Pergamon

Computers chem. Engng, Vol. 21, Suppl., pp. S1061-S1066, 1997

Printed in Great Britain

PII:S0098-1354(97)00190-7

0098-1354/97 $17.00+0.00

RTN-based Rolling Horizon Algorithms for Medium Term Scheduling of Multipurpose Plants

A.D. Dimitriadis, N. Shah* and C.C. Pantelides

Centre for Process Systems Engineering
Imperial College of Science, Technology and Medicine
London SW7 2BY, U.K.

Abstract. One of the major advances in the scheduling of multipurpose plants over the past decade has been the increasing usage of rigorous mathematical programming approaches based on increasingly general problem representations such as the Resource-Task Network (RTN) framework recently proposed by Pantelides (1994). These approaches often require the solution of large optimisation problems, the size of which increases with the length of the time horizon being considered. For medium-term scheduling problems typically considering time scales ranging from one month to a year, the solution via the direct application of the mathematical programming techniques mentioned above is very often impossible with the currently available computer hardware and solution methods. This paper presents three "rolling horizon" algorithms that are formally based on the rigorous aggregated RTN formulation presented by Wilkinson (1996).

INTRODUCTION

Modern multipurpose process plants are able to produce a wide range of different products using a variety of production routes. This characteristic makes such plants particularly effective for the manufacture of classes of products that exhibit a large degree of diversity and which are subject to fast-varying customer demands.

Given such flexibility, the efficient scheduling of such plants becomes a problem of crucial importance. Specifically, scheduling is concerned with finding the best (*e.g.* minimum cost or maximum profit) way of assigning the production resources to tasks over the time horizon of interest in order to meet the required production objectives.

The use of mathematical programming provides efficient ways of formulating and solving scheduling problems. Recently, Pantelides (1994) presented the concept of Resource-Task Network (RTN), as a unified framework for the representation and solution of a wide range of scheduling problems. Despite continuing research into scheduling for formulations based on continuous time representations, much of the successful applications of mathematical programming techniques to date have been achieved using formulations based on a uniform, discrete representation of time such that all events occur on time interval boundaries. Overall, this type of approach results in a mixed integer linear programming problem (MILP).

Discrete time formulations have proved to be very effective for short-term scheduling. However, when the time horizon of interest increases, the resulting MILP becomes too large to solve in a reasonable amount of time. Consequently, in order to produce medium-term schedules, one needs to decompose the scheduling problem into a number of smaller, easier to handle, subproblems.

Recently, Subrahmanyam *et al.* (1996) proposed a hierarchical algorithm that decomposes the overall scheduling problem into an upper level "Planning Super problem" which handles aggregate planning decisions and a lower level which verifies the feasibility of these decisions through detailed scheduling. Also, Zentner *et al.* (1996) suggested that only the first portion of a production schedule needs to be of sufficient detail. Thus, a multiple time discretisation scheme was introduced, where a coarser time discretisation is used in the later parts of the planning horizon.

"Rolling horizon" algorithms are based on separating the scheduling problem in a sequence of iterations, each of which models only part of the planning horizon in detail ("the detailed time block"), while the rest of the horizon ("the aggregate time block") is represented in an aggregate manner. In principle, this approach may produce close to optimal solutions with a significant reduction of the computational requirements. However, its effectiveness crucially depends on the accuracy of the aggregate representation used.

This paper starts with a brief summary of the RTN formulation and its aggregate representation proposed by Wilkinson (1996). Two "Rolling Horizon" strategies will be proposed, differing in the relative positions of the detailed and aggregate time blocks

* Author to whom all correspondence should be addressed, Email: n.shah@ic.ac.uk Fax: (44)-171-5946606

(TBs). A medium-term example will be used in order to demonstrate them and compare their efficiency. Finally, potential applications of this approach are outlined and comparisons with earlier work in this area are drawn.

RTN-BASED FORMULATION AND ITS AGGREGATE REPRESENTATION

The key property of the RTN formulation is the uniform description of all resources involved in the scheduling problem. There is no distinction between, for example, raw materials, equipment items and utilities, all of which are described by the same fundamental constraints.

In addition to the fundamental RTN concepts, Pantelides (1994) also presented a simple mathematical formulation of the problem of scheduling a process described by RTN. The formulation is based on a uniform representation of time and involves only three types of constraint (see appendix A). The operation of the plant is modelled in terms of discrete and continuous variables describing the timing and nature of all tasks taking place over the time horizon. More specifically, the discrete "task extent" variables N_{kt} characterise the number of instances of a task k that start at a time t. On the other hand, the continuous task extent variables ξ_{kt} describe the total amount of material that is processed by these instances.

Wilkinson (1996) proposed a family of aggregate formulations based on the RTN concept and derived from Pantelides' (1994) detailed scheduling formulation using a set of formal mathematical manipulations called "aggregation operators". The basic idea is to separate the planning horizon into *Aggregate Time Periods* (ATPs). Each ATP is described by aggregate variables, each of which is equivalent to the weighted sum of the corresponding exact variables over the time intervals spanned by that ATP.

In general, a p^{th} order *aggregation operator* $A^p[\,]$ is defined as following:

$$A^p[C_t(.)] \equiv \sum_{t'=t-h+1}^{t} (t-t'+1)^p C_{t'}(.) \qquad (1)$$

where h is the size of each ATP. The application of this operator to the constraints $C_t(.)$ of the detailed formulation results in the p^{th} order aggregate formulation.

It should be noted that the aggregate formulation does not eliminate *all* variables of the detailed formulation. In particular, variables that represent activities which start near the end of one ATP and continue into the next are retained as *linking variables* in the aggregate formulation.

In contrast to many *ad hoc* aggregate scheduling formulations presented in the past, the formulation proposed by Wilkinson (1996) has two important theoretical properties:

1. First, it reduces to the detailed formulation when the size of the ATPs reduces to 1.

2. Secondly, it is a strict relaxation of the detailed formulation.

From the practical point of view, this aggregate formulation has been shown to produce an accurate assessment of the production capacity of multipurpose plants (see Wilkinson (1996)).

FORWARD ROLLING HORIZON ALGORITHMS (FRH)

The key idea of this iterative approach is described bellow:

1. The planning horizon is separated into two Time Blocks (TBs). The first (*detailed*) TB is modelled using the discrete time RTN formulation (Pantelides (1994)). The second (*aggregate*) TB is modelled using the aggregate RTN formulation (Wilkinson (1996)).

   At the first iteration, the detailed TB spans a relatively small part of the entire planning horizon.

2. The resulting MILP is solved to optimality. If the detailed TB already covers the entire time horizon, then the algorithm stops.

3. Once the optimal solution is obtained, some of the variables of the detailed TB are fixed to their current optimal values for all subsequent iterations.

4. The size of the detailed TB is increased by a number of time periods and the size of the aggregate TB is decreased by an equal amount.

5. Repeat from Step (2).

An important decision in the above algorithm concerns the set of variables to be fixed at step 3. One possibility is to fix *all* variables in the detailed TB. However, this may be unnecessarily restrictive: the only aim of fixing variables is to reduce the computational complexity of the MILP solved at step 2 in subsequent iterations. This complexity is primarily a function of the number of discrete decisions. Hence, it may be sufficient to fix only these decisions while allowing the corresponding continuous variables to be free to vary. This has the advantage of allowing each iteration of the algorithm to re-assess any continuous decisions made at earlier iterations, potentially compensating for any inaccuracies caused by the use of the aggregate formulation.

In any case, when the algorithm terminates, we have a *detailed* schedule for the *entire* time horizon of interest.

This scheme is outlined in Figure 1 for a 4 week long scheduling problem. At each iteration, the size of the detailed TB is increased by 1 week.

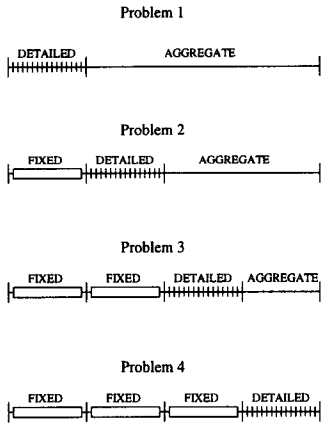

Figure 1: Forward rolling horizon algorithm

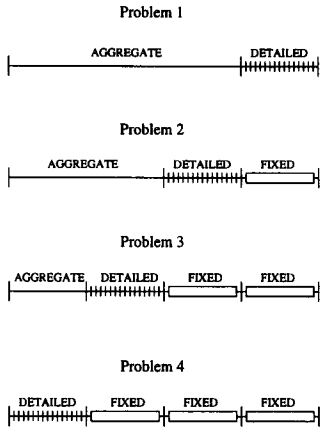

Figure 2: Backward rolling horizon algorithm

BACKWARD ROLLING HORIZON ALGORITHM (BRH)

In the FRH algorithm, the detailed TB precedes the aggregate TB. This may be justified on the basis that the first part of the planning horizon is more important as it is the one derived from data of least uncertainty.

We propose an alternative approach where the aggregate TB precedes (rather than follows) the detailed TB. This approach leads to a *Backward Rolling Horizon (BRH) algorithm.*

A somewhat related approach was used by Shah and Pantelides (1994). They also separate the planning horizon in a number of time blocks considering the scheduling problem separately in each one of them, starting from the last one. However, they do not include any aggregate or other information regarding earlier parts of the planning horizon.

The progress of the iterations is generally analogous to that in the FRH algorithm, with the detailed TB growing backwards (instead of forwards) at the expense of the aggregate one. This scheme is outlined in figure 2 for a 4 week long scheduling problem. At each iteration, the size of the detailed TB is increased by 1 week.

The aggregate nature of the first TB may cause overestimations in some of the continuous variables (product deliveries, excess resource levels,*etc*) in the subsequent detailed TB. These variables have to be reassessed in all iterations, thus, we select to fix *only* the integer variables of the detailed TB at step 3 of each iteration.

DEALING WITH INFEASIBILITIES

It should be emphasised that there is no theoretical guarantee of the optimality of the solutions produced by either of the three Rolling Horizon methods described above. Furthermore, the fixing of some variables at step 3 of each iteration may cause the MILPs solved at step 2 of later iterations to be infeasible, due to an inability to fulfill the necessary production requirements.

In order to avoid such infeasibilities, we treat all product demands as "soft" production targets rather than hard constraints. Fortunately, this is often the case in practice when dealing with medium or long term scheduling.

A less obvious source of infeasibilities is a problem arising at the boundary between the aggregate and the detailed TBs. Specifically, it is possible at a certain iteration for tasks to start near the end of a detailed TB and end near the beginning of the following aggregate TB. In this case, and due to the aggregate representation of the later behaviour of the plant, the continuous extents of such tasks may be overestimated in one iteration. This, in turn, may lead to infeasibilities at the next iteration when both neighbouring TBs are solved in detail.

Such infeasibilities can be avoided by expanding the detailed TB in each iteration by $\tau_{max} - 1$ discrete time periods where τ_{max} is the size of the longest task of the formulation . This measure ensures that any task that starts in the last time periods of the detailed TB complies with the strict detailed constraints and not just the looser aggregate ones.

However, when fixing *only* the integer variables of the detailed TB at step 3 of each iteration, the algorithm can revise the values of the continuous variables and avoid these cases of infeasibilities without needing to include any additional constraints.

EXAMPLE

We consider a simple blending and packing process, which is typical of many fast moving consumer goods (FMCG) manufacturing plants. The process is sufficiently simple to enable us to solve the detailed scheduling problem over the entire time horizon of interest.[1] This permits comparison with the solutions obtained using the RH strategies. The recipe, resources, and problem to be solved are described below.

[1]Of course, the RH approaches are particularly useful where it is *not* possible to solve the fully detailed problem in the horizon of interest.

	Tonnes required at end of Week:		
Product	2	4	6
Prod1	200 to 250	300 to 400	200 to 300
Prod2	200	400 to 600	300

Table 1: Desired delivery profile

Recipe

Two ingredients, FeedA and FeedB are blended and the resulting product UnProd is packaged to give either the 1 *kg* pack-size Prod1, or the 2 *kg* pack-size Prod2 as the final products. The recipe is summarised below:

1. *Blending:* Blend FeedA (50 %) with FeedB (50 %) to produce UnProd after 2 hours. This task requires the services of an operator.

2. *Packing1:* Use the continuous packaging line to package UnProd and produce Prod1 after 1 hour.

3. *Packing2:* Use the continuous packaging line to package UnProd and produce Prod2 after 1 hour.

4. *Retooling:* Re-tool the packaging line whenever it is to switch from one packing task to the other. This task takes 3 hours to complete and also requires the services of an operator.

Available Resources

The following resources are available:

1. Blenders *Blender1* (5 te) and *Blender2* (5 te) suitable for *Blending*.

2. A continuous packaging line *Pline* with a a capacity of 3 te/hr for *Packing 1* and 6 te/hr for *Packing 2*.

3. Unlimited storage for all feeds and products.

4. Storage tank *Tank1* (5 te) for UnProd.

5. Two operators available for *Blending* and *Retooling* tasks.

The scheduling problem

The plant operation is to be scheduled over six weeks of 5 working days each, the scheduling objective being the maximisation of the total value of deliveries. Deliveries are to be made at the end of every second week. Each of these deliveries is characterised by minimum and maximum amounts (see table 1). The unit values of Prod1 and Prod 2 are 4 and 3 cost units per tonne.

Iteration No	1	2	3
Detailed TB (weeks)	1 - 2	1 - 4	1 - 6
Aggregated TB	3 - 6	5 - 6	-
Fixed Cont. Space	-	1 -2	1 - 4
Fixed Int. Space	-	1 - 2	1 - 4
Optimal Objective	9240	8788	8730
Fully Relaxed Obj.	9300	9244	8790
No. Variables	7382	14582	21606
No. Int. Variables	1481	1481	1440
No. Constraints	14754	29154	43206
No. B & B nodes	629	582	435
CPU secs	2420	2102	2154

Table 2: Results of the FRH algorithm (Fixing all variables of the detailed TB)

Results

All solutions are obtained on a SUN Ultra workstation, using the OSL solver with a 5 % optimality margin. The maximum number of Branch & Bound (B & B) nodes allowed to bé searched is 10000. The time discretisation used for the detailed scheduling formulation was 1 hour. This results in 720 time periods in the time horizon considered.

Forward Rolling Horizon Algorithm

The detailed TB is initially set to 2 weeks and is increased by 2 weeks after each iteration. Thus, the problem is solved in 3 iterations. The results for each iteration are shown in table 2 for the case of fixing *all* variables of the detailed TB at step 3 of the algorithm, and in table 3 for the case of fixing *only* the integer variables.

An interesting point is the results obtained in the first iterations of both cases. As no variables have been fixed yet, both sub-problems are relaxations to the fully detailed problem and so, both solutions obtained are valid lower cutoffs to the fully detailed solution. Note that the two sub-problems are slightly different due to the presence of the additional constraints for preventing infeasibilities (see above) in the case of fixing all variables.

We also see that the optimal objective tends to decrease as iterations pass. As the size of the Aggregate TB decreases, the algorithm tends to overestimate in a lesser extent the production capacity of the plant. Actually, this is the case in all Rolling Horizon algorithms presented in this work.

A comparison of the two fixing strategies shows that when fixing only the integer variables, the algorithm remains more flexible during the iterative procedure and produces a better final solution.

Backward Rolling Horizon Algorithm

Similarly, the detailed TB is initially set to 2 weeks and is increased backwards by 2 weeks after each iteration. Thus, the problem is again solved in 3 iterations. The results in each iteration are shown in

Iteration No	1	2	3
Detailed TB (weeks)	1 - 2	1 - 4	1 - 6
Aggregated TB	3 - 6	5 - 6	-
Fixed Cont. Space	-	-	-
Fixed Int. Space	-	1 - 2	1 - 4
Optimal Objective	9300	8860	8856
Fully Relaxed Obj.	9300	9300	8860
No. Variables	7292	14492	21606
No. Int. Variables	1463	1463	1440
No. Constraints	14574	28974	43206
No. B & B nodes	577	3157	437
CPU secs	1912	7847	2216

Table 3: Results of the FRH algorithm (Fixing only the integer variables of the detailed TB)

Iteration No	1	2	3
Detailed TB (weeks)	1 - 2	1 - 4	1 - 6
Aggregated TB	3 - 6	5 - 6	-
Fixed Cont. Space	-	-	-
Fixed Int. Space	-	1 - 2	1 - 4
Optimal Objective	9300	9133	9112
Fully Relaxed Obj.	9300	9300	9300
No. Variables	7294	14494	21608
No. Int. Variables	1463	1463	1440
No. Constraints	14576	28976	43208
No. B & B nodes	475	507	498
CPU secs	698	2219	2083

Table 4: Results of the BRH algorithm

table 4 (note that we *only* fix the integer variables of the detailed TB at step 3 of the algorithm).

Again, the solution of the first iteration is a valid lower cutoff value to the fully detailed solution.

Detailed Solution

The fully detailed problem is formulated and solved. Then, the best solution obtained by the RH methods (9112) is used as a cutoff value in order to speed up the solving procedure. Both results are shown in table 5.

Cutoff used	-	9112
Optimal Objective	8362	9300
Fully Relaxed Objective	9300	9300
No. nodes examined	>10000	5053
CPU secs	129625	58429
No. Variables	21604	
No. Integer Variables	4320	
No. Constraints	43924	

Table 5: Detailed solution

	Obj.	B&B Nodes	CPU secs
FRH (all)	8730	1646	6676
FRH (int)	8856	4171	11975
BRH (int)	9112	1480	5000
Detailed	8362	>10000	129625
Det (cutoff)	9300	5053	58429

Table 6: Comparison of results

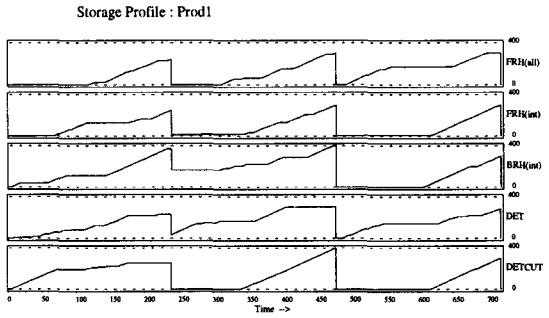

Figure 3: Storage profiles for Prod1

Comparative Results

A comparison of the solutions obtained is shown in table 6. The storage profiles of Prod1 and Prod2 for all solutions obtained are compared in figures 3 and 4.

All Rolling Horizon algorithms provide good approximations of the fully detailed solution with a significant reduction in computational requirements. In fact, within the allowed maximum number of B & B nodes, all Rolling Horizon algorithms provide *better* solutions.

When using the best Rolling Horizon solution (9112) as a cutoff to the fully detailed problem, the optimal detailed solution is found. However, the BRH method provides a solution only 2 % lower than this and it is almost 12 times faster.

CONCLUSIONS

The use of rolling horizon algorithms for the solution of scheduling problems is not new. The main advantage of the algorithms presented here over earlier work is the use of an aggregate formulation to provide a tight estimate of the plant production capacity over the part of the time horizon not being considered in detail at each iteration. Furthermore, the use of the RTN framework ensures that the algorithms are immediately applicable to a very wide range of scheduling problems.

In addition, the aggregate formulation of Wilkinson (1996) has the advantage of being entirely consistent with the RTN formulation used to describe the detailed TB, being derived from it in a rigorous mathematical fashion.

All rolling horizon algorithms presented here provide a fully implementable production plan. Although the optimality of the solutions obtained cannot be guaranteed, example results show that the

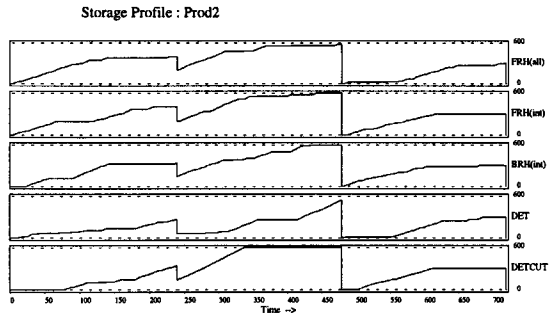

Figure 4: Storage profiles for Prod2

solutions obtained are very good approximations of the detailed solution. Additionally, all algorithms are significantly faster than solving the complete scheduling problem.

Finally, all algorithms provide a *lower bound* on the optimal solution of the complete problem. If it is desired to obtain the latter, this lower bound can be used to initialise the branch and bound solution procedure, thereby decreasing its computational cost.

A Constraint types of the RTN formulation

Excess Resource Balance Constraints

These constraints relate the excess resource levels for each resource type in successive time intervals.

$$R_{rt} = R_{r,t-1} + \sum_{k}\sum_{\theta=0}^{\tau_k}(\mu_{kr\theta}N_{k,t-\theta} + \nu_{kr\theta}\xi_{k,t-\theta}) + \Pi_{rt} \qquad \forall\ r,t \tag{2}$$

where N_{kt} and ξ_{kt} are respectively the fixed and variable extents of task k starting at time t. R_{rt} is the amount of resource r at time t and R_{r0} is the initially available amount of the resource r. Π_{rt} is the amount of resource r made available from external resources at time t (a negative value represents a demand).

Excess Resource Capacity Constraints

These constraints ensure that the amount of excess resource at any time cannot be negative and that it is always less than a predefined maximum value R_r^{max}.

$$0 \le R_{rt} \le R_r^{max} \qquad \forall\ r,t \tag{3}$$

Operational Constraints

These constraints ensure that the batch size of a unit will not exceed predefined minimum (V_{kr}^{min}) and maximum (V_{kr}^{max}) levels.

$$V_{kr}^{min}N_{kt} \le \xi_{kt} \le V_{kr}^{max}N_{kt} \qquad \forall\ k,t,r \in PE_k \tag{4}$$

Finally, the objective function used depends on the type of problem being solved. Typically, it represents the net profit over the planning horizon taking into account utility and storage costs:

$$max\{\sum_{r\in M}\eta_r(R_{rH} - R_{r0}) + \sum_{r\in U\cup M}\sum_{t}C_rR_{rt}\} \tag{5}$$

Here M is the set of material resources, R_{rH} is the amount of excess resource r which is available at the end of the planning horizon and η_r is the unit value of resource r. U is the set of utility resources and C_r is the unit cost associated with having an excess of each resource. C_r is positive for $r \in U$ and negative for $r \in M$, representing the cost of storage for the material resources.

REFERENCES

Pantelides, C. C., Unified frameworks for optimal process planning and scheduling. In Rippin, D. W. T. and J. Hale, editors, *Proc. Second Conf. on Foundations of Computer-Aided Operations*, pages 253–274 (1994).

Shah, N. and C. C. Pantelides, Scheduling of lubricants blending operations, a case study in optimal scheduling. AIChE Annual Meeting, Los Angeles (1994).

Subrahmanyam, S., J. F. Pekny, and G. V. Reklaitis, Decomposition approaches to batch plant design and planning. *Ind. Eng. Chem. Res.*, **35**, 1866 – 1876 (1996).

Wilkinson, S. J., *Aggregate Formulations for Large-Scale Process Scheduling Problems.* PhD thesis, University of London (1996).

Zentner, M. G., J. F. Pekny, G. V. Reklaitis, and J. N. D. Gupta, Practical considerations in using model-based optimization for the scheduling and planning of batch/semicontinuous processes. *J. Proc. Cont.*, **35**, 259 – 280 (1996).