4-2006

Hybridizing Discrete- and Continuous-Time Models For Batch Sizing and Scheduling Problems

Siqun Wang

Monique Guignard
University of Pennsylvania

Hybridizing Discrete- and Continuous-Time Models For Batch Sizing and Scheduling Problems

Abstract
This paper proposes a new hybrid technique called "partial parameter uniformization" (hereafter PPU). The technique simplifies problems by ignoring the different values that certain problem parameters can take, which may facilitate the solution of some hard combinatorial optimization problems. PPU is applied to complex batch sizing and scheduling problems. Some information can be obtained from a discrete-time model in which job durations have been made uniform. This information is then exploited by a more detailed continuous-time model to generate feasible solutions and further improve these solutions. Good, or optimal solutions to the Westenberger and Kallrath Benchmark problems have been obtained in this way, at relatively low computational cost, as have solutions to the newer problems of Blömer and Günther.

Keywords
batch, machine, scheduling, integer programming, hybrid, continuous-time, discrete-time

Disciplines
Programming Languages and Compilers

Hybridizing Discrete- and Continuous-Time Models for Batch Sizing and Scheduling Problems*

Siqun Wang† Monique Guignard‡

Modified July 20, 2003

Abstract

This paper proposes a new hybrid technique called "partial parameter uniformization" to hybrid discrete- and continuous-time models for batch sizing and scheduling problems. The uniformization method simplifies a problem by ignoring the different values of some parameters that the problem can take, which can facilitate the solution of some complex combinatorial optimization problems. Good feasible solutions can then be obtained at relatively low computational cost for the Westenberger and Kallrath Benchmark problems, as well as for the newer problems of Blömer and Günther.

Keyword: batch, machine, scheduling, integer programming, hybrid, continuous-time, discrete-time.

1 Introduction

Various discrete-time and continuous-time MILP (Mixed Integer Linear Programming) formulations have been built and utilized for capacitated batch sizing and scheduling problems in process industries. Initially, discrete-time formulations (with time-indexed variables) were proposed (see, for instance, [Kondili, Pantelides and Sargent (1993)], [Guignard, Spielberg and Yan (1996)], [Yan (1996)], [Blömer and Günther (1998)], [Wang (1998)], and [Guignard and Wang (1998)]). The major advantage of discrete-time models is the capability of considering complex job flows, handling resource constraints and inventory situations. The major problems with this approach, however, are (1) the large size of the MILP model (especially when each job processing time is different) and (2) the approximation of the actual points in time horizon when the events take place. Given the weaknesses of the discrete-time representations, researchers in this field have recently started to look into continuous-time formulations (with continuous variables for time points or time

*Research partially supported by NSF Grant DMI-9900183.

†Singapore Management University, School of Business, sqwang@smu.edu.sg

‡University of Pennsylvania, Wharton School, Dept. of OPIM, guignard@wharton.upenn.edu

of events) based on different continuous-time representations (see, for instance, [Zhang and Sargent (1993)], [Mockus and Reklaitis (1997)], [Ierapetritou and Floudas (1998a,b)], [Wang and Guignard (1999)], [Burkard, Rortuna and Hurkens (2000)], [Lin and Floudas (2001)] and [Wang and Guignard (2001c)]). One may define non-uniform time periods whenever needed as in the slot-based continuous-time representation, or order activities according to their occurrence on each machine and link them with time sequence constraints as in the activity-based continuous-time representation. One such possible approach may view an event as taking place when a production process starts or stops, or both. Another approach may take the view that one needs only to keep track of points in time (or somewhere in an interval if there is flexibility) when some material gets transferred. One common problem with these continuous-time approaches is that the necessary number of time points or events is not known in advance, and it may be quite large.

In spite of all the modeling improvements on batch sizing and scheduling problems, it seems rather unlikely that any exact method using a single model could both produce optimal schedules and prove their optimality rapidly enough to be practical when problem size gets large. Indeed, with demands that are many times the maximum batch size, it is even difficult to provide initial feasible solutions in reasonable computational time by the existing exact models. In order to quickly bring in a feasible solution, one may want to decompose the original problem into several smaller size sub-problems. However, because of the highly connected structure between the tasks and machines in the batch process as we will show in the next section, additional information will be needed to construct a good feasible solution.

In this paper, we propose a new hybrid technique called "partial parameter uniformization" that can facilitate the solution of some complex combinatorial optimization problems. This approach simplifies a problem by ignoring the different values of some parameters that the problem can take. The idea is as follows. The feasible solutions of a problem are often characterized in terms of several dimensions or characteristics: time, space, amounts, etc. In addition, the problem data usually impose certain relationships on the components of a solution. Suppose that the problem would simplify if one ignored certain of its dimensions or characteristics (for instance, suppose that certain complications relative to time or space would disappear), one can then model this simplified problem and try to solve it, preferably optimally. The best solution found, which satisfies all technical constraints of the problem except those relative to the ignored dimension, can then be used to set target values for the original problem. Good feasible solutions can then be obtained by using this uniformization technique, which can be embedded later on in a modular heuristic method. Like some metaheuristics, this heuristic consists of two main stages: the first performing a search for a feasible solution, the second trying to improve it. It is the first stage that introduces the proposed "partial parameter decomposition" to quickly bring in a good starting solution. Finally, if possible, one may want to introduce a third stage that is used to evaluate the quality of the final feasible solution obtained.

In the following sections, we will first describe the basic features of batch sizing and scheduling problems

by showing the Westenberger-Kallrath benchmark problem. We will then discuss the new hybrid method and the *Partial Parameter Uniformization* methodology. We conclude in presenting computational results on benchmark data sets, based on the original Westenberger-Kallrath benchmark problem and its extension by Blömer and Günther (2000).

2 Problem Definition

The problem under study is concerned with short-term capacitated batch sizing and scheduling problems in process industries. Batch processing is a common manufacturing method in the fine chemical and pharmaceutical industries. Distinct from a continuous process which has a constant material flow feed into and withdrawal from each operating unit, a batch process is intermittent and consists of a collection of processing where batches of the various products are produced by scheduling a set of processing tasks or operations like reaction, mixing or distillation on multiple equipments/machines. *A **batch*** is the process of transforming some product or mix of products on a machine into different products, and the amount of the products processed as a single operation/job by that machine is called the ***batch size***. To satisfy each short-term customized demand, production lines of the facility can vary substantially over time, and batches need to be scheduled (batch scheduling) on multi-functional equipments/machines with various batch sizes (batch sizing) during a short time period. Because of the great diversity and variety of batch processing modes, efficient problem formulation and exploitation of problem structure came out as both a challenge and new opportunities for non-linear and integer programming.

One unpublished yet well-known benchmark problem from [Westenberger and Kallrath (1994)] shows a complicated production process with all the typical features that need to be addressed by solution approaches. Figure 1 and Tables 1, 2, 3 &4, given in and summarized from [Westenberger and Kallrath (1994)], depict the motivating example that represents a process containing typical complicating features as follows:

- a complicated flow structure
  - multiple machines (can produce several products);
  - multiple products (can be produced on several machines);
  - multiple production lines (convergent/ divergent/ cyclic) with proportion requirement in production;
- batch mode production:
  - different batch lengths (processing times) for different products on different machines;

Table 1: Description of production units (machine)

Production unit	Number of processing lines (machines)	Number of products per unit	Maximum batch size[kg]	Minimum batch size[kg]
R1	1 (M1)	1 (P11)	10	3
R2	1 (M2)	2 (P21&P22)	20	5
R3	1 (M3)	2 (P31&P32)	10	4
R4	1 (M4)	4 (P41,P42,P43,P44)	10	4
R5	1 (M5)	2 (P51,P52)	10	4
R6	2 (M6,M7)	3 (P61,P62,P63)	7	3
R7	2 (M8,M9)	5 (P71,P72,P73,P74,P75)	12	4

  - different batch sizes on different machines - have upper and (possibly non-zero) lower limits;

- different storage situations
  - some intermediate products have storage tanks with limited capacity;
  - some intermediate products do **not** have storage tank;
- proportion requirement in production
- non-preemptive processes

Broadly speaking, there are demands for several final products, which are produced from some raw material along multiple production lines. Raw material undergoes a series of transformations before becoming final products. Some production lines merge, some others divide with specified proportion requirement. A given machine can only process one job (or operation) at a time, and the output is one or several new products. Some machines are able to process several jobs, and some jobs can be executed on several machines. Batch sizes have upper and (possibly non-zero) lower limits. ***Batch length*** (i.e., the processing time necessary to produce a batch) is independent of batch size, but varies from machine to machine. The processing order for a product line is exactly specified, and some products are allowed to wait between two processes, while others are not. Backlogging is not allowed. In addition, there can be different kinds of objectives as suggested in [Westenberger and Kallrath (1994)]:

1) the minimization of makespan (execution time),
2) the maximization of profit by optimization of product mix,
3) the minimization of investment cost by optimization of stock capacities and production capacities.

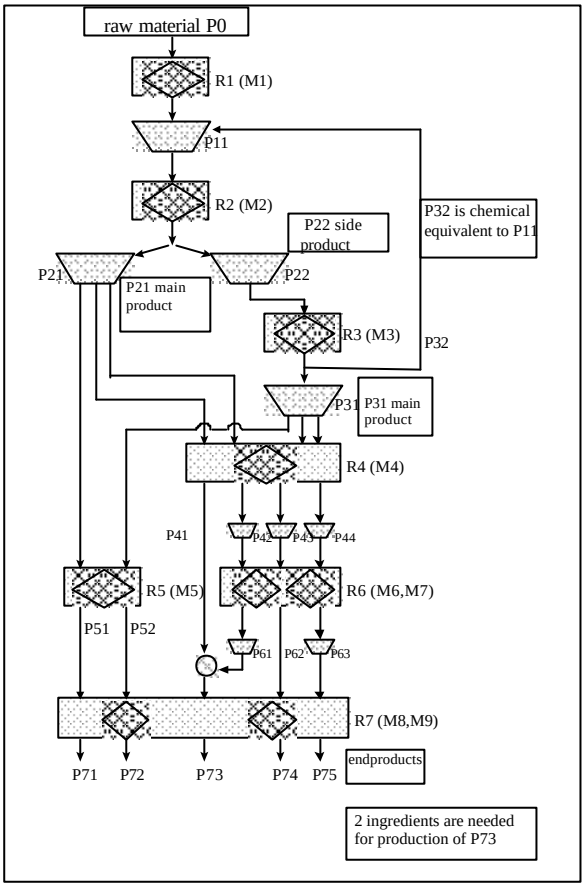

Figure 1: Production assembly with product flow chart

Table 2: Description of batch processes (tasks)

job	Main product (p)	Side product (p)	Fraction of main product $f^{out}(p)$	Production unit (machine)	Processing time per batch	
					in days	in time units (u)
T1	P11	-	1.	M1	0.05	2u
T2	P21	P22	$f^{out}(P21)$	M2	0.1	4u
T3	P31	P32	$f^{out}(P31)$	M3	0.05	2u
T41	P41	-	1.	M4	0.1	4u
T42	P42	-	1.	M4	0.1	4u
T43	P43	-	1.	M4	0.1	4u
T44	P44	-	1.	M4	0.1	4u
T51	P51	-	1.	M5	0.15	6u
T52	P52	-	1.	M5	0.15	6u
T61	P61	-	1.	M6	0.1	4u
T61	P61	-	1.	M7	0.125	5u
T62	P62	-	1.	M6	0.125	5u
T62	P62	-	1.	M7	0.15	6u
T63	P63	-	1.	M6	0.15	6u
T63	P63	-	1.	M7	0.15	6u
T71	P71	-	1.	M8	0.1	4u
T71	P71	-	1.	M9	0.15	6u
T72	P72	-	1.	M8	0.1	4u
T72	P72	-	1.	M9	0.15	6u
T73*	P73	-	1.	M8	0.1	4u
T74	P74	-	1.	M8	0.15	6u
T74	P74	-	1.	M9	0.15	6u
T75	P75	-	1.	M8	0.15	6u
T75	P75	-	1.	M9	0.15	6u

* T73 has products p41 and p61 as ingredient, the fraction of ingredients is 0.5 for both material p41,p61.

** $f^{out}(p21)$ is restricted by $0.2 \leq f^{out}(p21) \leq 0.7$

*** $f^{out}(p31)$ is restricted by several situations for different requirement, here in this paper the restriction is $f^{out}(p31)=0.69$

"u" in the final column is the time unit (0.025 day).

Table 3: Initial stock condition [kg]

Product	Initial stock ($juo0(p)$)	Min. Stock	Max. Stock ($juo1(p)$)
P11	20	0	30
P21	20	0	30
P22	0	0	15
P31	20	0	30
P32	Recycle as P11	-	-
P41	Non-storable	-	-
P42	0	0	10
P43	0	0	10
P44	0	0	10
P51	Non-storable	-	-
P52	Non-storable	-	-
P61	0	0	10
P62	Non-storable	-	-
P63	0	0	10

Table 4: Two instances of product requirement

Product Requirement [kg]	P71	P72	P73	P74	P75
Instance A	30	30	40	20	40
Instance B	0	0	90	50	40

The short-term batch sizing and scheduling problem considered here is similar to, but even more general than, the flexible job shop problem (FJSP), which itself is a generalization of the classical job shop problem (JSP). The FJSP problem [Mastrolilli M. and Gambardella L.M. (1998)] is NP-hard. It schedules a set of jobs on a group of machines. To complete each job, there is a sequence of operations to perform in a given order, and each operation must be processed by one out of a set of machines. The processing times for different operations on different machines might be different. Although the problem considered in this paper has similar characteristics, it is more general, as it also involves batch-sizing, and this requires decisions on how much to produce and how many repetitions of each operation to perform, in order to meet given final demands. The number of repetitions of jobs is not known a priori, it is also a decision choice.

The objective considered in this paper is the minimization of the makespan, i.e., of the completion time of the whole project. Such kind of multi-stage/multi-product batch processing problems are much more complicated than the single-stage ones. The relaxations are either weak or hard to solve, and for large size problems, it is difficult to find good feasible solutions and tight bounds. To our best knowledge, no good solutions has been reported for the two Westenberger-Kallrath benchmark problems (instances A and B of Table 4).

In order to handle realistic size problems, [Blömer and Günther (2000)] suggested a two-stage heuristic solution procedure, which uses a time-grid heuristic to generate initial solutions and a left-shifting heuristic along the time-axis for solution improvement. Numerical results have been reported on 22 new testing data sets, which were constructed for the same process structure of the Westenberger and Kallrath benchmark (Figure 1) with various demand requirements for final products ($P71, P72, P73, P74, P75$). To our best knowledge, neither lower bound estimation has been provided, nor better solutions reported.

In the following sections, we describe the idea of our new hybrid approach and its application to the large batch sizing and scheduling problems of Westenberger and Kallrath, as well as the twenty-two new instances of [Blömer and Günther (2000)]. As we will see, our hybrid approach has better computational performance than either single model methods, or the time-grid heuristic by [Blömer and Günther (2000)].

3 A New Hybrid Method - Partial Parameter Uniformization

Due to the complex structure of the batch production, when demands increase for the Westenberger and Kallrath benchmark problems, it could be very time-consuming to generate a feasible solution by solving a single MILP formulation, even with only a few hundred binary variables. [Blömer and Günther (1998)] developed several heuristics based on different decomposition ideas, but the computational costs are still very high according to their later reports in [Blömer and Günther (2000)] for large size testing data.

We proposes a new hybrid method, which utilizes the discrete- and continuous-time models, by adopting the novel *Partial Parameter Uniformization* strategy. This strategy was inspired by the following observations and thoughts:

a) The non-uniform batch length (i.e., processing time) requirement is one of the main reasons that the problem is difficult to solve. While keeping the other requirements, if we changed all the processing times to the same unit-length), we would get a *simplified* problem which can be solved much more easily by a discrete time model with uniform-length assumption. For example, we can use the discrete-time model[1] of the Appendix, while letting all processing times $l(j,m)$ equal to 1 for each task $j \in J$, on machine $m \in M$.

b) Comparing the formulations between the activity-based continuous-time model with the time-index-based discrete-time model of the Appendix, one sees that the constraints are all very similar to one another except for the ***time*** variables and constraints. For example, if we let all processing times $l(j,m)$ equal to 1 for each task $j \in J$, on machine $m \in M$ in the discrete-time model, and if we ignore the different symbol for the time indices for the two models ("t" in the discrete-time model, "n" in the continuous-time model), it is clear that the variables and constraints are almost the same for the two models, except that the continuous-time model has continuous time variables ($T(p,n)$, $T^s(m,n)$, $T^f(m,n)$) in those time constraints (14), (16), (17), (18), (19), (20), 21), (22) and (23), etc. As such, there is a tight connection between the continuous- and discrete-time models. They both have the same basic structure, which can be utilized to hybridize different models.

Based on these analyses, we propose the *Partial Parameter Uniformization* strategy, which utilizes the basic structure of the batch scheduling problem, but simplifies what makes it hard, the "non-uniform batch length" condition. This strategy turns out to be very effective to decompose the original problem, and becomes the link for different modules on consecutive stages. The basic idea behind the *Partial Parameter Uniformization* strategy is very simple, see Figure 2. It consists of two main steps: 1) by selecting and forcing some of the condition parameters ($\alpha_i, i = 1, 2, ...n$) to be the same unit, the original problem can be transferred into a simpler frame which makes it easier to solve; 2) from the solutions of the new easier problem obtained in the previous step, we can get information which can be "moved back" to the original frame.

Notice that this is somewhat different from "relaxing " some constraint(s). If making a certain set of parameters uniform makes the problem easier to solve, one can "simplify " the problem by assuming that, say, all machines have the same production rate, all customers the same demand, or all arcs the same capacity.

[1]Notice that, since the simplified batch production problem is based on the assumption of identical machine processing time, discrete-time models will be more advantageous in this case than continuous-time models.

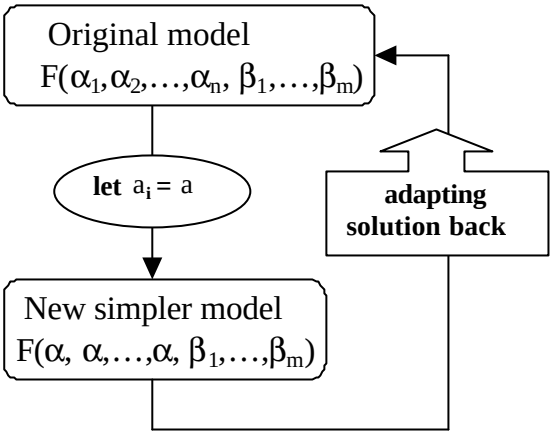

Figure 2: Partial Parameter Uniformization

After solving that " simplified " model, one may be able to find a feasible solution for the original problem by first splitting it into smaller sub-models. In order to truly simplify the problem, each sub-problem should be sufficiently small to be solved easily. The sub-problems are then solved sequentially, and the solution of one problem is used to define some of the complicating variables of the next sub-problem.

For the batch sizing and scheduling problems, all the values of different processing times are the "parameters" we make uniform. Notice that the solution from the discrete time model (with uniform batch length) will be obtained much more easily, and it still satisfies the batch sizing limitation and the production order requirement, because the production structure has not been changed and all the other parameters and requirements are the same as before. Furthermore, the inventory capacity requirements will also be satisfied if we do not consider the time delay for material transfers between machines and storage tanks. Therefore, if one can get a solution from the discrete time model with uniform batch lengths, the sequence of the production schedule can be plugged into the continuous model. With fewer binary variables, we can retrieve all the batch sizing and scheduling information quickly from the continuous time model.

In practice, there are two ways to apply this idea to the batch sizing and scheduling problems. The first one is easier. It directly obtains a sequence of jobs for each machine by solving the discrete-time model with uniform batch length. By limiting these production sequences (i.e., by limiting the binary assignment variables) in the activity-based continuous-time model, one can quickly get a real feasible solution, including all the time schedules as in Figure 3.

The second way of getting a feasible production sequence might get a better feasible solution, but there is a trade-off, as it requires more steps. It is based on a demand decomposition strategy and needs to use the continuous-time model *several times.* Unlike the decomposition heuristics developed in [Blömer and

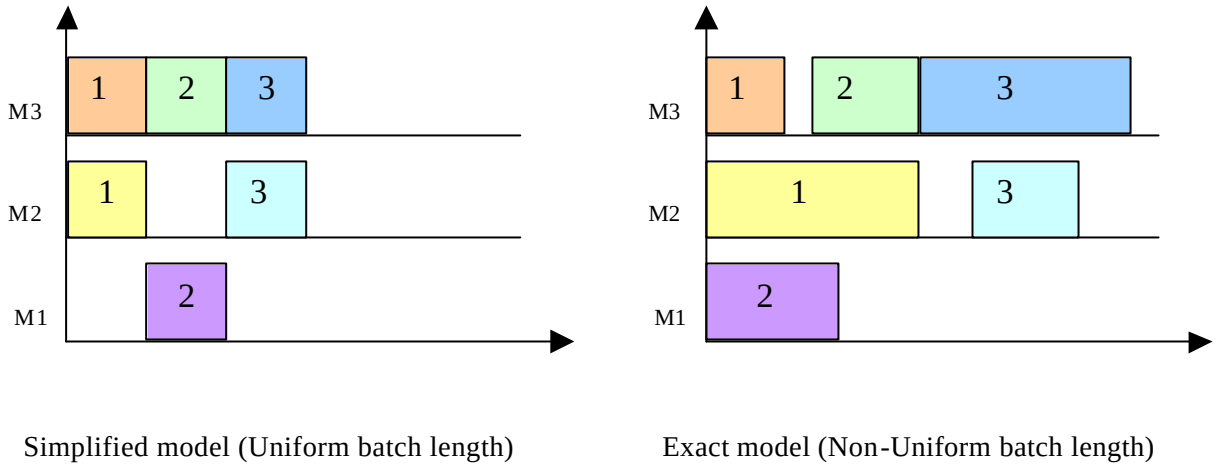

Figure 3: Generating true solution by using production sequence information from the simplified discrete-time model

Günther (1998)], our demand decomposition strategy breaks the original large size problem into smaller ones linked by several time periods. This demand decomposition is actually equivalent to an inventory level decomposition, giving a chronological sequence of feasible inventory-level targets. Figure [4] gives an idea of the procedure, where the I_n's represent inventory levels for various intermediate and final products at various stages.

Through any feasible production path, the inventory level at stage 0 corresponds to the initial inventory condition, and the inventory level at the final stage should meet the final demand. Feasible solutions correspond to feasible paths of the inventory level decomposition. On the other hand, if we have a feasible inventory level decomposition path (generated by the discrete time model) over the time horizon, we can arrange and schedule the batch production to meet the final demand (given the initial condition). The advantage is that we can divide the whole production procedure into several periods, the final stage of each period being the starting stage of the next period. If the demands are not too large, we should be able to obtain very quickly an optimal solution for each production period by an efficient continuous-time model. After solving the production problems for each period, we will have a feasible solution by linking those optimal sub-solutions period by period.

Although it seems to be more complicated, the second method of using demand decomposition is, in fact, more general than the first one, as it can be used to hybrid the uniform-length discrete-time model with any other exact models, e.g., with the slot-based continuous-time model, or even with a non-uniform discrete-time model.

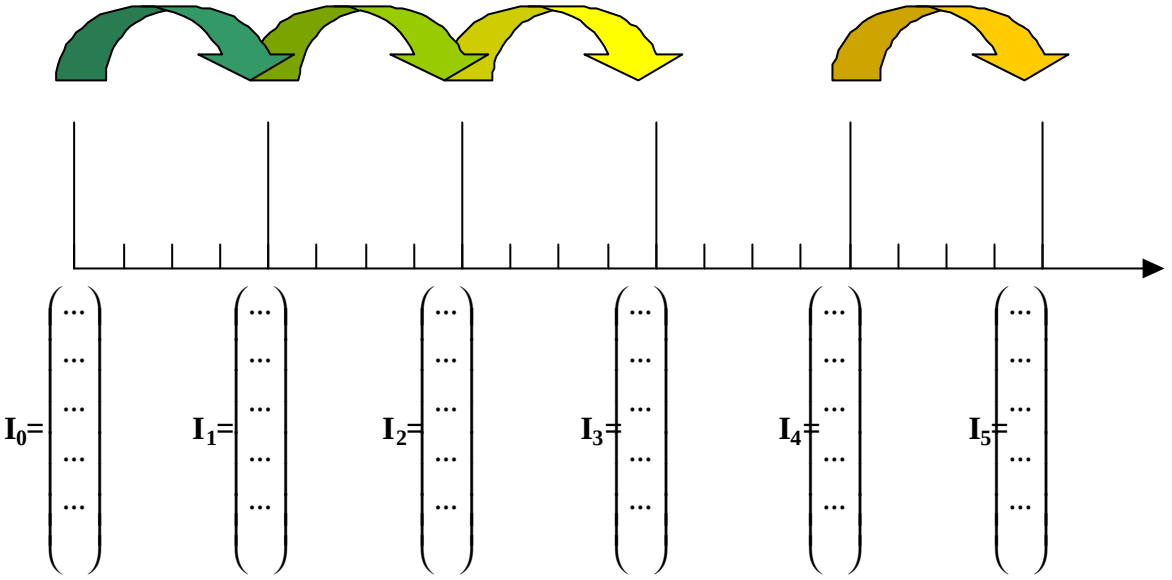

Figure 4: An illustration of the demand decompostion

Once we get a feasible solution, we can then step to a solution-improvement stage in our modular heuristic approach, which will result in a final solution. The detail are described in another paper [Wang and Guignard (2001a)] .

4 Computational Results

The test runs were conducted on an HP-UX/9000 with dual $440MHz$ CPUs. All code is written in GAMS, calling the LP/MIP solver CPLEX 7.5. To evaluate the efficiency, we first apply our modular approach to the 22 test problems given by [Blömer and Günther (2000)]. These 22 test problems, with 22 combinations of various demands for final products ($P71, P72, P73, P74, and\ P75$), use the same structure as the original Westenberger-Kallrath Benchmark (Figure 1). In addition to different demand requirements, the other change to parameters is that the initial stock condition for materials $P11, P21$ and $P31$ is reduced to 10 kg, from the original 20 kg as in Table 3 (the original Westenberger-Kallrath Benchmark). Such reduction of initial inventory levels increases the complexity of the test problems, since, other things being equal, the smaller the initial stock, the more production is needed to satisfy the demand.

For comparison, we put numerical results of the initial solutions from the hybrid approach in Table 5, as well as the numerical results via running the single activity-based continuous-time model. From Table 5, if the final demands are small (for example, instances 1-10), the single model approach by continuous-time

performs better than the hybrid approach. But as demands increase, the hybrid approach is much better than the single model (see instances 011-022), both on CPU time and makespan solutions.

To make further comparison with the grid-based heuristic method by [Blömer and Günther (2000)], we categorize the 22 test problems into 4 groups according to the sums of total demands for the final products. Group 1 is for data sets 1-10, whose total demand for final products is 60. Group 2 is for data sets 11-14, whose total demand for final products is 90. Group 3 is for data sets 15-18, whose total demand for final products is 135. Group 4 is for data sets 19-22, whose total demand for final products is 180.

Group 1		Group 2		Group 3		Group 4	
Test problem	Final Demands	Test problem	Final Demands	Test problem	Final Demands	Test problem	Final Demands
1	20-20-20-0-0	11	10-10-20-20-30	15	15-15-30-30-45	19	20-20-40-40-60
2	20-20-0-20-0	12	30-20-20-10-10	16	45-30-30-15-15	20	60-40-40-20-20
3	20-20-0-0-20	13	10-20-30-20-10	17	15-30-45-30-15	21	20-40-60-40-20
4	20-0-20-20-0	14	18-18-18-18-18	18	27-27-27-27-27	22	36-36-36-36-36
5	20-0-20-0-20						
6	20-0-0-20-20						
7	0-20-20-20-0						
8	0-20-20-0-20						
9	0-20-0-20-20						
10	0-0-20-20-20						

Two different methods of creating initial feasible machine-schedules are compared as in Figure 4: 1) the grid-based heuristic algorithm by [Blömer and Günther (2000)]; 2) our hybrid method with discrete- and continuous-time models. Four groups of data instances are tested, whose average objective values for the initial solutions are shown in Figure 5. For each group, the score on left-hand side is the one obtained from using the grid-based heuristic algorithm by [Blömer and Günther (2000)]. The score on the right-hand side is the one obtained from using our hybrid method - without using the demand decomposition method.

The average CPU time for the initial solutions is also shown in Figure 6. Notice that these reports are created on different type of computers by different solvers. The $2 \times 440MHz$ PA-8500 processor is between 3 and 4 times as fast as a II/266. Therefore, we multiply the CPU time with 5, and put it in Figure 6 as the rightmost score for each group for comparison. Further, if we consider the growth rate of CPU time vs. demand, we can see that, from Figure 7, as demand increases, the CPU time using our hybrid methods increases much slower than that using the grid-based heuristic reported by [Blömer and Günther (2000)].

Furthermore, our modular heuristic which utilizes the new hybrid method performs very well in the original Benchmark examples from [Westenberger, Kallrath, 1994]. For the two Benchmark test problems

Table 5: Initial solution comparison between modular approach and single continuous-time model (without cleaning requirement) by GAMS/Cplex 7.5

Blömer-Günther Problem[a]	Final Demands	Hybrid Approach		Activity Based Continuous-time Model[b]	
		CPU Time for finding initial sol.	Makespan initial sol.	CPU Time for finding initial sol.	Makespan initial sol.
1	20-20-20- 0- 0	2.86	32u	1.65	36u
2	20-20- 0-20- 0	4.81	36u	2.35	34u
3	20-20- 0- 0-20	4.38	34u	2.16	38u
4	20- 0-20-20- 0	4.43	32u	1.89	32u
5	20- 0-20- 0-20	5.34	32u	1.61	32u
6	20- 0- 0-20-20	4.52	39u	4.90	47u
7	0-20-20-20- 0	6.19	36u	4.13	33u
8	0-20-20- 0-20	5.84	37u	2.36	34u
9	0-20- 0-20-20	5.54	44u	5.38	50u
10	0- 0-20-20-20	6.05	44u	4.55	43u
011	10-10-20-20-30	6.87	50u	17.99	58u
012	30-20-20-10-10	8.14	44u	12.43	42u
013	10-20-30-20-10	5.70	44u	374.26	49u
014	18-18-18-18-18	13.47	43u	31.23	50u
015	15-15-30-30-45	77.69	73u	107.71	76u
016	45-30-30-15-15	49.72	62u	133.54	66u
017	15-30-45-30-15	97.03	60u	226.85	64u
018	27-27-27-27-27	49.72	60u	600.24	***
019	20-20-40-40-60	41.09	83u	118.24	90u
020	60-40-40-20-20	50.17	72u	600.30	***
021	20-40-60-40-20	20.50	71u	600.33	***
022	36-36-36-36-36	50.97	76u	316.07	83u

[a] All these problems have initial stock 10[kg] for product $p11, p21, p31$;

[b] System limit for continuous time model: CPU limit =600s; "***" means no feasible solution obtained within the system limits.

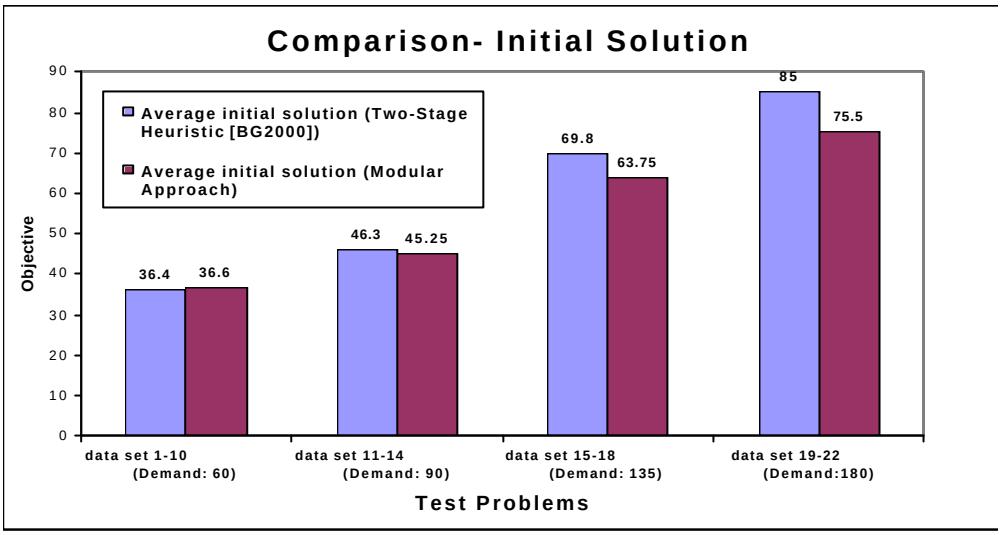

Figure 5: Comparison - initial solution

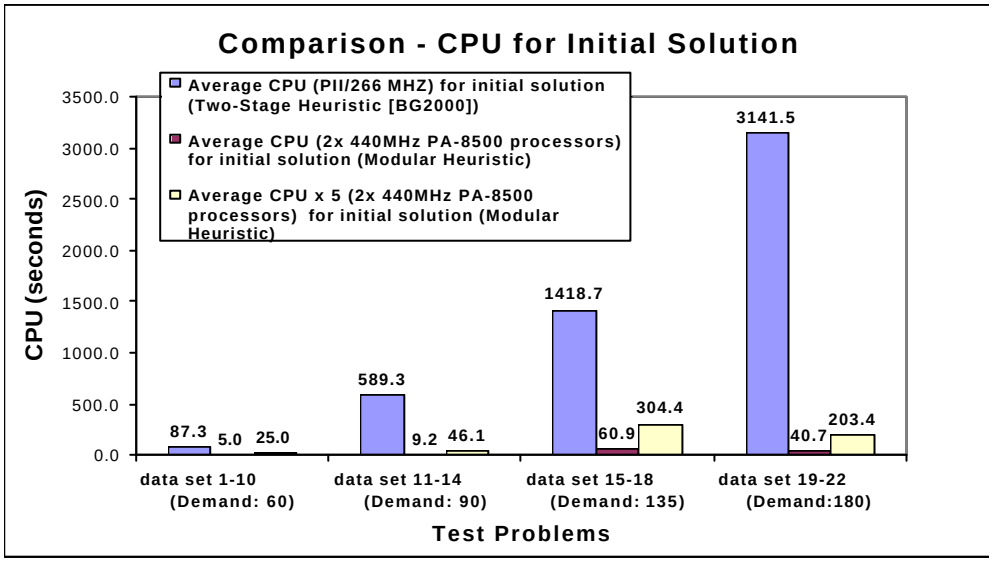

Figure 6: Comparison - CPU for initial solution

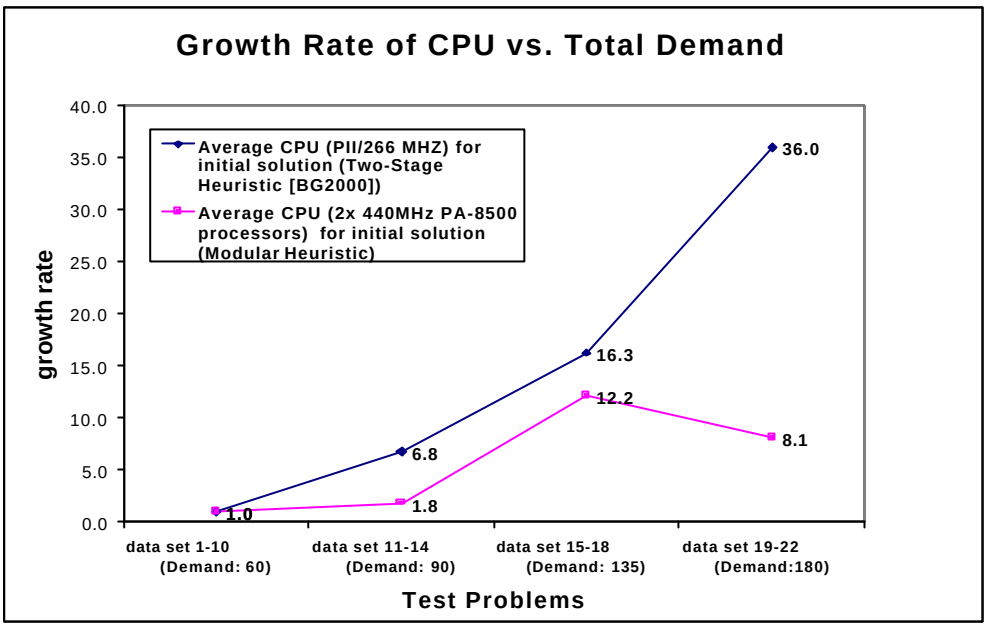

Figure 7: Growth rate of CPU vs. total demand

in Table 6 (i.e., problems A&B), the tightest lower-bounds we can get are 52u and 92u respectively, while the best solutions we can get by using a heuristic are 56u and 92u respectively. The gap is less than 7%.

5 Summary

Discrete-time and continuous-time MILP formulations have been built and utilized in the literature for minimizing makespan in batch sizing and scheduling problems. Yet, good feasible solutions and tight lower-bounds are very difficult to get when demands get larger.

We propose a new partial parameter uniformization strategy, as a simplifying tool for any problem, which introduces related problems with simpler data structures, whose solution is much easier, and, upon inspection, can be exploited and, in some way, used as input information for solving the harder problem. This tool turns out to be very helpful to decompose the original problem, and becomes the link for different modules on consecutive stages. We can now obtain good feasible solutions by using this new integer programming hybrid method, which uses discrete-time and continuous-time models in a specific order, and the results of one model are fed into the next in the chain. Good solutions have been obtained at relatively cheap computational costs for the newer test problems of Blömer and Günther, as well as for the Westenberger and Kallrath Benchmark problems within a 7% optimality gap. In fact, this new hybrid method can also be extended to batch problems with cleaning requirements (see [Wang, 2003]).

Table 6: Numerical results of the modular approach on Westenberger-Kallrath Benchmark (without cleaning requirement) by GAMS/Cplex 7.5

Westenberger-Kallrath Benchmark problems[a]	Final Demand	Modular approach[b] Lower bound[c]	best. obj. (found) Makespan	CPU time for finding best sol.	gap
A	30-30-40-20-40	52u	56u	201.77	7%
B	0-0-90-50-40	92u	92u	685.23	0

[a] All these problems have initial stock 20[kg] for products $p11, p21, p31$;

[b] System limits: for each runs in the improver module, CPU limit = 120s, Node limit =5000; Total CPU limit = 1200s.

[c] Here the lower-bound estimation method will be described in [Wang and Guignard (2001b)] .

References

[1] [Anonymous (1993)] Anonymous "Industry and science put heads together on batch processes". Chemical Week. 152(21): 31. 1993 Jun 2.

[2] [Blömer and Günther (1998)] F. Blömer and H-O. Günther. (1998). "Scheduling of a multi-product batch process in the chemical industry". Computers in Industry. 36(3): 245-259. Jun 1. 1998.

[3] [Blömer and Günther (2000)] F. Blömer and H-O. Günther. (2000) "LP-based heuristics for scheduling chemical batch processes". International Journal of Production Research 35: 1029-1052.

[4] [Burkard, Fortuna and Hurkens (2000)] R.E. Burkard, T. Fortuna and C.A.J. Hurkens, "An event-driven model for chemical batch processes," July 2000, SFB194, Tech. Univ. Graz.

[5] [Cross (1999)]Cross, Jim "Back to the future." Management Review. 88(2): 50-54. 1999 Feb

[6] [Greene (1991)]Greene, Alice H "System Trends in the Batch Process Industries". Production & Inventory Management Review & Apics News. 11(2): 28,31-34. 1991 Feb.

[7] [Guignard, Spielberg and Yan (1996)] M. Guignard, K. Spielberg and H. Yan, "Production planning in batches," INFORMS Meeting, Washington, DC, November 1996.

[8] [Guignard and Wang (2000)] M. Guignard and S. Wang, "An Integer Programming Model for Continuous-Time Batch Processing," OPIM Department Report 00-05-02, University of Pennsylvania, May 2000.

[9] [Ierapetritou and Floudas (1998a)] M.G. Ierapetritou and C.A. Floudas, "Effective Continuous-Time Formulation for Short-Term Scheduling. 1. Multipurpose Batch Processes", Industrial and Engineering Chemistry Research, 37, 1998, 4341-4359.

[10] [Ierapetritou and Floudas (1998b)] M.G. Ierapetritou and C.A. Floudas, "Effective Continuous-Time Formulation for Short-Term Scheduling. 2. Continuous and Semicontinuous Processes", Industrial and Engineering Chemistry Research, 37, 1998, 4360-4374.

[11] [Kondili, Pantelides and Sargent (1993)] E. Kondili, C.C. Pantelides and R.W.H. Sargent, "A general algorithm for short-term scheduling of batch operations-I. MILP formulation," Computers Chem. Engng 17, 1993, 211-227.

[12] [Lin and Floudas (2001)] Lin X. and C.A. Floudas,"Design, Synthesis and Scheduling of Multipurpose Batch Plants via an Effective Continuous-Time Formulation", Computers Chem. Engng, 25, 665-674.

[13] [Luo and Guignard (1997)] Y.-C. Luo, M. Guignard, "A Job Shop Problem in Process Industry," OPIM Department Report 97-08-05, 1997.

[14] Mastrolilli M., Gambardella L.M., (1998) "Effective Neighborhood Functions for the Flexible Job Shop Problem", Journal of Scheduling, Volume 3, Issue 1, 2000. Pages: 3-20.

[15] [Mockus and Reklaitis (1997a)] L. Mockus and G.V. Reklaitis, "Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization," Computers Chem. Engng 21, 1997, 1147-1156.

[16] [Mockus and Reklaitis (1997a)] L. Mockus and G. Reklaitis, "Bayesian Approach to Batch Process Scheduling," Int. Trans. Opl. Res., 4(1) 55-65, 1997.

[17] [Parkinson (1990)] Parkinson, Gerald "Batch Process Control to Adopt Common Guidelines." Chemical Engineering. 97(9): 30-35. 1990 Sep.

[18] [Qian. (2000).] H. Qian. (2000). Oral communication, November.

[19] [Sahinidis and Grossmann (1991)] N.V. Sahinidis and I.E. Grossmann, "Reformulation of multiperiod MILP models for planning and scheduling of chemical processes," Computers Chem. Engng 16, 1991, 255-272.

[20] [Wang (1998)] S. Wang. . "Short Term Scheduling for Chemical Industries," OPIM Department Summer Research Paper, under direction of Professor M. Guignard. August1998.

[21] [Wang (2003)] S. Wang. "New Modeling and Solution Approaches for Combinatorial Optimization, With Application To Exam Timetabling and Batch Scheduling In Chemical Manufacturing," OPIM Department Doctoral Dissertation, under direction of Professor M. Guignard. 2003.

[22] [Wang and Guignard (1999)] S. Wang, M. Guignard," A Continuous Time Model for Short Term Scheduling in Batch Processing," OPIM Department Report 99-11-01, November 1999.

[23] [Wang and Guignard (2000)] S. Wang, M. Guignard," Redefining Event Variables for Efficient Modeling of Semi-Continuous-Time Batch Processing," Annals of Operations Research 116,113-126, 2002.

[24] [Wang and Guignard (2001c)] S. Wang, M. Guignard," An Improved Continuous-Time model for Batch Sizing and Scheduling Processes," OPIM Department Report 01-12-03, December 2001.

[25] [Wang and Guignard (2001b)] S. Wang, M. Guignard, "An Efficient Lower Bound Estimation Model For The Batch Sizing/Scheduling Problem" OPIM Department Report 01-10-06, November 2001.

[26] [Wang and Guignard (2001a)] S. Wang, M. Guignard, "Integer Programming Approaches for Batch Sizing and Scheduling Problem", OPIM working Report 01-11-02 , November 2001.

[27] [Watson (1997)] Watson, Edward F "An application of discrete-event simulation for batch-process chemical-plant design". Interfaces. 27(6): 35-50. 1997 Nov/Dec.

[28] [Westenberger and Kallrath (1994)] H. Westenberger and J. Kallrath, "Formulation of a job shop problem in process industry," Working paper, Bayer AG, Leverkusen and BASF AG, Ludwigshafen, Germany, 1994.

[29] [Yan (1996)] Huan Yan, "Solving some difficult mixed-integer programming problems in production and forest management," Ph.D. Dissertation, OPIM Department, University of Pennsylvania, October 1996.

[30] [Zentner and Reklaitis (1992)] M.G. Zentner and G.V. Reklaitis, "An interval-based mathematical model for the scheduling of resource-constrained batch chemical processes," Proceedings of NATO ASI on Batch Processing Systems Engineering, Antalya, Turkey, 1992.

[31] [Zhang and Sargen (1993)] Z. Xueya and R. W. H. Sargent. A new unified formulation for process scheduling. In AIChE Annual Meeting, St. Louis, MO, 1993.

Appendix: Models and Formulation

A Notation

Indices

- p a product
- j a job
- m a machine

Sets

P	Set of products, includes raw material, intermediate products, and final products
F	Set of final products
$P^{in}(j)$	Set of products that are input of job j
$P^{out}(j)$	Set of products that are output of job j
J	Set of jobs
$J(m)$	Set of jobs that can be processed by machine m
$J^{in}(p)$	Set of jobs that use p as input
$J^{out}(p)$	Set of jobs that produce p as output
J^{multi}	Set of jobs that have multiple output
$P^{main}(j)$	Set of main products as output of j, if $j \in J^{multi}$. Notice $\|P^{main}(j)\| = 1$
M	Set of Machines
$M(j)$	Set of Machines that can process job j

Main Parameters:

$maxbsize(m)$	maximum batch size on machine m
$minbsize(m)$	minimum batch size on machine m
$l(j,m)$	batch length of job j processed on machine m
$f_{out}^{MIN}(j)$	minimum proportion requirement on main-product of output for job j
$f_{out}^{MAX}(j)$	maximum proportion requirement on main-product of output for job j
$f^{in}(j,p)$	fixed fraction that product p as part of the input for job j
$demand(p)$	final demand of product p
$juo0(p)$	initial stock for product p
$juo1(p)$	maximum stock for product p
H	a large number (upper bound on makespan)

B A Discrete-Time Model (Wang 1998)

Let the time horizon be equally divided by the greatest common divider of all the processing times of different tasks on different machines, and let $t \in \{1, 2, ...tott\}$ be the time index on the time horizon, where $tott$ is a large number which should be enough to schedule all the tasks to satisfy the final demands. (See, Figure 8.)

Then we can define the following variables and our discrete-time model.

Decision Variables

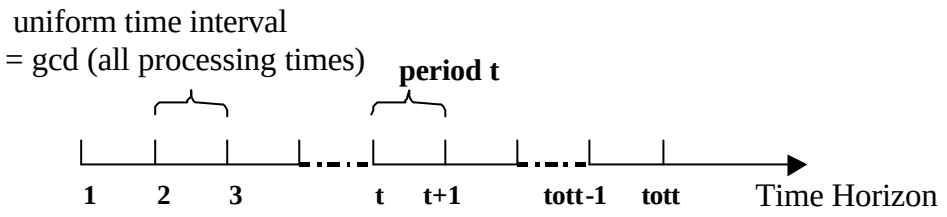

Figure 8: Discrete-time representation on the time horizon

$TSUM$	total time needed to complete the project, i.e., to produce all intermediary and finished products to meet the final demand.
$W(j,m,t)$	a 0-1 variable; equals to 1 if one specific batch of j is scheduled.on machine m beginning from time period t
$B(j,m,t)$	the amount of material which starts undergoing task j on machine m beginning from time period t, i.e., the batch size corresponding to $W(j,m,t)=1$.
$IP(p,t)$	inventory of product p at the beginning of time period t, i.e, at the time index point t
$PP(p,j,m,t)$	output of product p from job j on machine m at the end of period t, i.e, at time index point $t+1$.
$Busy(t)$	continuous variable, but only 0-1 value; equals to 1 if at least one machine is busy performing one task in period t, or equals to 0 if no machine is busy and no task is performed in period t.
	All variables nonnegative.

All variables are nonnegative.

Discrete-Time formulation

$$\min TSUM = \sum_{t} Busy(t) \tag{1}$$

st.

$$\sum_{j\in J(m)} \sum_{t_1=t-l(j,m)+1}^{t} W(j,m,t_1) \le 1,\ \forall m \in M,\ t < tott \tag{2}$$

$$minbsize(m)W(j,m,t) \le B(j,m,t) \le maxbsize(m)W(j,m,t),\ \forall m \in M,\ j \in J(m),\ t < tott \tag{3}$$

$$IP(p,t) \le juo1(p),\ \forall t \le tott, p \in P \tag{4}$$

$$IP(p,t) - IP(p,t-1) = \sum_{j\in J^{out}(p)} \sum_{m\in M(j)} PP(p,j,m,t-1) - \sum_{j\in J^{in}(p)} f^{in}(j,p) \sum_{m\in M(j)} B(j,m,t),$$
$$\forall p \in P, t \le tott, where\ IP(p,0) = juo0(p) \tag{5}$$

$$IP(p,tott) \ge demand(p), \forall p \in F \tag{6}$$

$$B(j,m,t) = \sum_{p \in P^{out}(j)} PP(p,j,m,t+l(j,m)-1), \forall j \in J^{multi}, m \in M(j), t < tott \quad (7)$$

$$f_{out}^{MIN}(j)B(j,m,t) \leq PP(p_1,j,m,t+l(j,m)-1) \leq f_{out}^{MAX}(j)B(j,m,t), \quad \forall j \in J^{multi}, m \in M(j), p_1 \in P^{main}(j) \quad (8)$$

$$Busy(t) \geq \sum_{j \in J(m)} \sum_{t_1=t-l(j,m)+1}^{t} W(j,m,t_1), \forall j \in J, m \in M(j), t < tott \quad (9)$$

$$Busy(t) \leq Busy(t-1), t < tott \quad (10)$$

$$0 \leq Busy(t) \leq 1, t < tott \quad (11)$$

The interpretation of the constraints is as follows. (1) states the objective function: the completion time equals to the number of periods during which the job shop is in busy status. (2) is the machine allocation constraint: at any given time period t, at most only one task can be executed on each machine (the operation is non-preemptive.). (3) is the batch sizing constraint: the amount of material that starts to undergo task j on machine m at the beginning of period t is bounded by the maximum and minimum capacities of that machine or unit. Note that this constraint forces the batch size $B(j,m,t)$ to be zero if $W(j,m,t) = 0$. (4) is the storage capacity constraint: the amount of material stored for product p at any time must not exceed the maximum storage capacity $juo1(p)$. (5) is the inventory balance constraint: This constraint simply states that the net increase $(IP(p,t) - IP(p,t-1))$ in the amount of material stored for product p during time period $t-1$ is given by the difference of the amount produced in this period and used for the next period. It also implies that the amount produced for product p is immediately sent to storage for product p or to a next job that uses p. (6) is the demand constraint: demand requirement for final products should be satisfied at the end of the whole process. (7) and (8) are proportion requirement constraints: the batch size of each job j should be equal to its output. Given a job $j \in J^{multi}$ with multiple output, whose main product is, say, $p_1 \in P^{main}(j)$, the ratio of the production quantity over the total output should be within the range $[f_{out}^{MIN}(j)\ ,\ f_{out}^{MAX}(j)]$. (9), (10) and (11) are the job shop performance constraints: these constraints state when the job shop is in busy status (i.e. variable $busy(t)$ equals to 1), and that if at least one machine is busy performing one task in period t, the previous periods should also be busy on at least one machine.

C Example of Activity-Based Continuous-Time Model (Wang and Guignard 2001)

Instead of using time index in a discrete-time model, continuous-time models employ time variables and time constraints. In our activity-based continuous-time formulation, all these time variables are indexed by related activities, which are numbered in a timing order via the time constraints in the formulation. The

concepts of activities were motivated by the "event" idea found in Ierapetritou & Floudas (1998). They develop the idea of using "*a* ***necessary number of event points*** corresponding to either the initiation of a job or the beginning of unit utilization".

Let n be an activity index, which denote the n^{th} activity on a machine or the n^{th} transfer of a product. More specifically, let (m,n) denote the n^{th} activity on machine $m \in M$, and (p,n) denote the n^{th} transfer of a product $p \in P$. The activities on every machine can be either "real" or "dummy" job. **Notice that there is one activity n per machine, and for a given product p, an activity can either use p, produce p, use and/or produce products other than p, or be dummy activities.**

We then use binary variable $W(j,m,n)$ to denote whether it is job $j \in J$ that is scheduled for the n^{th} activity of machine $m \in M$. Time variables $T^s(m,n)$ and $T^f(m,n)$ are continuous variables, which respectively denote the starting and finishing time of the n^{th} activity of machine m. We must be very careful in defining the inventory variable for each product $p \in P$. This variable is necessary for the "material balance" and the "demand" constraints. For the sake of convenience, we use non-negative variable $IP(p,n)$ as the inventory variable for product $p \in P$, as well as a non-negative time variable $T(p,n)$ as a time for transferring product p between machines and/or inventory storage. A complete variable definition list is presented as follows:

Variables

$TSUM$	makespan - the total time needed to complete the project, i.e., to produce all intermediary and finished products to meet the final demand.
$W(j,m,n)$	a 0-1 variable; equal to 1 if one specific batch of $j \in J$ is scheduled on activity $n \in N$ of machine $m \in M$
$B(j,m,n)$	A continuous variable. It is the amount produced for job $j \in J$ by machine $m \in M$ of activity $n \in N$ corresponding to $W(j,m,n) = 1$.
$T(p,n)$	a time when it is possible to change the inventory status of product p, it should be before starting to use p for activity n of a machine which consumes p (if n is a real activity of that machine), and it should be after the end of activity n-1 on a machine which produces p (if $n-1$ is a real activity on that machine)
$IP(p,n)$	the inventory level of product $p \in P$ between time $T(p,n)$ and $T(p,n+1)$
$PP(p,j,m,n)$	output of product $p \in P$ from job $j \in J$ of activity $n \in N$ on machine $m \in M$
$T^s(m,n)$	time at which activity n of machine m start to take place.
$T^f(m,n)$	time at which activity n of machine m finishes.

All variables are nonnegative.

We can now formulate the problem as a mixed-integer linear programming model. The challenge is to number the activities on all machines and the transfers of all products in a coherent, consistent fashion, so as to meet all scheduling requirements.

$$\min TSUM \tag{12}$$

s.t.

$$\sum_{j\in J(m)} W(j,m,n) \le 1,\ \forall m \in M,\ n < TOTN \tag{13}$$

$$T^f(m,n) \le T^s(m,n+1),\ \forall m \in M, n < TOTN - 1 \tag{14}$$

$$T^f(m,n) - T^s(m,n) = \sum_{j\in J(m)} l(j,m)W(j,m,n),\ \forall m \in M,\ n < TOTN \tag{15}$$

$$T(p,n) \le T(p,n+1) \qquad \forall p \in P,\ n < TOTN - 1 \tag{16}$$

$$T(p,n) \le T^s(m,n) + H(1 - W(j,m,n)) \qquad \forall p \in P,\ j \in J^{in}(p),\ m \in M(j) \tag{17}$$

$$T^f(m,n-1) \le T(p,n) + H(1 - W(j,m,n-1)) \qquad \forall p \in P,\ j \in J^{out}(p), m \in M(j) \tag{18}$$

$$T^s(m,n-1) \le T(p,n) + H(1 - W(j,m,n-1)) \qquad \forall p \in P,\ j \in J^{in}(p), m \in M(j) \tag{19}$$

$$T(p,n) \le T^f(m,n) + H(1 - W(j,m,n)) \qquad \forall p \in P,\ j \in J^{out}(p), m \in M(j) \tag{20}$$

$$T^s(m',n) \le T^f(m,n) + H(2 - W(j,m,n) - W(j',m',n)) \qquad \forall n < TOTN \tag{21}$$

$$T(p,n) \ge T^s(m,n) + H(W(j,m,n) - 1) \qquad \forall p \in P_N,\ j \in J^{in}(p), m \in M(j) \tag{22}$$

$$T^f(m,n-1) \ge T(p,n) + H(W(j,m,n-1) - 1) \qquad \forall p \in P_N,\ j \in J^{out}(p), m \in M(j) \tag{23}$$

$$TSUM \ge T^f(m, TOTN - 1) + lag(m) \qquad \forall m \in M \tag{24}$$

$$IP(p,n) - IP(p,n-1) = \sum_{j\in J^{out}(p)} \sum_{m\in M(j)} PP(p,j,m,n-1) - \sum_{j\in J^{in}(p)} f^{in}(j,p) \sum_{m\in M(j)} B(j,m,n), \quad \forall p \in P, \quad n \le TOTN \tag{25}$$

$$IP(p,n) \le juo1(p),\ \forall n \in N, p \in P \tag{26}$$

$$minbsize(m)W(j,m,n) \le B(j,m,n) \le maxbsize(m)W(j,m,n),\ \forall m \in M,\ j \in J(m),\ n < TOTN \tag{27}$$

$$IP(p,n) \ge demand(p)(1 - \sum_{j\in J^{out}(p), n\le n' < TOTN} \sum_{m\in M(j)} W(j,m,n')) \qquad \forall p \in F,\ n \le TOTN \tag{28}$$

$$B(j,m,n) = \sum_{p\in P^{out}(j)} PP(p,j,m,n),\ \forall j \in J^{multi}, m \in M(j), n < TOTN \tag{29}$$

$$f_{out}^{MIN}(j)B(j,m,n) \le PP(p_1,j,m,n) \le f_{out}^{MAX}(j)B(j,m,n),\ \forall j \in J^{multi}, m \in M(j),\ p_1 \in P^{main}(j) \tag{30}$$

The interpretation of the constraints is as follows. (13) is the allocation constraint: activity n of machine m can perform at most one job $j \in J(m)$; (14) is the "No Time Overlap" constraint: no time overlap is allowed for jobs on the same machine. These two constraints (13)(14) imply that no two jobs can be scheduled on the same machine at the same time.

(15) is the processing length constraint: the duration of activity n on machine m must equal the processing time $l(j,m)$ of the scheduled job j. (16) is the time sequence constraint: for product p, the n^{th} material transfer time should be no later than the $(n+1)^{st}$ transferring time.

Constraints (17) and (18) give the rules for numbering the material transfers: in order for product p to be available for the n^{th} activity on machine m, if that real activity corresponds to a job j that consumes p, the n^{th} transfer of p must take place before the start of that activity; In order for product p to become available at the end of activity $n-1$, if that real activity on machine m corresponds to a job j that produces p, the n^{th} transfer of p must take place after the end of that activity. Notice that the above time constraints (16), (17) and (18) imply the following inequality: for every pair of consecutive jobs (j,j') processed respectively on machines (m,m') as activities n and n', if $n < n'$, one must have: $T^f(m,n) \leq T^s(m',n') + H(2 - W(j',m',n') - W(j,m,n)), \forall n < n' < TOTN$

In addition to the restrictions placed by constraints (17) and (18), the two constraints (19) and (20) further restrict the variable $T(p,n)$: the n-th transfer of product p should occur after the start of all **real** activities $n-1$ on machines which consume p, and it should occur before the end of all **real** activities n on all machines which produce p.

Further more, constraint (21) gives the numbering rule for consecutive activities: for every pair of consecutive jobs (j,j'), if the two activities are real and both are numbered as the n-th activity on two machines m and m', since job j'can only use the output of job j from activities with a number smaller than n, the starting time $T^s(m',n)$ for job j' cannot be later than the finishing time $T^f(m,n)$ for job j.

The two constraints (22) and (23) are the reverse of the "numbering of transfers" constraints (17) and (18). Together with the next constraint (25), they will not allow any waiting time between consecutive jobs that produce or consume product $p \in P_N$ if there is no storage tank for that product.

(24) is the makespan constraint which defines the objective function: all final activities $(TOTN-1)$ on any machine m should be finished before time point $TSUM$, which is the makespan. Here $lag(m)$, the estimated minimum time lag from machine m to reach the final production, is a parameter used to tighten the constraints. For example, in Figure 1, there is no further step to go through after finishing the jobs on machine $M8$ or $M9$, therefore, $lag(\prime M8\prime) = lag(\prime M9\prime) = 0$. In addition, after processing the jobs on machine $M5$, one still needs at least $4u$ to reach the final production on $M8$ or $M9$, therefore, $lag(\prime M5\prime) = 4u$. Similarly, $lag(\prime M4\prime) = 4u$. The parameters $lag(m)$ can be easily obtained by using the shortest path algorithm.

The following constraints are almost the same as in the discrete-time model. (25) is the inventory balance constraint: at time point $T(p,n)$, the inventory level of product p must be adjusted by the amount of product p coming from all activities $n-1$ that produce p, or the amount of product p needed by all activities n that consume p. Value $f^{in}(j,p)$ is the fixed proportion of product p to the ingredients for job j. (26) is

the inventory capacity constraint: during time period $[T(p,n), T(p,n+1]$, the inventory level of product p cannot exceed the maximum inventory capacity $juo1(p)$. (27) is the batch sizing constraint: each batch has a variable batch size $B(j,m,n)$. Batch sizes $B(j,m,n)$ are restricted to remain between the maximum size $maxbsize(m)$ and the minimum size $minbsize(m)$. (28) is a tightened final-demand constraint. (29) and (30) are proportion requirement constraints: the batch size for each job j should be equal to its output. Given a job $j \in J^{multi}$ with multiple output, whose main product is, say, $p_1 \in P^{main}(j)$, the ratio of the production quantity of p over the total output of j should be within the range $[f_{out}^{MIN}(j)\ ,\ f_{out}^{MAX}(j)]$.