
Optimal Makespan	36	37	44
Total CPU sec	4.70	198.24	35.24

As shown in Table 1, reasonable computational times are required for the solution of the small instances of the three examples, which means that the proposed algorithm can be used for the short-term scheduling of medium complexity process networks. The computational requirements, the solution statistics of the largest MILP model and the number of LP and integer infeasible MILP's for the large instances are reported in Table 2. Note that even the larger instances of Examples 1 and 3 are solved in less than three minutes. This is due to the fact that the LP relaxation of these examples is very tight. Thus, the first LP feasible model is also MILP feasible, i.e. only one MILP has to be solved. The LP relaxation of Example 2, on the other hand, is less tight. Many MILP infeasible models, therefore, are LP feasible and in order to prove infeasibility several nodes have to be examined. Since seven MILP's are integer infeasible (but not LP infeasible), extensive branch-and-bound trees are built seven times resulting in long computational time.

Table 2: Computational statistics for the minimization of makespan for the large instances.

	Example 1	Example 2	Example 3
Estimated Makespan (hours)	98	92	147
Optimal Makespan (hours)	108	100	158
Total CPU sec	163.12	9,637.57	158.16
LP infeasible MILPs	10	1	11
Integer infeasible MILPs	-	7	-
Solution Statistics of Last MILP			
CPU seconds	149.13	23.31	34.14
Nodes	5,411	404	99

Deleted:

6. Conclusions

An iterative algorithm for the minimization of makespan of batch plants with fixed demands, based on the discrete-time STN formulation has been proposed. The algorithm presented in this paper appears to be the first efficient discrete time STN method for the minimization of makespan of multipurpose batch plants with fixed demand. The application of the algorithm is illustrated through a number of examples. The results show that the algorithm is computationally efficient for many classes of problems and process networks of medium complexity.

Nomenclature

Indices

t	Time points
i	Tasks
j	Equipment units
u	Utilities
s	States
Sets	
FP	Set of final products
$I(j)$	Set of tasks that can be scheduled on equipment j
$SI(s)$	Set of tasks consuming state s
$SO(s)$	Set of tasks producing state s
Parameters	
H	Time horizon
p_i	Fixed duration of task i
α_{ir}	Fixed amount or utility r required for task i
β_{ir}	Variable amount of utility r required for task i
r_{st}	Maximum amount of state s available at time period t
d_{st}	Minimum amount of state s that should be delivered at time period t
Dem_s	Total demand for product s
ρ^I_{is}/ρ^O_{is}	Mass balance coefficient for the consumption/production of state s in task i
$S0_s$	Initial amount of state s
C_s	Storage capacity for state s
U_r^{MAX}	Upper bound for utility r
V_j^{MIN} / V_j^{MAX}	Lower/upper bounds on the batch size of task i
ζ_s	Price of state s

Binary Variables

W_{ijt}	$=1$ if task i starts at time point n

Continuous Variables

B_{ijt}	Batch size of task i that starts at time point n
S_{st}	Amount of state s available at time point n
U_{ut}	Amount of utility r utilized at time point n

Acknowledgements

The authors would like to gratefully acknowledge financial support from the National Science Foundation under Grant ACI-0121497.

References

Kondili, E.; Pantelides, C. C.; Sargent, R. A General Algorithm for Short-Term Scheduling of Batch Operations – I. MILP Formulation. *Comput. Chem. Eng.* **1993**, 17, 211-227.

Kyu-Hwang Lee; Heung Il Park; In Beum Lee. A Novel Nonuniform Discrete Time Formulation for Short-Term Scheduling of Batch and Continuous Processes. *Ind. Eng. Chem. Res.*, **2001**, 40, 4902-4911.

Maravelias, C.T.; Grossmann, I.E. A New General Continuous-Time State Task Network Formulation for the Short-Term Scheduling of Multipurpose Batch Plants. *Ind. Eng. Chem. Res.*, **2003**, 42 (13), 3056-3074.

Maravelias, C.T.; Grossmann, I.E. A Hybrid MILP/CP Decomposition Approach for the Short Term Scheduling of Multipurpose Batch Plants. Submitted for publication (2003).

Pantelides, C. C. Unified Frameworks for the Optimal Process Planning and Scheduling. *In Proceedings on the Second Conference on Foundations of Computer Aided Operations*. **1994**, 253-274.

Papageorgiou, L.G.; Pantelides, C.C. Optimal Campaign Planning/Scheduling of Multipurpose Batch/Semicontinuous Plants. 2. Mathematical Decomposition Approach. *Ind. Eng. Chem. Res.*, **1996**, 35, 510-529.

Schilling, G.; Pantelides, C. C. A Simple Continuous-Time Process Scheduling Formulation and a Novel Solution Algorithm. *Comput. Chem. Eng.* , **1996**, 20, S1221-1226.

Shah, N.; E.; Pantelides, C. C.; Sargent, R. A General Algorithm for Short-Term Scheduling of Batch Operations – II. Computational Issues. *Comput. Chem. Eng.* **1993**, 17, 229-244.

Zhang, X.; Sargent, R. W. H. The Optimal Operation of Mixed Production Facilities – General Formulation and Some Approaches for the Solution. *Comput. Chem. Eng.*, **1996**, 20, 897-904.