A Hybrid Formulation for Multipurpose Batch Process Scheduling

Benjamin Roe1, Lazaros G. Papageorgiou2 and Nilay Shah1 *

1CPSE, Imperial College, London, England, SW7 2AZ
2CPSE, University College, London, WC1E 7JE

Abstract

This paper presents a hybrid algorithm for optimal scheduling of multipurpose batch process plants. The algorithm decomposes the scheduling problem into two parts: an aggregate scheduling stage, based on Mixed Integer Linear Programming (MILP), and a detailed scheduling stage, applying Constraint Logic Programming (CLP). This decomposition utilises the complementary strengths of the two solution techniques in an attempt to reduce the inherent combinatorial complexity of process scheduling. A summary of the structure of the algorithm is presented and encouraging results for scheduling an example process are shown.

Keywords: scheduling, multipurpose, MILP, CLP, hybrid algorithm

1 Introduction

Multipurpose production plants have many advantages over more traditional, unipurpose plants. The ability to produce multiple products using the same equipment provides greater flexibility in production, increasing capacity utilisation and the ability to respond to changing market conditions. This flexibility comes at the cost, however, of increased complexity in operation. Determining optimal schedules for multipurpose processes is a computationally hard problem; in fact, it is NP-hard. Much research in process scheduling has used MILP formulations to solve the scheduling problem. The formulations of Kondili, Pantelides & Sargent (1993) and Shah, Pantelides & Sargent (1993) schedule a process defined by a State Task Network (STN) over a time horizon divided into a fixed number of equal length periods. Process events are constrained to occur at intervals between time periods and batch processing times are assumed to be constant. Further research (Pantelides 1994, Castro, Barbosa-Povoa & Matos 2001, Ierapetritou & Floudas 1998, Giannelos & Georgiadis 2002) has attempted to reduce the size of the MILP model that must be solved by using a continuous time model, which divides the time horizon into fewer, variable sized time periods. CLP as a scheduling tool has been used extensively in discrete scheduling problems - timetabling and jobshop scheduling, for example. The work

* Author to whom correspondence should be addressed. Email : n.shah@imperial.ac.uk
of Harjunkoski & Grossmann (2002) and Jain & Grossmann (2001) has shown that a hybrid approach, combining MILP and CLP, can be used to efficiently solve complex process scheduling problems.

2 Algorithm

The approach used in this paper decomposes the scheduling problem into two subproblems: an aggregate batch sizing step, followed by a detailed scheduling step. The first is solved using an MILP formulation, while the second is solved using CLP. By using MILP to solve the flow related part of the problem and CLP to solve the sequencing part, the overall algorithm exploits the strengths of both methods. The process is described using a State Task Network (Kondili et al. 1993). An aggregate solution, specifying the number of tasks performed on each unit in the process and the sum of their extents, is generated by the MILP formulation. This aggregate solution is then passed to the CLP portion of the solver, which attempts to generate an optimal schedule. Cuts are then added to the MILP model and the process is repeated until no better solution is found.

2.1 MILP Formulation

The constraints for the MILP formulation can be divided into the constraints which make up the basic model, and additional constraints that improve the efficiency of the solution algorithm.

2.1.1 Basic Constraints

The main constraints in the MILP formulation are the material balances. The nett consumption of each state must be less than the initial amount of the state available. Sufficient amount of final product states must also be produced to satisfy deliveries to customers.

\[
L_s^I + \sum_{i \in I_s} \sum_{j \in U_i} f_{ij}^p E_{ij} - \sum_{i \in I_s} \sum_{j \in U_i} f_{ij}^c E_{ij} = L_s^F \quad \forall s
\]  

(1)

\[
L_s^F \geq d_s \quad \forall s
\]  

(2)

The total extent of each task on a unit is limited by the number of the task performed on that unit and the unit size:

\[
V_{ij}^{min} N_{ij} \leq E_{ij} \leq V_{ij}^{max} N_{ij} \quad \forall i, j \in U_i
\]  

(3)

An estimate of the makespan of the schedule is required in the aggregate model. The time taken for the tasks on a unit is the sum of the duration of the tasks assigned to it and the changeovers required between the tasks. Only modelling sequence-independent changeovers, the minimum possible changeover time can be used as an estimate. The following constraints are required to model the makespan:

\[
W_{fj} \leq \sum_{i \in F_f \cap T_j} N_{ij} \leq MW_{fj} \quad \forall f, j
\]  

(4)

\[
D_j = \sum_{i \in T_j} l_i N_{ij} + \left( \sum_{f \in F^a} W_{fj} - 1 \right) C_j \quad \forall j
\]  

(5)
where $W_{fj}$ is a binary variable indicating whether a task in family $f$ occurs on unit $j$ and $L$ is an estimate of the schedule makespan as the maximum of the utilisation times of every unit, $D_j$. The second sum in constraint 5 is the minimum duration of changeovers required given the tasks to be performed on unit $j$, assuming that all tasks in each family are performed consecutively. The objective function value minimises the cost of the schedule – the value of deliveries made, the makespan cost, unit setup cost and task costs minus the sum of product state values. The cost of the makespan is defined by the multiplier $\beta$. The objective function for the aggregate scheduling stage is therefore:

$$ \beta L + \sum_{i} \sum_{j \in U_i} V_{ij} + \sum_{j} V_j G_j - \left( \sum_{s} V_s L_s^F + \sum_{s} V_s d_s \right) $$

where $G_j$ is a binary variable indicating that unit $j$ is used in the process, constrained thus ($M$ being a sufficiently large integer):

$$ G_j \leq \sum_{i \in T_j} N_{ij} \leq MG_j \quad \forall j $$

### 2.1.2 Additional Constraints

Several classes of constraint can be added to the formulation to improve the accuracy of the aggregate solution and reduce the number of MILP/CLP iterations required. The first type is based on work by Maravelias & Grossmann (2003). It uses knowledge of the process State Task Network (STN) to exclude infeasible solutions. Its basis is to determine the earliest start and latest finish times (EST and LFT) for tasks from their consumption and production of process intermediates and final products. For example, in the STN shown in figure 1, if the initial inventory of state B, then no instance of task TB can begin until at least one instance of task TA has been performed, so the earliest start time of task TB is equal to the duration of task TA. This method can also be applied in reverse for the latest finish time, working backwards from the process final products. The EST and LFT can be used to exclude infeasible solutions from the MILP. The bounds of the $N_{ij}$ variables can be tightened thus:

$$ 0 \leq N_{ij} \leq \frac{LFT_i - EST_i}{l_i} \quad \forall i, j $$

The makespan estimate can also be adding the EST/LFT to the minimum duration of batches on each unit:

$$ D_j = \sum_{i \in T_j} l_i N_{ij} + \left( \sum_{j \in F_a} W_{fj} - 1 \right) C_j $$

$$ + EST_i^{\min, i \in T_j} + H - LFT_i^{\max, i \in T_j} \quad \forall j $$

where $EST_i^{\max, i \in T_j}$ indicates the maximum value of $EST_i$ for $i$ in the set $T_j$. The second cut is used to exclude solutions generated in a previous iteration of the solver from the
MILP solution. The simplest method is to add no-good constraints for a set of values of the $N_{ij}$ variables. This is achieved by writing each $N_{ij}$ variable as the sum of a series of binary variables $n_{ijt}$ and then adding a standard integer cut for a specific combination of these binary variables.

$$N_{ij} = \sum_{t=0..N_{ij}^{\text{max}}} t n_{ijt} \quad \forall i, j \in U_i$$  \hfill (11)

$$N_{ij}^{\text{max}} = \frac{\text{LFT}_i - \text{EST}_i}{l_i}$$  \hfill (12)

$$\sum_t n_{ijt} = 1 \quad \forall i, j$$  \hfill (13)

All solutions in which a set of $N_{ij}$ variables take specific values are therefore removed by adding the constraints:

$$\sum_{i,j,t \in V} n_{ijt} \leq |V| - 1$$  \hfill (14)

where $V$ is a set of triplets specifying the $i$ and $j$ indices and the value $t$ to be excluded.

### 2.2 CLP Formulation

The detailed solution is generated using a CLP formulation, implemented using the ECLiPse (Wallace, Novello & Schimpf 1997) solver. The aggregate solution from the previous section assigns tasks to units and decides their total extent; the CLP formulation then assigns a start time, $T^S_n$, and individual extent, $S_n$, to each batch. The bounds of these variables are:

$$T^S_n :: [\text{EST}_{i_n}..\text{LST}_{i_n}]$$  \hfill (15)

$$S_n :: [0.0..E_{i_n,j_n}^{\text{max}}]$$  \hfill (16)

$$L \geq T^E_n \quad \forall n$$  \hfill (17)

where $i_n$ and $j_n$ are the tasks and unit batch $n$ is assigned respectively and $E_{i_n,j_n}^{\text{max}}$ is the maximum possible extent of task $i$ on unit $j$. The basic process constraints - non-preemption and changeovers - are expressed using built-in ECLiPse constraints:

$$\text{disjunctive}([T^a_n, [l_{i_n}]], \forall j)$$  \hfill (18)

$$\text{disjunctive} \ (\forall i, j, n' \in (B^j_i \cap B^i_j))$$  \hfill (19)

The sum of the batch sizes $S_n$ is constrained to be equal to the appropriate $E_{ij}$ from the MILP solution:

$$\sum_{n \in (B^j_i \cap B^i_j)} S_n = E_{ij} \quad \forall i, j \in U_i$$  \hfill (20)
Material balance constraints are implemented by adding constraints on nett production of each state before each time point in the horizon. A series of reified constraints are added to the model:

\[ P_{pn}^{S} = \begin{cases} 0 \implies T_{n}^{s} \leq pd^{p} & \forall n, p = 1..H \\ 1 \implies T_{n}^{s} > pd^{p} & \forall n, p = 1..H \end{cases} \]  

(21)

\[ P_{pn}^{E} = \begin{cases} 0 \implies T_{n}^{s} + D_{n} \leq pd^{p} & \forall n, p = 1..H \\ 1 \implies T_{n}^{s} + D_{n} > pd^{p} & \forall n, p = 1..H \end{cases} \]  

(22)

Where \( d^{p} \) is the duration of one time period in the horizon. The nett production of each state can therefore be constrained thus:

\[ \sum_{n} (f_{s1}^{p} E_{n} P_{pn}^{S}) - \sum_{n} (f_{s1}^{c} E_{n} P_{pn}^{E}) \leq (L_{s}^{M} - L_{s}^{I}) \quad \forall s \]  

(23)

where the right-hand term is the difference between the maximum and initial inventory levels of the state. In order to remove degenerate solutions, an ordering is imposed between identical batches on each unit through a series of pairwise constraints of the form:

\[ T_{n}^{E} \leq T_{n}^{S} \]  

(24)

The objective function value for the CLP portion of the solver is identical to the MILP. The unit setup cost, tasks costs and final state inventory terms have been fixed by the MILP solution, but the delivery values and makespan cost can be changed by the CLP solution. Deliveries of product states during the problem horizon are modelled as zero-duration batches that consume the state to be delivered. The value of \( d_{s} \) in aggregate solution is the upper bound of the sum of the extents of these delivery batches. The objective function for the CLP formulation is therefore:

\[ \sum_{s} V_{s} C_{s}^{F} + \sum_{i} \sum_{j \in U_{i}} V_{ij}^{i} + \sum_{j} V_{j}^{i} G_{j} + \sum_{d} V_{d}^{D} S_{n} + \beta L \]  

(25)

### 3 Results

In order to determine the efficiency of this hybrid approach, it was compared to the pure MILP formulation of Shah et al. (1993). The solution times given are for the solver running on an PC with an Athlon-XP 1800 processor, using XPRESS-MP 13.26 on Gentoo Linux 1.4. The process STN is shown in figure 2. The objective is to minimise the schedule makespan while producing 10/5/5/5 units respectively of the 4 product states. The hybrid approach used in this paper finds the optimal solution of a 15 hour makespan in 10 seconds, compared to a solution time of 2500 seconds for the MILP formulation.

### 4 Conclusion

The results in this paper show that a hybrid CLP/MILP scheduling algorithm can greatly outperform a pure MILP formulation for a range of processes. The ability of CLP to handle complex disjunctive constraints and the more intelligent search methods mean that CLP is well suited to solving multipurpose scheduling problems. Areas for further research in the area include improvements to the efficiency of the CLP material balance constraints, tighter cuts for the MILP formulation to reduce the number of CLP/MILP iterations required and further work on the multiperiod MILP formulation.
Figure 2: State Task Network for Example 1

References


Acknowledgements

The ESPRC and the Centre for Process Systems Engineering, Imperial College, London are gratefully acknowledged for financial support for this work. We would also like to thank Mark Wallace and IC-PARC, Imperial College, London for their suggestions regarding this work.

Nomenclature

<table>
<thead>
<tr>
<th>MILP Formulation Symbols</th>
<th>CLP Formulation Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Description</td>
<td>Variable Description</td>
</tr>
<tr>
<td>$L_i^s$</td>
<td>Duration task $i$ on machine $j$</td>
</tr>
<tr>
<td>$L_i^f$</td>
<td>Duration task $i$ on machine $j$</td>
</tr>
<tr>
<td>$U_i$</td>
<td>Number of task $i$ on unit $j$</td>
</tr>
<tr>
<td>$T_j$</td>
<td>Total demand product state $s$</td>
</tr>
<tr>
<td>$I_s$</td>
<td>Unit which batch $n$ is allocated</td>
</tr>
<tr>
<td>Parameter Description</td>
<td>Variable Description</td>
</tr>
<tr>
<td>$N_{ij}$</td>
<td>Total extent of task $i$ on unit $j$</td>
</tr>
<tr>
<td>$T_{S,max}$</td>
<td>Maximum value of $T_{n}$</td>
</tr>
<tr>
<td>$V_{min}$</td>
<td>Minimum capacity of unit $j$</td>
</tr>
<tr>
<td>$V_{max}$</td>
<td>Minimum value of $T_{n}$</td>
</tr>
<tr>
<td>$f_{p,n}^{i}$</td>
<td>Task of batch $n$</td>
</tr>
<tr>
<td>$f_{c,n}^{j}$</td>
<td>Start time of batch $n$</td>
</tr>
<tr>
<td>$f_{p,n}^{j}$</td>
<td>Extent of batch $n$</td>
</tr>
<tr>
<td>$f_{c,n}^{j}$</td>
<td>Extent of batch $n$</td>
</tr>
</tbody>
</table>