

A HEURISTIC ALGORITHM FOR SCHEDULING ‘BLOCKED OUT’ UNITS IN CONTINUOUS PROCESSING INDUSTRY : PRELIMINARY RESULTS

Sumit Kumar Bose and Subir Bhattacharya

Management Information Systems Group, Indian Institute of Management Calcutta, Joka, Kolkata -700104, India

Abstract: This paper addresses the problem of scheduling cascaded ‘blocked out’ continuous processing units separated by finite capacity storage tanks. The raw materials for the product lines arrive simultaneously on the input side of the first unit. However every unit can process only one product line at a time, thus giving rise to the possibility of spillage of raw material due to limited storage capacity. The need to process multiple product lines, and the added constraint of multiple intermediate upliftment dates aggravate the problem. This problem is quite common in petrochemical industry. The paper provides a MINLP formulation of the problem. However, for any realistic scheduling horizon, the size of the problem is too large to be solved by standard packages. We have proposed a heuristic propelled depth first branch and bound algorithm to help the planners in tackling the problem. The suggested algorithm could output near optimal solutions for scheduling horizons of 30 time periods when applied on real life situations involving 3 units and 3 product lines. The preliminary results reported here are quite encouraging.

Key words: Process scheduling, Heuristic search, Depth first branch and bound, Continuous processing

1. INTRODUCTION

The scheduling problem addressed in this paper occurs quite frequently in refineries and other chemical processing industries. In a refinery set up, for example, streams get split into intermediate products each of which then possibly moves through a network of processing units and blending points before being converted to finished products. As a result, the scheduling of a unit depends on the scheduling of upstream units and influences the scheduling of downstream units. Some of the units in the network may be responsible for producing a range of products. In this situation, the unit usually works in a ‘blocked-out’ fashion, i.e., at any point of time it processes only one product line or stream. However, arrival of raw materials for other products is simultaneous, being outputted from some upstream unit(s). Inputs for product lines, other than the one currently being processed, must either be stored in tanks for future processing, or it will get ‘spilled’ or wasted. The spilled amount usually gets downgraded to substantially lower valued products. Spillage can be reduced by quick changeovers. But a changeover from one product line to another has its associated cost and time. Thus, given fixed capacity storage tanks, there is always a trade-off between spillage and changeover while processing multiple products on the same processing unit. Sometimes, these blocked-out units are cascaded one after the other – separated by fixed capacity storage tanks for each product line – and thus complicates the scheduling decision further. The problem of scheduling gets aggravated by the requirement of meeting the upliftment schedule with its associated penalty for not producing the required amounts of finished products within the due dates. The objective of this paper is to study similar scheduling problems and to suggest heuristic propelled search scheme to get ‘near optimal’ solutions.

The processing units under consideration are continuous processing units. In a batch processing unit like a reactor, inputs in right amount and proportion are fed into the unit, ‘treated’ for a fixed amount of

time in a non-preemptive style, and outputs are obtained after the complete ‘batch’ gets processed. Scheduling of batch processing units addresses the issue of determining the optimal sequence in which the product batches should be produced. On the other hand, in a continuous processing unit, like a distillation column for example, inputs for a product line are fed in continuously at a fixed rate and the outputs flow out simultaneously at a fixed rate. Here, one product line can be processed in a number of stretches during the scheduling horizon interleaved with the processing of other streams. The length of a stretch is not fixed and each such stretch is called *run length* (see figure 1).

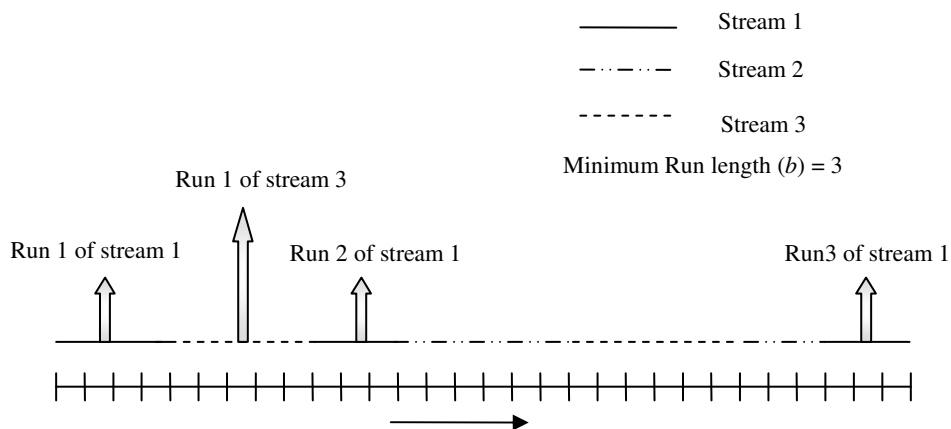


Figure 1. Example of Run Lengths in a unit

Longer stretches of run are desirable to reduce changeover cost, while shorter stretches may reduce spillage cost and upliftment failure penalty. The scheduling problem aims to find out an optimal mix of product sequences of varying run lengths so as to minimize the total cost/penalty. The authors encountered a similar problem while scheduling production of different categories of lube oils using a sequence of blocked-out units in a refinery.

Section 2 of the paper gives a broad review of the literature in this area. The problem scenario under consideration is explained in detail in section 3 and the corresponding mathematical model is discussed in section 4. Section 5 elaborates the heuristic solution procedure suggested for solving the problem. Our computational experience with the algorithm is summarized in section 6. Concluding remarks are presented in section 7.

2. LITERATURE REVIEW

The problem of scheduling in the context of process industry, as opposed to discrete manufacturing industry, has received considerable attention of the researchers only in the recent past. However, the bulk of the research work addresses the problem of scheduling batch processing systems. Work on scheduling of continuous processing systems is picking up only very recently. Rippin (1993) and Sargent (2004) describe the general status of process systems. Reklaitis (2000) outlines the generic families of existing methodologies for process scheduling operations. Shah (1998) extensively reviews the techniques available for optimizing the production schedules in the process industries. Grossmann et al. (1996)

discusses the mixed integer mathematical programming methods for the scheduling of batch processes. Kondili et al. (1993), in their seminal paper propose 'state task network' representation for batch processes and formulate a mixed integer linear program based on discrete time representation for handling different kinds of scheduling problems arising in batch processing plants. The computational issues related to the MILP formulation presented by Kondili et al. are discussed in Shah et al. (1993). Pinto and Grossmann (1998) summarizes the various assignment and sequencing models used in the process industry. Frauendorfer and Konigsperger (1996) discusses the various scheduling objectives and performance evaluation criteria in the context of the chemical process industry. Graham et al. (1979) and Garey and Johnson (1979) have shown that almost all the scheduling problems are NP-complete, implying that a large number of feasible solutions have to be examined before chancing upon a provably optimal solution. Thus efficient solution procedures can only result from applying problem specific heuristics. Kudva et al. (1994), Ku and Karimi (1990, 1991), Wiede and Reklaitis (1987a, 1987b), Blomer and Gunther (1998) have suggested application of heuristic techniques to various multi product batch scheduling problems. Schilling and Pantelides (1996), Zhang and Sargent (1996), Mendez and Cerda (2002) have developed continuous time formulations for the short term scheduling problem in batch processes. Ierapetritou and Floudas (1998) and Ierapetritou et al. (1999) formulate mathematical models for continuous processes and account for multiple intermediate due dates. Jain and Grossmann (1998) builds a mathematical model for cyclic scheduling of continuous processing plants with parallel units and decaying performance. Alle and Pinto (2004) develop a mathematical model for cyclic scheduling of multi product multistage continuous processing plants. Jia and Ierapetritou (2004) consider the short term scheduling of refinery operations and develop mixed integer models based on continuous time formulations. To the best of our knowledge there is no effective solution procedure in the scheduling literature to deal with the scheduling problem in multi stage, multi product continuous facilities with finite intermediate storage, multiple upliftment due dates and simultaneous arrival of inputs. This work is an attempt to advance the research in this direction.

3. PROBLEM SCENARIO

The set of units under consideration consists of a sequence of n 'blocked-out' processing units, $U_1, U_2 \dots U_n$, buffered by fixed capacity storage tanks to hold intermediate products (Figure 2). Each unit can process only one product line (also referred to as *stream*) at a time. The presence of intermediate storage tanks obviates the need for the units to process the same product in tandem. The finished products coming out of the last unit (U_n) also get stored in fixed capacity tanks to be uplifted according to some pre-specified upliftment schedule. The inputs for the product lines arrive at the input tanks of Unit U_1 simultaneously at constant rates. The rates at which these inputs arrive depend on factors that are beyond the control of this block of units. Each of these streams gets stored in a fixed capacity tank if there is room (*ullage*) in it; otherwise it spills, and is downgraded to lower valued products. Each product line has its exclusive set of tanks before and after every processing unit. Thus, each unit processes one product at a time taking its input from the corresponding input tank if it has enough stock and depositing the output into corresponding output tank if it has enough ullage. Note that spillage is not allowed for intermediate or finished products. In essence, spillage can occur only at the input of U_1 because the inputs for all the product lines are arriving simultaneously and also because these units have no control on the rates at which these inputs arrive.

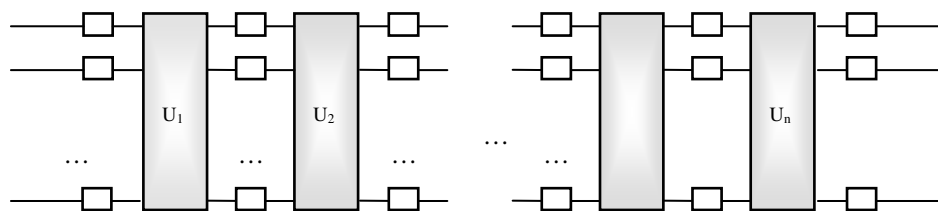


Figure 2. Continuous processing plant with n stages and m products

We consider m product lines, $P_1, P_2 \dots P_m$, each of which requires to be processed by the n units in succession. The processing capacity (known as the feedrate measured in MtPD, Metric tons per day) of a unit for a product line is considered fixed, but varies from product to product. The processing of a unit involves splitting the feed with the help of reagents into intermediate streams according to a yield percentage fixed for a product line for the unit. The intermediate streams that are relevant for the final products under consideration get deposited in the tanks on the output side. What happens to the other streams is beyond the purview of this discussion. For our purpose, if p_{ij} is the yield percentage corresponding to product P_i in unit U_j , then x unit of input to U_j gives rise to $x * p_{ij}$ units of output from U_j .

Given the above set-up, the problem is to schedule the production of the m products in each of the n units so as to minimize the total penalty. Three factors contribute to the penalty – the per unit spillage penalty for the raw material of each product line, the cost for changeover, and the per unit penalty for failing to meet the upliftment schedule.

4. THE MATHEMATICAL MODEL

The objective of the proposed model is as follows: given a scheduling horizon of H time periods, decide the product lines to be processed in each time period in each unit so as to minimize the total cost. A product i taken up for production in unit j must run for at least b consecutive time periods. This restriction is imposed to efficiently utilize the minimum amount of reagents that need to be added when changing over to product i . The number of consecutive time periods during which a product is processed (*run*) in a unit is called the *run length*. Thus, run length of a product in a unit must be greater than or equal to the minimum run length b . For simplicity, the products currently running at the units are assumed to have completed their minimum run lengths at the beginning of the scheduling horizon. That is, each unit is free to changeover to some other product if it so desires.

Changing over from one product to another requires some set up time. However, for simplicity, we assume the changeover to be immediate. The changeover costs are assumed to be sequence independent. Product-wise feedrates and yield percentages for the units are fixed data. The arrival rates of the inputs in front of (on the left of) U_1 and the upliftment schedule for the planning period are known at the time of deciding the units' schedule. The same product can be uplifted several times by different quantities during the given H time periods. The upliftment schedule may specify upliftment of more than one product at the same time period. All the units are assumed to be up and running for the entire scheduling horizon.

To facilitate our discussion, the storage tanks are assumed to belong to certain *levels*. The tanks in front of U_1 belong to 'level 0' and the tanks immediately after a unit U_k are said to be in 'level k'.

Table 1. Notations

Sets:

- i : Stream identifier, $i \in [1, 2, \dots, m]$
- j : Unit identifier, $j \in [1, 2, \dots, n]$
- t : Time period, $t \in [1, 2, \dots, H]$
- l : Level of tanks, $l \in [0, 1, 2, \dots, n]$

Parameters:

- r_i : Per unit spillage penalty of stream i .
- a_i : Rate of arrival of raw material of product line i .
- C_{il} : Capacity of tanks corresponding to product line i and level l .
- q_j : Cost of changeover from one stream to another in unit j .
- u_i : Per unit penalty for unfulfilled demand of stream i .
- p_{ij} : Percentage yield of stream i in unit j .
- dem_{it} : Quantity of stream i to be uplifted at the beginning of period t .

Variables:

- S_{it} : Quantity of material in tank for product i at level l at the start of time period t .
- Y_{ijt} = 1 if stream i is processed on unit j in time period t ; 0 otherwise.
- X_{it} = 1 if $S_{i0t} + a_i > C_{i0}$ and $Y_{ijt} = 0$; 0 otherwise.
- V_{ijt} = 1 if $Y_{ijt} = 1$ and $Y_{ijt+1} = 0$; 0 otherwise.
- W_{it} = 1 if $(dem_{it} - S_{int}) > 0$; 0 otherwise.

We will use the notations of Table 1 to formulate the problem. The short term scheduling problem of continuous processing blocked out units can be formulated as an MINLP as follows:

Minimize

$$Z = \text{Min} \sum_t \sum_i r_i (S_{i0t} + a_i - C_{i0}) X_{it} + \sum_t \sum_j q_j (\sum_i V_{ijt}) + \sum_t \sum_i u_i (dem_{it} - S_{int}) W_{it} \quad (1)$$

Subject to:

$$\sum_{t'=t}^{t-b+1} Y_{ijt'} \geq b V_{ijt} \quad \forall t, j, i \quad (2)$$

$$S_{i0t+1} = X_{it} C_{i0} + (1 - X_{it})(S_{i0t} + a_i) - f_{il} Y_{ilt} \quad \forall i, t \quad (3)$$

$$S_{ijt+1} = S_{ijt} + f_{ij} p_{ij} Y_{ijt} - f_{ij+1} Y_{ij+1t} \quad \forall i, t, j = 1, \dots, n-1 \quad (4)$$

$$S_{int+1} = (1 - W_{it})(S_{int} - dem_{it}) + f_{in} p_{in} Y_{int} \quad \forall i, t \quad (5)$$

$$0 \leq S_{il} \leq C_{il} \quad \forall i, l = 0, \dots, n \quad (6)$$

$$\sum_i V_{ijt} \leq 1 \quad \forall j, t \quad (7)$$

$$\sum_i Y_{ijt} \leq 1 \quad \forall j, t \quad (8)$$

Y_{ijt} s are the binary decision variables that define the scheduling of streams in the units. Y_{ijt} takes the value 1 if stream i is processed in unit j during time period t . The binary variables X_{it} , V_{ijt} and W_{it} represent spillage, changeover and upliftment failure respectively during time period t . If stream i spills during time period t , then X_{it} is 1, otherwise 0. The value of V_{ijt} is 1 when stream i is processed in unit j during time period t but is not processed during time period $t+1$, indicating changeover to some other stream i' in period $t+1$. W_{it} equals 1 when the amount of finished product i available at time t for upliftment falls short of the quantity demanded at that point of time.

The objective is to develop a schedule that would lead to the least overall cost. The terms in the objective function express the cost of spillage, the cost of changeover and the upliftment failure cost respectively. Equation (2) ensures that changeover cannot occur in a unit j till the *minimum run length* of the stream being processed in the unit is over. Equation (3) updates the stock positions in tanks dedicated for storing the raw materials to be used as feed to the unit, U_1 . The stock in the tank corresponding to stream i at the start of the time period $t+1$ is equal to the stock at the start of time period t adjusted by the amount of raw material arriving in period t , the spillage, if any, in period t and the withdrawal, if the stream is processed by unit U_1 in period t . Equation (4) updates the stock positions of the intermediate tanks at the beginning of each period. Equation (5) models the update of the stock positions of the finished product tanks. Equation (6) constrains the stock of a tank not to exceed the capacity of the tank at any point of time. Equation (7) ensures that a unit cannot change to more than one stream at any time period. Equation (8) ensures that at most one stream is assigned to a unit for processing during period t .

Employing linearization techniques the above mathematical model was reformulated as an MILP and solved using GAMS/XA on a 2.4 GHz Pentium 4 workstation with 1 GB RAM. Obtaining feasible schedules for horizons of more than 8 days (time periods), with $m=3$ and $n=3$, proved difficult even with very high values for resource count and iteration limit. Since the planners are frequently interested in deciding the schedule of the units for a longer period of time, say, 30 days, the above model, as of now, does not serve its purpose. To meet their requirement, we need to look for some other mechanism, which can provide 'reasonably good' solutions though not provably optimal. To that end we propose a heuristic algorithm below such that near optimal solutions can be obtained for scheduling horizons up to 30 days in reasonable time.

5. HEURISTIC SOLUTION PROCEDURE

As mentioned, the above mathematical formulation becomes computationally expensive for scheduling horizons of any practical significance. To take care of this problem, we propose below a heuristic depth-first branch-and-bound (DFBB) solution procedure for a simplified version of the problem. The scheme has been tried out using real-life data of a situation where there are 3 product lines ($m=3$) to be processed using 3 cascaded blocked-out units ($n=3$) for a planning horizon of 30 days ($H=30$). Our computational experience in section 6 will demonstrate that this heuristic scheme can give optimal/near-optimal solutions in reasonable time for real life situations.

The search space is a tree where a node represents a feasible partial schedule till the time period t . A feasible partial schedule till a period t contains feasible assignments of product lines (*streams*) to the processing units for every time period up to and including t . An assignment of a stream to a unit at any time period is said to be *feasible* if the stream has enough input stock and output ullage for processing the stream in this unit for one time period. A partial schedule till period t can be extended to partial schedules till period $t+1$ by feasibly assigning streams to the units for the $(t+1)^{\text{th}}$ time period, given the assignments till period t . Thus, given 3 streams and 3 units, a node in the search tree can give rise to a maximum of 27 child nodes, all of which, however, may not be feasible.

The root node of the search tree represents partial schedule till period 0. A leaf node is one which contains partial schedule till the scheduling horizon H and hence represents a feasible solution to the scheduling problem. The tree is unveiled and traversed in a depth-first manner in its quest for the solution node with the least cost.

The above search tree can have a maximum of 27^{30} leaf nodes for a 30 period scheduling horizon. Obviously, an exhaustive search scheme is not computationally tenable. We augment the depth-first search using the following mechanisms:

- a) An admissible heuristic for the upliftment failure penalty;
- b) An admissible heuristic for the spillage cost;
- c) An efficient node ordering; and
- d) A 'reasonably good' initial solution that provides a tight upper bound.

In the sub-sections below we elaborate on each of the above schemes.

The heuristic scheme discussed in this paper does not take into account the changeover cost. That is, the cost function being minimized is the sum of spillage costs and the upliftment failure penalties only. As explained below, our heuristics for spillage cost and upliftment failure penalty at a node give us good lower bound on the total cost of the potential extensions of a partial schedule and, hence, efficiently prune provably sub-optimal branches of the search tree. However, given a partial schedule, we have so far not been able to come up with a good admissible heuristic to get a lower bound on the number of changeovers for the extensions of the schedule. Work is currently going on to factor in changeover cost as well. This paper discusses the efficacy of the rest of the scheme and preliminary results obtained thereon.

5.1 Heuristic for Upliftment failure penalty

Upliftments for the various finished products are met from the output tanks of the last unit, U_3 . Upliftment time periods and quantities are known a priori for the entire scheduling horizon. Please recall that dem_{it} is the quantity of the i^{th} stream to be uplifted at the beginning of time period t . Material in stock, S_{i3t} , in the output tank of the i^{th} stream at the end of time period $t-1$ is considered available for meeting the upliftment at the t^{th} time period. If the stock contains more than dem_{it} , only dem_{it} amount is uplifted and if it is less, S_{i3t} is the amount uplifted and the penalty incurred is $(dem_{it} - S_{i3t}) * u_i$ where u_i is the penalty per unit for the shortfall for the i^{th} product. Shortfall in one upliftment cannot be compensated for by providing more during subsequent upliftments.

Let us assume that we have a partial schedule till time period $t-1$. To keep our heuristic admissible, we assume that there is no dearth of stock for any of the streams on the input side of unit U_3 . That is, U_3 can process any stream by any amount so long as there is ullage at the output tanks. It is easy to see that, during actual processing, penalty can only go up due to non-availability of the corresponding input stocks. The calculation of the heuristic consists of building possible scenarios of meeting the upliftments by assigning periods of U_3 processing to various streams, calculating the cost of each of the scenarios, and picking up the least cost one. It is worth bearing in mind that assigning time periods to a stream has nothing to do with the scheduling of the stream which includes sequencing as well. It just sets aside a number of time periods for processing of the stream.

To explain the process of calculating the heuristic for upliftment failure penalty we will use the notations given in Table 2 for each finished product (stream) i .

Table 2. Notations for Heuristic Calculations

dt_{ik} : Upliftment time period for stream i for the k^{th} remaining upliftment
n_i : Number of remaining upliftments of stream i on time periods $dt_{ik} (dt_{ik} > t)$, for $k = 0, 1, 2, \dots, n_i$
h_{ik} : Time periods available for meeting the k^{th} upliftment of stream i , that is number of unassigned time till dt_{ik}
h : Time periods available till the last upliftment in the schedule.
d_{ik} : The number of U_3 processing time periods required to meet the k^{th} upliftment given the tank stock after $(k-1)^{\text{th}}$ upliftment. d_{ik} need not necessarily be an integer.
\hat{d}_i : Total amount of U_3 processing time periods required to meet all the remaining upliftments. \hat{d}_i need not necessarily be an integer.

We start by assigning periods for the ‘most vulnerable’ stream. The ‘most vulnerable’ stream is the one for which the product $f_{i3} * p_{i3} * u_i$ is maximal among all the streams. Conceptually, if all the finished products require one more period of processing and only one period of processing is available, then the processing of this stream will minimize the total upliftment penalty. Streams are taken up for assignment in decreasing order of vulnerability. For a stream currently under consideration, we keep on assigning processing periods for meeting the quantities needed for *all* its upliftments starting from the next upliftment. When all the upliftments of a stream are taken care of, we repeat the same process for the next vulnerable stream and so on.

We now elaborate the scheme for developing the possible scenarios for assigning time periods. Let us assume, for the time being, that the stream running in U_3 at $t-1$ has completed its minimum run length b , that is, U_3 is free to changeover to some other stream if it so desires. Let us assume that currently we are considering the 1st upliftment of the i^{th} stream after time period t , and i^{th} stream was not being processed in U_3 at $t-1$. Since there *has* to be a changeover to stream i to meet its 1st upliftment (assuming $d_{i1} > 0$), we can exploit the fact that any assignment of periods to stream i must be at least for the minimum run length b . d_{i1} , the number of processing days required, given the output tank stock at the beginning of period t , need not necessarily be (and is usually not) an integer. The number of periods to be assigned will depend on many factors, combinations of which give rise to the various scenarios. If i^{th} stream is not the one that was being processed in time period $t-1$, then the leaf nodes of figure 3 give all the possible assignments that need to be considered. For example, if d_{i1} is less than or equal to the number of available periods for processing the i^{th} stream for the 1st upliftment (h_{i1}), we need to first check whether the demand is enough to be assigned for minimum run length b . If yes, we have to try two scenarios – one by assigning $\text{ceil}(d_{i1})$ periods and the other with $\text{floor}(d_{i1})$. However, if the needed processing periods is less than the minimum run length b , we need to consider \hat{d}_i , the total number of processing periods required for meeting all the remaining upliftments. If the total requirement is also less than or equal to b , we need to try two scenarios by assigning 0 and b time periods to stream i . Otherwise, depending on whether \hat{d}_i is greater than two times b or not we will either try two scenarios by assigning 0 or b days, or we will construct four scenarios assigning 0, b , $\text{ceil}(\hat{d}_i)$ and $\text{floor}(\hat{d}_i)$ periods to stream i .

For apportioning processing periods for subsequent upliftments of the same stream i , however, we cannot impose the restriction of minimum run length since it will depend on actual sequencing of the streams. For subsequent upliftments, we just compare $d_{ik} (k > 1)$ with corresponding h_{ik} . If d_{ik} is greater than h_{ik} , we assign h_{ik} time periods to stream i , else we consider the scenarios of assigning $\text{ceil}(d_{ik})$ and $\text{floor}(d_{ik})$. We have to follow the same method to generate scenarios for meeting upliftments of the stream that was being processed on period $t-1$.

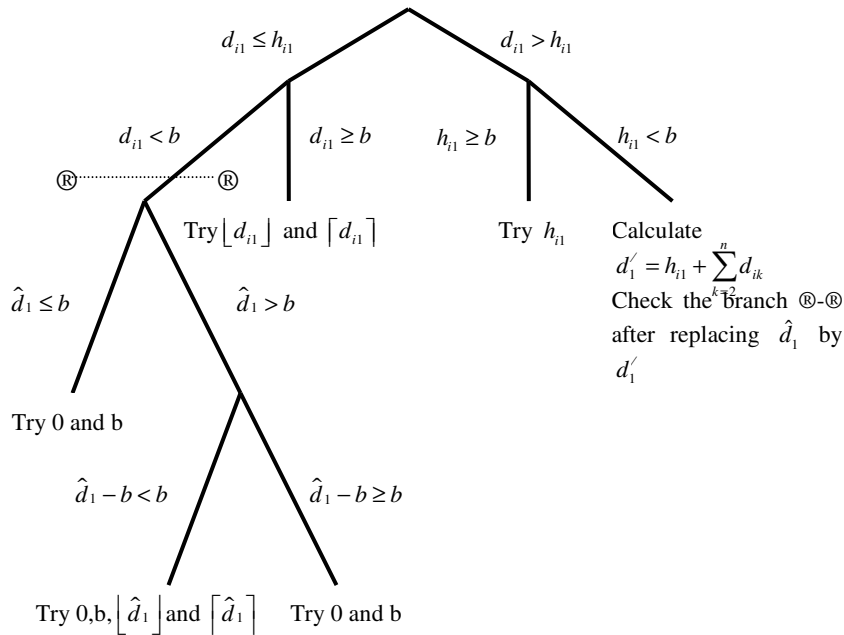


Figure 3. Decision tree for heuristic upliftment

After each assignment of time periods to an upliftment, we need to carefully update the available processing periods for the upliftments of all other streams, which are yet to be taken up (less vulnerable). For example, assume that, at $t=1$, there remains only one upliftment of the most vulnerable stream i at $t=9$ requiring 6 processing periods and only one upliftment of less vulnerable stream i' at $t=5$ requiring 3 processing periods. That is, initially, $h_{i1}=8$, $d_{i1}=6$ and $h_{i'1}=4$, $d_{i'1}=3$. We first take up stream i , and assign 6 processing periods to it. In that scenario, the upliftment of stream i' at $t=5$ will have only 2 periods available for processing, i.e., $h_{i'1}$ becomes 2. However, if the upliftment of stream i required 4 processing periods or less, then after the assignment of stream i , $h_{i'1}$ will still be 4.

For every scenario that assigns $\text{ceil}(d_{ik})$ processing period for an upliftment, we need to adjust the required processing period of the next upliftment, if any, of stream i as the extra amount beyond the requirement of the j^{th} upliftment will be available to meet the $j+1^{\text{th}}$ upliftment of the same stream.

It is worth noting that we need not calculate the upliftment penalty heuristic afresh at time period t if the stream running on $t-1$ period at U_3 has not completed its minimum run length.

5.2 Heuristic for Spillage cost

Suppose we changeover to a stream i at time period t at unit U_1 , i.e., some stream other than i has been running in U_1 on and till time period $t-1$. Since every stream scheduled in a unit must run for the minimum run length b , stream i , once scheduled on time period t must run till time period $t+b-1$. Given the ullage positions of the input tanks of U_1 at time period t and the fixed incoming rate of the raw materials, we can calculate the time periods when the other two streams will start spilling if not processed. The heuristic for spillage cost has two components. First, if a stream (or streams) starts spilling within time period $t+b-1$, we calculate the spillage amounts till that period and the corresponding spillage costs. This is the spillage cost that will definitely be incurred during this period. Next we check whether

more than one stream will start spilling in the period between $t+b$ to $t+2b-1$ (known as *closepill* in refinery parlance). If the answer is yes, then we would definitely be incurring spillage of at least one of the streams. To keep our heuristic admissible, we calculate the spillage amount of each stream during the period $t+b$ to $t+b-1$ and the corresponding cost. The lower of the two is added to the heuristic spillage cost.

If the stream scheduled on t is same as that scheduled on $t-1$ and it has completed its minimum run length on or before $t-1$, closepill needs to be checked only during the period $t+1$ to $t+b$ since the unit is free to changeover to some other stream in the next time period. However, if the stream scheduled on t is yet to complete its minimum run length b , then its spillage cost remains the same for all the periods till it has run for b periods.

As mentioned earlier, every node in the search tree represents a partial schedule till time period t . This is obtained by extending the partial schedule till time period $t-1$ of the parent node by feasible assignments of streams to the three units for the time period t , given the assignments till $t-1$. For every such node we update the cost incurred so far till period t , and calculate the heuristic upliftment penalty and spillage costs for the rest of the scheduling horizon. If the sum of the cost incurred and the heuristic costs is more than or equal to the currently best known cost of a complete solution, then the sub-tree rooted at this node need not be explored further.

5.3 Node Ordering

While extending a partial schedule till period $t-1$ to period t , in which order should we try assigning the feasible streams to a unit in period t ? A pre-determined, static sequence of the streams may give rise to unnecessary search and may delay reaching the optimal solution. It would make better sense to look below the most potential assignment first which, hopefully, will lead us to a lower cost, and hence induce better pruning during the rest of the search. For example, consider assigning streams in unit U_3 . If the most vulnerable stream has an upliftment for which the processing periods required is less than the available periods before the upliftment, it would be prudent to try out this stream first before other streams. For this purpose, before expanding a node – that is, before extending a partial schedule – we first dynamically order the feasible streams for a unit based on the information available at that point in time. We do this for unit U_1 and unit U_3 in the following manner.

For unit U_1 , if the minimum run length for the stream i running in period $t-1$ is not yet complete, then stream i is the only stream to be tried in period t . Let us consider the situation where stream i has completed its minimum run length at time period $t-1$ or before. Let us assume that all the streams are feasible, that is, stream i has input stock and output ullage for one more time period and the other two streams have input stocks and output ullages for b periods. We first check whether the other two streams currently not being processed have a closepill in near future, that is, whether both the streams will start spilling within a gap of b periods if not processed in next $2b$ periods. If there exists any such closepill, the stream that will spill first will be tried first followed by the other stream followed by stream i . If there is no possibility of closepill, we will first try stream i for one more period (to have longer stretches of runs) followed by the stream that will spill first followed by the remaining one.

While ordering nodes for unit U_3 , we first calculate \hat{d}_i , the total number of processing periods required for each stream i to meet all the subsequent upliftments of the stream. If the summation of \hat{d}_i^s is more than the available processing periods h , we adjust the \hat{d}_i^s by reducing periods of processing of the streams starting from the least vulnerable stream. Since available processing periods h is an integer and the \hat{d}_i^s are, in general, not integers, we consider all possible combinations of floor and ceiling functions of reduced \hat{d}_i^s that add up to h , and pick up that combination that gives minimum upliftment failure penalty. This is necessary so that we do not waste one period of production to meet, say, remaining demand worth 0.1 period of processing of some stream. With only three streams, the combinations are not too many.

Given the adjusted integer \hat{d}_i^s , we order the streams for trial in the following manner. Streams with adjusted \hat{d}_i^s greater than zero having unmet demand in the next upliftment will be tried in order of decreasing vulnerability. Next to try are the streams with unmet demand in the next upliftment with no regard to their adjusted \hat{d}_i^s . If all streams have not been included, we next consider streams with \hat{d}_i^s greater than zero irrespective of their upliftment periods. If that too fails to put all the streams in the sequence, we next consider streams with \hat{d}_i^s equal to zero as well.

5.4 Initial Solution

The DFBB algorithm was fed with an initial solution which provided an upper bound on the optimal solution of the problem. The following is a broad overview of the deterministic algorithm that was used to obtain the initial solution. Conceptually, the algorithm considers unit U_1 alone responsible for minimizing the spillage cost, unit U_3 alone responsible for reducing the upliftment failure penalty, and unit U_2 as a service unit trying to help out U_1 and U_3 in meeting their objectives. Each unit obeys the minimum run length restriction. The algorithm first calculates the minimum spillage that is definitely going to be wasted, and the minimum upliftment penalty that is definitely going to be incurred irrespective of the sequencing of the streams in the units. The input rates, the initial stocks at level 0 and the feed rates for U_1 are used to get the spillage estimates, and the processing capacity of U_3 , the initial stocks at level 3 and the upliftment schedule are used to estimate the minimal upliftment penalty.

The algorithm incrementally builds the schedule by assigning streams to units in successive time periods. For each unit, this decision is guided by a decision tree developed for the unit. At any given time period t , U_1 will be the first to decide the stream to be run at t . Since goal of U_1 is to reduce spillage, it may find it necessary to schedule a stream which does not have sufficient output ullage for the purpose. In that case, it logs in a request to U_2 to changeover to that stream, if possible. It also decides what it should run at t , if the request gets rejected. The next unit to choose its stream to be run at t is U_3 . U_3 decision tree is geared to meet the upliftment schedule. If U_3 thinks it fit to run a stream for which it does not have enough input stock to sustain minimum run length, it can log in a request to U_2 to changeover to that stream, if possible. Like U_1 , U_3 also keeps an alternative ready in case the request gets rejected. U_2 is the last to choose its stream for period t . It rejects the request(s) from U_1 and/or U_3 , in case the stream running in U_2 at $t-1$ is yet to complete its minimum run length. However, if it has no such restriction, it tries to changeover to the requested stream. If request has come from only one side, say U_1 , and it can be honored (for example, there is enough ullage on the output side of U_2 for the requested stream for the minimum run length), U_2 does change over to the requested stream. However, if U_2 has received requests at time t from both U_1 and U_3 to change over to different streams, it checks the spillage and upliftment penalty costs incurred till time period t and compares them with the corresponding least costs that have been calculated at the beginning. Based on the deviations, U_2 decides to honor one of the requests and rejects the other. If there is no request from either side, U_2 follows its own decision tree to choose the run for time period t .

6. COMPUTATIONAL EXPERIENCE

The proposed DFBB algorithm was coded in C++ and was run on 2.4 GHz Pentium 4 workstation with 1 GB RAM. The fixed data (Table 3) used for the test cases are real data collected from a refinery. The per unit spillage costs and the per unit upliftment failure penalties have been masked. However, their relative importances have been reflected in the values used. The capacity of each storage tank is assumed to be 10000 Mt for easy comparison of the input data for the cases considered.

Given the fixed refinery data in Table 3, the algorithm has been tried out in a total of 25 cases. For each case, the scheduling horizon was 30 time periods (days). The input of a case consists of the initial stock positions at all the four levels at the beginning of the scheduling horizon, the upliftment schedule, and information about the streams running in the units on the day just before the start of the scheduling horizon.

Table 3. Fixed Data

Number of Units: 3 { U_1 : FEU; U_2 : SDU; U_3 : HFU}									
Number of Streams: 3 {1:IO; 2:HO; 3:DAO}									
Minimum Run Length (b): 3									
Capacity (Mt): 10000 (For each stream in each level)									
Scheduling Horizon(H): 30									

Stream	Input Rate (MtPD)	Feed Rate(MtPD)			Yield Percentage			Spill Penalty	Upliftment Penalty
		FEU	SDU	HFU	FEU	SDU	HFU		
1	620	1250	780	546	60	70	99	10	14
2	255	1030	576	403.2	50	70	99	12	12
3	310	1050	521	364.7	65	70	99	14	10

Table 4. Case data

Tank Stock Positions

Streams	Stock Positions (Mt)			
	Level 0	Level 1	Level 2	Level 3
1	7000	3000	2000	1000
2	9000	5000	5000	1000
3	9500	5000	5000	2000

Scheduled Upliftments (Mt)

Day	Streams		
	1	2	3
15	5000	3200	1500
28	6000	0	0

Last Run: (Unit:Stream:Run) : (U_1 :1:3); (U_2 :2:3); (U_3 :3:3)

To generate these test cases, we first designed a base case (Table 4). The inputs of the base case are chosen in a manner so as to require collaboration among the units to come up with the least cost schedule. A look at Table 4 data would reveal that unit U_3 needs to process two streams (streams 1 and 2) to fulfill the upliftments, and the intermediate tank stocks for these streams are such that the U_3 requirements for feed would necessitate processing of the corresponding streams in U_1 . Again, the initial tank stocks at level 0 are chosen to create a situation for close spill. The Input Rate column (Table 3) and the Level 0 stocks (Table 4) considered together, show that streams 1, 2 and 3, if not processed in U_1 , would start spilling on day 5, day 4 and day 2 respectively. It may be noted that the scheduled upliftment of stream 3 can be fully met from the stock available at level 3, and as such stream 3 does not require processing in U_1 from upliftment point of view.

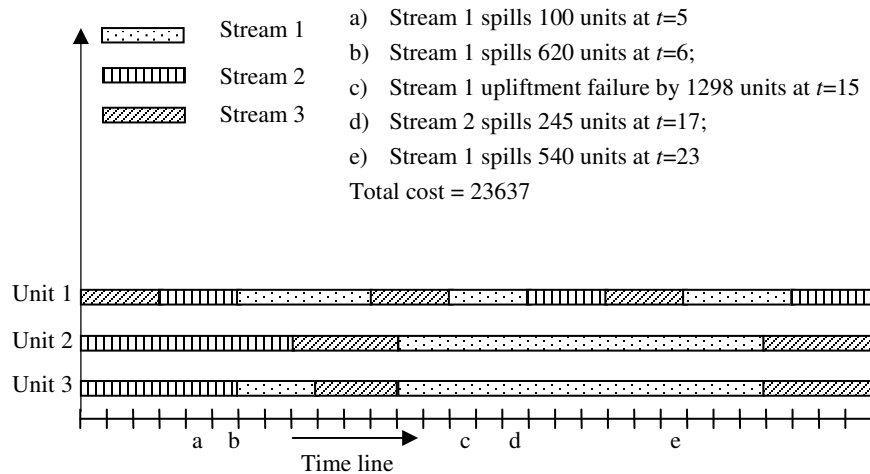


Figure 4. Gantt chart for the Initial Solution

Figure 4 is the Gantt chart of the Units' schedules as suggested by the initial solution while figure 5 shows the schedules as suggested by the DFBB scheme with solution values of 23637 and 5204 respectively. The initial solution acted in a greedy manner. U_1 started with stream 3 to take care of the close spill, and in the process, spilled stream 1 on days 5 and 6. U_3 was forced to run stream 3 on day 10 since there was not enough input stock for the other streams. The improvement of 78% over the initial solution was achieved by the DFBB in about 8 minutes after expanding 6.3 million nodes with 10 updates. The optimal solution was found after expanding 5 million nodes but the algorithm needed to expand 1.3 million more nodes before concluding the value to be the optimal one. The algorithm solves the problem by judicious allocation of days to streams in units. It continued processing stream 1 in U_1 on the first day to create cushion for delaying its spillage and also to meet the requirement of stream 1 in U_3 , switched to stream 3 on day 2 to avoid its spillage, allowed stream 2 to spill for some time till it could switch over to it on day 5. The solution manages to strike a balance between spillage and upliftment failure by quick changeovers in U_1 in a sequence that helps U_3 to fulfill its upliftment schedule to the extent possible.

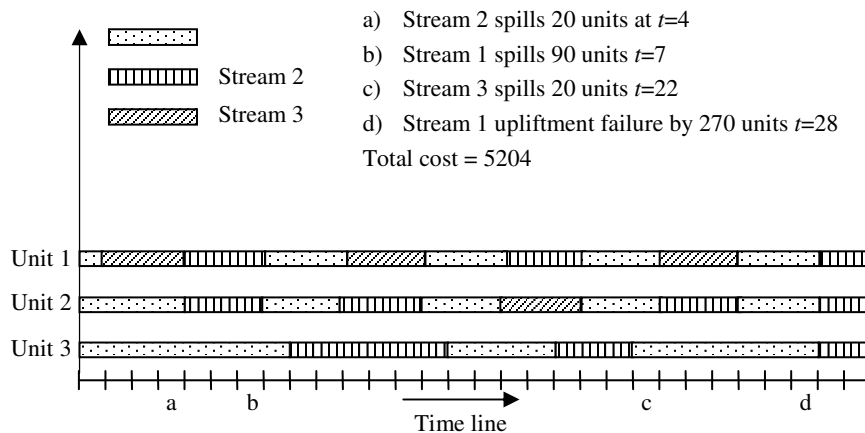


Figure 5. Gantt chart for the DFBB solution

Table 5 gives the details of the 25 case instances that include the base case as Case 1. These cases were generated by varying the input parameters of the base case. Each case was obtained by altering the initial stock positions and/or the streams to be uplifted along with their amounts. Table 6 summarizes our experiences with the DFBB scheme when applied to these 25 cases. The algorithm was run on each case with a maximum time limit of one hour. The initial and the DFBB solutions have been rounded to nearest integers. The initial solution was optimal in three cases with a cost of 0. The DFBB algorithm could output optimal solutions in 17 of the remaining 22 cases. In each of the five cases where the algorithm could not terminate in the stipulated one hour, the last update was a substantial improvement over the initial one.

Table 5. Case Instances

Case No.	Initial Stock Position ('00 Mt)				Scheduled Upliftments ('00 Mt)				
	Levels				Day	Streams			
	0	1	2	3		1	2	3	
1	70	30	20	10	15	50	32	15	
	90	50	50	10		28	60	0	0
	95	50	50	20					
2	70	30	20	10	15	32	50	15	
	90	50	50	10		28	60	0	0
	95	50	50	20					
3	70	30	20	10	15	32	15	50	
	90	50	50	10		28	60	0	0
	95	50	50	20					
4	70	30	20	10	15	15	32	50	
	90	50	50	10		28	60	0	0
	95	50	50	20					
5	70	30	20	10	15	15	50	32	
	90	50	50	10		28	60	0	0
	95	50	50	20					
6	70	30	20	10	15	15	50	32	
	90	50	50	10		28	0	0	60
	95	50	50	20					
7	70	30	20	10	15	15	50	32	
	90	50	50	10		28	0	60	0
	95	50	50	20					
8	50	30	20	10	15	50	32	15	
	70	50	50	10		28	60	0	0
	65	50	50	20					
9	50	60	20	10	15	50	32	15	
	70	5	50	10		28	60	0	0
	65	20	50	20					
10	50	60	40	10	15	50	32	15	
	70	50	10	10		28	60	0	0
	65	20	20	20					
11	50	50	50	10	15	50	32	15	
	50	50	50	10		28	60	0	0
	50	50	50	20					
12	50	50	50	10	15	32	15	50	
	50	50	50	10		28	60	0	0
	50	50	50	20					
13	50	50	50	10	15	32	15	55	
	50	50	50	10		28	0	0	60
	50	50	50	20					

Case No.	Initial Stock Position ('00 Mt)				Scheduled Upliftments ('00 Mt)				
	Levels				Day	Streams			
	0	1	2	3		1	2	3	
14	50	50	50	10	15	32	15	0	
	50	50	50	10		28	0	55	60
	50	50	50	20					
15	50	70	50	10	15	32	15	0	
	90	50	50	10		28	0	55	60
	55	50	80	20					
16	70	30	20	10	15	32	15	0	
	90	50	50	10		28	0	55	60
	95	50	50	20					
17	70	80	40	10	15	50	32	15	
	90	70	50	10		28	60	0	0
	95	60	80	20					
18	70	80	40	10	15	50	32	15	
	20	70	50	10		28	60	0	0
	95	60	80	20					
19	70	80	40	10	15	50	32	15	
	90	70	50	10		28	60	0	0
	15	60	80	20					
20	70	80	40	20	15	50	32	15	
	90	70	50	10		28	60	0	0
	85	20	10	0					
21	87	10	10	10	15	50	32	15	
	90	10	10	10		28	60	0	0
	95	10	10	20					
22	60	10	10	10	15	50	32	15	
	90	10	10	10		28	60	0	0
	95	10	10	20					
23	50	10	10	10	15	50	32	15	
	50	10	10	10		28	60	0	0
	50	10	10	20					
24	50	10	10	10	15	50	32	15	
	90	10	10	10		28	60	0	0
	95	10	10	20					
25	50	20	10	10	15	50	32	15	
	90	10	10	10		28	60	0	0
	95	10	10	20					

Table 6. Case Results

Case No.	Initial Solution	DFBB Solution	Time	Number of Updates	Total nodes expanded	Node count at last update
1	23637	5204	7m59.3s	10	6306816	5095442
2	23637	2050	17.9s	11	139304	19925
3	31205	3746	0.14s	14	1392	1380
4	33314	6146	0.2s	13	826	814
5	30964	2688	0.2s	11	1143	1093
6	33261	19141	49.7s	7	66905	18504
7	32370	20408*	9m20.5s [†]	4	DNT	882 [†]
8	18162	0	0.01s	2	37	37
9	18919	0	0.01s	1	31	31
10	0	0	0.01s	0	1	0
11	0	0	0.01s	0	1	0
12	34632	5002	6.1s	5	75882	146
13	53811	18488	5m20.5s	10	2778764	343
14	35500	22097*	42m31.1s [†]	4	DNT	19687292 [†]
15	24100	22097*	9m53.3s [†]	1	DNT	31 [†]
16	44430	18995*	10m19.6s [†]	14	DNT	223951 [†]
17	50551	1420	12.26s	28	110979	7844
18	23384	0	2.50s	18	139714	139714
19	0	0	0.01s	0	1	0
20	25297	11758*	5m35.6s [†]	6	DNT	951 [†]
21	70109	52027	19.34s	8	462315	456489
22	69864	39407	28.2s	13	684973	167125
23	51639	3784	4.07s	14	95494	91218
24	71892	39407	57.2s	19	1438665	353513
25	71892	24271	19.4s	12	433665	84929

*DFBB solution at the last update count.

[†]Time and Total nodes expanded till the last update.

DNT: Did not terminate in 1 hour.

7. CONCLUDING REMARKS

In this paper we have suggested a heuristic scheme for scheduling cascaded blocked-out continuous processing units with simultaneous arrivals of the raw materials and multiple upliftment due dates. The scheme has been found to work well under varying input situations. However, the planners may not feel very comfortable with the 'optimal' solutions being obtained from the suggested algorithm. The scheme, in its present state, solves a simplified version of the original problem since it does not include changeover cost in its objective function. There is no assurance that the schedule suggested by the scheme would still remain optimal when changeover cost is factored in. As mentioned earlier we had to leave out changeover cost from this version of the algorithm since, given a partial schedule, we are yet to come up with a 'good' admissible estimate for the changeover cost over the remaining scheduling horizon. Work is currently in progress in this regard. Once this heuristic is ready, we expect the complete DFBB scheme to be able to give optimal solutions for most of the real life problems in reasonable time.

REFERENCE:

- Alle, A and Pinto, J. M., "A mathematical programming approach for cyclic production and cleaning scheduling of multistage continuous plants," *Computers and Chemical Engineering*, 28(2004), 3–15.
- Blömer, F. and Günther, H.O., "Scheduling of a Multi-Product Batch Process in the Chemical Industry," *Computers in Industry*, 36(1998), 245-259.
- Frauentorfer, K. and Königspurger, E., "Concepts for improving scheduling decisions: An application in the chemical industry," *International Journal of Production Economics*, 46-47, (1996), 27-38
- Garey, M. R. and Johnson, D. S., "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H Freeman and Company, (1979).
- Graham, R.L., Lawler, E.L., Lenstra, J.K. and Kan, A.H.G.R., "Optimization and approximation in deterministic sequencing and scheduling: A survey", *Ann. Discrete Mathematics*-5(1979), 287-326.
- Grossmann I.E., Quesada J., Raman R. and Voudouris V., "Mixed integer optimization techniques for the design and scheduling of batch processes," *Batch Processing Systems Engineering*, eds. G.V. Reklaitis, A.K. Sunol, D.W.T. Rippin and O. Hortacsu, Springer, Berlin(1996), 451-494.
- Ierapetritou, M.G. and Floudas, C.A., "Effective Continuous-Time Formulation for Short-Term Scheduling.-2. Continuous and Semicontinuous Processes," *Ind. Eng. Chem. Res.* , 37, (1998), 4360-4374.
- Ierapetritou, M.G. Hene, T.S. and Floudas, C.A., "Effective Continuous-Time Formulation for Short-Term Scheduling.-3. Multiple Intermediate Due Dates," *Ind. Eng. Chem. Res.* , 38, (1999), 3446-3461.
- Jain, V. and Grossmann, I.E., "Cyclic scheduling of continuous parallel-process units with decaying performance," *AIChE Journal*, 44(1998), 1623.
- Jia, Z. and Ierapetritou, M.G., "Efficient short term scheduling of refinery operations based on continuous time formulation," *Computers and Chemical Engineering*, 28(2004), 1001–1019.
- Kondili, E., C.C. Pantelides and R.W.H. Sargent, "A General Algorithm for Short-Term Scheduling of Batch Operations - 1. Mixed Integer Linear Programming Formulation," *Computers and Chemical Engineering*, 17,(1993), 211-227
- Kudva, Elkamel, Pekny and Reklaitis, G.V., "Heuristic Algorithm for Scheduling Batch and Semicontinuous Plants with Production Deadlines, Intermediate Storage Limitations and Equipment Changeover Costs," *Computers and Chemical Engineering*, 18,(1994), 859-875.
- Ku and Karimi, "Completion Time Algorithms for Serial Multiproduct Batch Processes with Shared Storage," *Computers and Chemical Engineering*, 14, (1990), 49-69
- Ku and Karimi, "Scheduling algorithm for serial multiproduct batch processes with tardiness penalties," *Computers and Chemical Engineering*, 15,(1991), 283-286
- Mendez, C. and Cerda, J., "An MILP continuous-time formulation for short-term scheduling of multiproduct continuous facilities," *Computers and Chemical Engineering*, 26,(2002), 687-695
- Pinto, J.M. and Grossmann, I.E., "Assignment and Sequencing models for the scheduling of process systems," *Annals of Operations Research*, 81(1998), 433-466.
- Rippin, D.W.T., "Batch process systems engineering: a retrospective and prospective review," *Computers and Chemical Engineering*, 17 Suppl.(1993), S1 – S13.
- Reklaitis, G.V., "Overview of the Planning and Scheduling technologies," *Latin American Applied Research*(2000)

- Wiede and Reklaitis, G.V., "Determination of completion times for serial multiproduct processes-2. A multiunit finite intermediate storage system," *Computers and Chemical Engineering*, 11,(1987a),345-356
- Wiede and Reklaitis, G.V., "Determination of completion times for serial multiproduct processes-3. Mixed intermediate storage systems," *Computers and Chemical Engineering*,11,(1987b),357-368
- Sargent R.W.H., "Introduction:25 years of progress in process systems engineering," *Computers and Chemical Engineering*,28,(2004),437-439
- Schilling, G. and Pantelides, C.C., "A simple continuous time process scheduling formulation and a novel solution algorithm," *Computers and Chemical Engineering*, 20,(1996),S1221-S1226
- Shah, N., "Single-and multisite planning and scheduling: current status and future challenges," *AIChE Symposium Series*(1998).
- Shah, N., Pantelides, C.C. and Sargent, R.W.H., "A General Algorithm for Short-Term Scheduling of Batch Operations-2. Computational Issues," *Computers and Chemical Engineering*,17,(1993),229-244
- Zhang, X. ad Sargent, R.W.H., "The optimal operation of mixed production facilities-A general formulation and some approaches for the solution," *Computers and Chemical Engineering*, 20,(1996) 897-904.