HUNGARIAN JOURNAL
OF INDUSTRIAL CHEMISTRY
VESZPRÉM
Vol. 30. pp. 305 - 312 (2002)

# SCHEDULING OF MULTIPURPOSE BATCH PROCESSES WITH MULTIPLE BATCHES OF THE PRODUCTS

T. HOLCZINGER, J. ROMERO[1], L. PUIGJANER[1] and F. FRIEDLER

(Department of Computer Science, University of Veszprém,
Veszprém, Egyetem u. 10, H-8200, HUNGARY
[1]Chemical Engineering Department, Universitat Politècnica de Catalunya,
E.T.S.E.I.B., Diagonal 647, E-08028 Barcelona, SPAIN)

Received: December 12, 2002

Optimal scheduling of batch processes becomes excessively complex if large number of batches of the products has to be generated. In most cases, however, the search space of the optimization procedure can be drastically reduced by eliminating the redundant solutions through additional combinatorial constraints. The proposed approach that generates these additional constraints is described here for a graph-theoretical method for batch process scheduling, nevertheless, it can be conveniently embedded into other methods developed for optimal scheduling.

**Keywords:** multipurpose batch plants, multiple batches, optimal scheduling, S-graph, search space reduction

## Introduction

In multipurpose plants, a variety of production resources (raw materials, equipment, utilities, manpower) are shared by a number of processing operations that manufacture several products. The types of operations involved can vary from continuous to batch, a fact that given the inherent flexibility of such types of plants, leads to complex scheduling problems.

The problem of short-term scheduling of batch plants seeks to determine the optimal strategy for satisfying the production demand of a variety of products at specific dates and/or at the end of a given production horizon. The short-term scheduling is specially relevant for flexible production networks (multipurpose plants), where the production of individual batches, even for the same product, does not follow the same pattern but must be specified according to an overall performance index and is subject to capacity and time constraints. It involves the allocation of equipment and resources to orders, the sequences of these orders and the route determination of the material flows through the plant.

Short-term scheduling of multipurpose batch plants has received considerable attention over the past two decades. Many diverse approaches, mathematical formulations and solution algorithms have been proposed. Recent reviews can be found in [1-4].

A major question in any scheduling algorithm deals with the time problem representation. Kondili *et al.* [5] present a formulation to the modeling of a series of scheduling problems that arises in batch plants, namely the State Task Network (STN). This representation is based on a uniform time discretization and on the assumption that events only happen at the boundaries of these intervals, which implies the generation of a large number of integer variables in problems of industrial relevance. The computational effort has been reduced by reformulating the allocation constants [6] or by heuristic decomposition [7]. These improvements have encouraged recent work toward the development of efficient methods of comparable generality based on the continuous-time representation, which was first introduced by Sahinidis and Grossmann [8].

The Resource-Task-Network (RTN) representation proposed by Pantelides [9] was the basis for a continuous time formulation by Zang and Sargent [10]. Later, Shilling and Pantelides [11] presented a simpler RTN formulation. However, the resulting MILP problem is still very cumbersome to solve.

Otherwise, an approach that has been traditionally used is based on a graph representation combined with a branch and bound method. These techniques for the case of scheduling, known as edge finding methods [12-14] have proved to be very effective for solving special types of job shop scheduling problems. An extensive

Correspondence concerning this article should be addressed to F. Friedler (friedler@dcs.vein.hu)

computational study of the problem was also performed by Applegate and Cooke [15], in which the authors develop heuristics for finding feasible schedules, cutting planes for obtaining lower bounds and a specialized branch and bound method. Very recently, a new graph representation called S-graph, appropriate for combinatorial algorithms, has been introduced [16, 17]. Once all process tasks are represented in a recipe-graph, an appropriate search strategy permits to generate the S-graph of the optimal schedule effectively, i.e., a drastic reduction of computation time can be achieved compared to mathematical programming solution techniques. In this paper, the properties of scheduling problems are further exploited by embedding an additional acceleration tool in the solver, which results in an increased computational efficiency needed in the solution of large scheduling and re-scheduling problems.

## S-graph representation

The present work is based on the S-graph representation [16] and a corresponding general framework [17] that can solve different types of production scheduling problems. In this framework, a node (so-called task node) is assigned to each task given in the recipe for producing a single batch of a product or products. An additional node (so-called product node) is introduced for each product for technical reasons. It is supposed that there is at least one equipment unit to perform each task; set *Si* denotes the set of those equipment units that can perform the task represented by task node *i*. The processing orders of the tasks are given by weighted arcs (so-called recipe-arcs) of the graph. The processing time of a task may be different for different equipment units. In this case, the weight of the recipe-arc is the minimum of the processing times of the plausible equipment units. In this representation, the value assigned to an arc expresses a lower bound for the difference of the starting times of the two related tasks. The resultant graph is the recipe-graph for the single batch production. For generating multiple batches of the products, the appropriate part of this S-graph is repeated according to the number of batches to get the recipe-graph of multiple batches.

An S-graph can represent both non-intermediate storage (*NIS*) and unlimited intermediate storage (*UIS*) policies appropriately. For the former, let $t_j$ denote the set of those tasks that follow task *j* according to the recipe. If equipment unit *Ei* is assigned to task *j* and consecutively to task *k*, then, a zero weighted arc (or an arc whose weight is equal to the length of the changeover time if applicable), called schedule-arc, is established from each element of $t_j$ to *k*. For example, 2-6 is the task sequence of equipment unit E1 shown in *Fig.1*. In this example, E1 is assigned to tasks 2 and 6 and zero weighted arcs are established from the tasks that follow task 2 ($t_j$={3,4}) to task 6 (consecutive task of task 2). If *UIS* policy is applied in any part of the scheduling problem or globally for the whole problem,

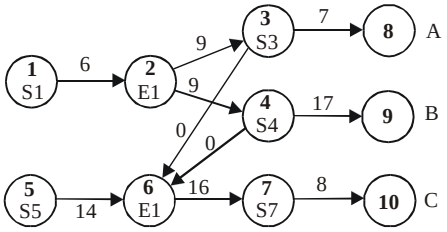


*Fig.1* S-graph representation of task sequence 2-6 for equipment unit E1 with *NIS* policy

the introduction of an additional node or nodes for the "unlimited" storage unit or units transforms the policy of the problem to *NIS*. In the following, only the *NIS* policy will be considered.

Formally, directed graph *G* can be given as a pair $(N, A)$ where *N* is a finite set, the set of nodes, and *A* is a set of pairs of nodes identifying the arcs of the graph (i.e., $A \subseteq N \times N$ ). In an S-graph, two classes of arcs, the so-called recipe-arcs and schedule-arcs are specified. Therefore, an S-graph is given in the form of $G(N, A_1, A_2)$ , where *N*, $A_1$, and $A_2$ denote the sets of nodes, recipe-arcs, and schedule-arcs, respectively. It is supposed that $A_1 \subseteq N \times N$ , $A_2 \subseteq N \times N$ , and $A_1 \cap A_2 = \varnothing$ ; furthermore, a nonnegative value, $c(i, j)$ , assigned to each arc, denotes the weight of the arc $(i, j)$ . In practice, if an arc is established from node *i* to node *j*, i.e., $(i, j) \in A_1 \cup A_2$ , then it is supposed that the task corresponding to node *j* cannot start its activity earlier than $c(i, j)$ time after the task corresponding to node *i* started.

## Multiple batches of a product

A simple way of describing the multipurpose scheduling problems with more than one batch of the products is by considering each batch as an individual product. Even though the basic algorithm published in [17] can be used directly for solving this simplistic model, it is not necessarily efficient enough if the number of batches of the same product is excessively large. Embedding additional tools into the basic algorithm, however, may result in a sufficient acceleration for solving large-scale problems. First, a simple example will illustrate the source of inefficiency to be overcome by the embedded tools.

Example 1

Suppose that two batches of product A and one batch of product B are to be produced. Product A is produced in three consecutive steps that can be performed by any element of the sets of equipment units S1, S2, and S3, respectively. Product B is produced in two consecutive steps, where the first step and second step can be performed by any equipment unit given in sets S4 and

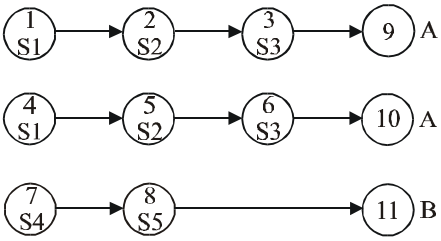


*Fig.2* Recipe-graph of Example 1

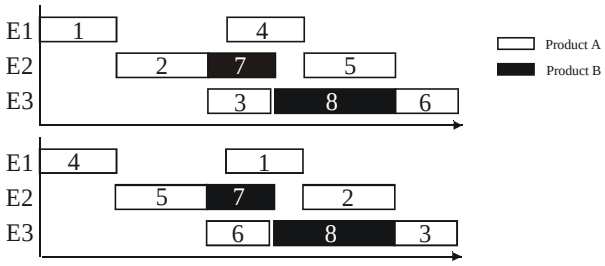


*Fig.3* Two technically identical schedules

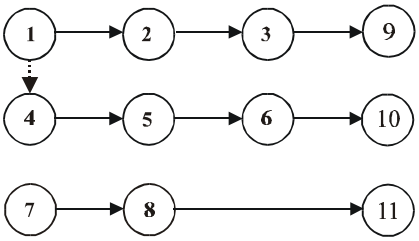


*Fig.4* Recipe-graph of Example 1 extended by an auxiliary-arc expressing the order of the two batches of product A

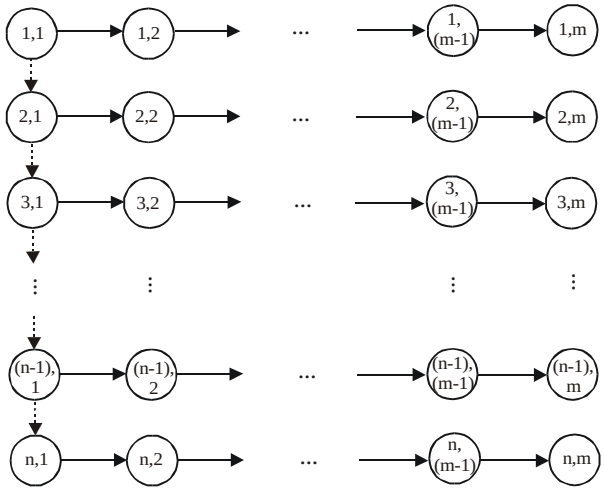


*Fig.5* Recipe-graph extended by auxiliary-arcs if n batches of the product are to be produced

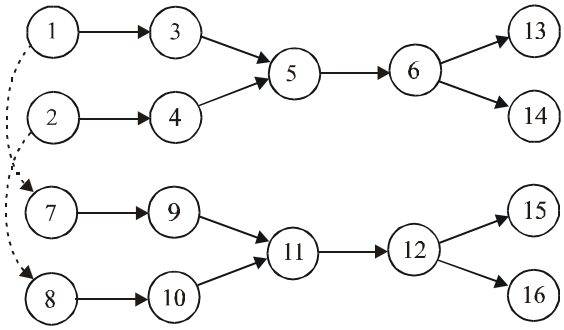


*Fig.6* Recipe-graph extended by auxiliary-arcs in case of complex recipe for 2 batches

S5, respectively. The recipe-graph of this example is given in *Fig.2*.

It is assumed that equipment units E1, E2, and E3 are in sets S1, S2∩S4, and S3∩S5, respectively. Two technically identical schedules are shown in *Fig.3*; only one of them should be considered, even though both of them are generated by the basic algorithm.

As far as the example illustrates, the source of inefficiency is the possibility of generating different orders of technically identical batches. This redundancy in the set of solutions can, however, be eliminated by additional constraints. For example, if it is supposed that the activity denoted by task node 4 cannot start before the starting time of the activity denoted by task node 1, redundancy appearing in *Fig.3* is excluded. This constraint can be conveniently represented by an additional arc from node 1 to 4 in the S-graph (see *Fig.4*). Note that, the additional arc or arcs to exclude redundant solutions are not considered as recipe-arcs or schedule-arcs, they will be cited as auxiliary-arcs and illustrated by dotted lines.

In general, if more than two batches of a product are to be produced, the redundancy is even more serious. For instance, *n* factorial, practically identical solutions can be generated for *n* batches of a product. An ordering on the starting time of the task nodes of the first task of the product excludes the unnecessary permutations of the batches from consideration. This ordering is represented by arcs in the S-graph as *Fig.5* illustrates. Naturally, the optimal solution of the scheduling problem with the additional constraints is also an optimal solution of the original problem. If more than one initial tasks appear in the recipe of one product, the redundancy can be similarly treated (see, e.g., *Fig.6*).

The auxiliary-arcs exclude all but one permutations of the batches from the search space of the scheduling algorithm; nevertheless, further reductions can be achieved in certain cases.

*Single equipment unit for a task*

Assume that one equipment unit is available to perform a task of a product. In this case, the ordering of the starting times of the task nodes of the second task of the product must be the same as the ordering of the starting times of the task nodes of the first task. Similarly, the ordering of the starting times of the task nodes of the third task is the same as that of the task nodes of the second task, etc. The resultant auxiliary-arcs are shown in *Fig.7*. Since *NIS* policy is considered, further sharpening of the constraints can be achieved.

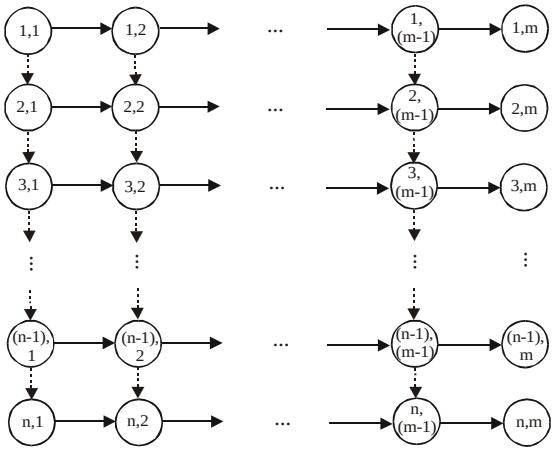


*Fig.7* Recipe-graph extended by auxiliary-arcs if one equipment unit is available to perform a task of a product

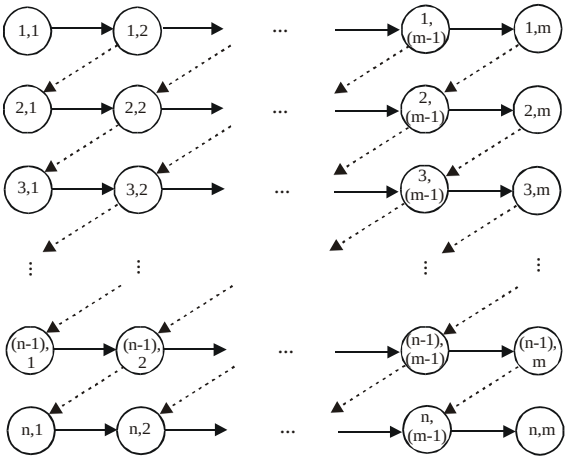


*Fig.8* Recipe-graph with transformed auxiliary-arcs if one equipment unit is available for each task

An equipment unit assigned to task node (*i*, *j*) shown in *Fig.7* (*i*-th batch, *j*-th task) can be reassigned to task node (*i*+1, *j*) when the activity represented by task node (*i*, *j*) is performed and the produced material is transferred to the equipment unit assigned to node (*i*, *j*+1). Consequently, the starting point (*i*, *j*) of an auxiliary-arc can be moved one step forward to (*i*, *j*+1) as shown in *Fig.8*. In addition to the exclusion of the redundant permutations of the batches, this additional transformation of auxiliary-arcs sharpens the bound in the procedure of generating the optimal solution resulting in additional acceleration.

A complex recipe can be treated similarly, however, if more than one task follow a task in the recipe, an auxiliary-arc is established from each of the task nodes of the consecutive tasks (see, e.g., *Fig.9*).

### *Optional equipment units for a task with identical processing time*

If at least two equipment units are available to perform a task, the statement given in the previous section and illustrated in *Figs.8* and *9* is not valid since a concurrent

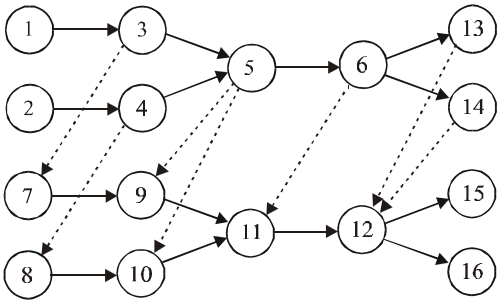


*Fig.9* Recipe-graph with transformed auxiliary-arcs in case of a complex recipe for two batches: one equipment unit is available for each task

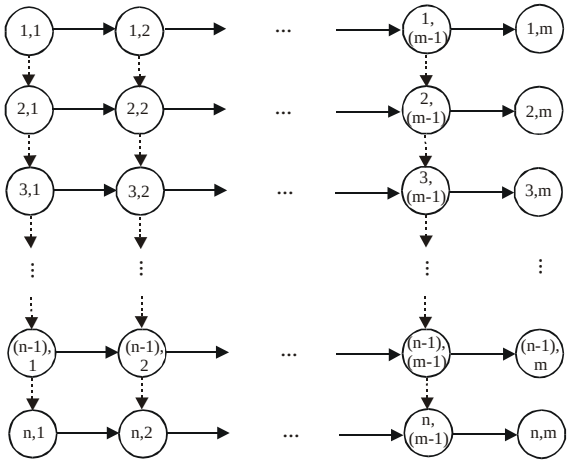


*Fig.10* Recipe-graph with auxiliary-arcs: multiple equipment units are available for a task, their processing times are identical

equipment unit may start the operation of a task earlier than the corresponding task is finished in a batch started earlier. Nevertheless, if the processing times of all equipment units available for a task are identical, there is an optimal solution with the property that the order of the starting times of the corresponding task nodes of a task is the same as the starting times of the task nodes of their previous task. This results in auxiliary-arcs, e.g., shown in *Fig.10* and in *Fig.11*.

In general, the combination of cases given in the previous and this sections occurs in practice. Furthermore, some tasks in the recipe that can be performed by multiple equipment units may have different processing times. In this case, the initial task nodes can be chained with zero weighted arcs as shown in *Figs.5* and *6*, nevertheless, consecutive tasks can be chained only if the following two additional properties are satisfied.

(P1) The task can be performed by one equipment unit or the task can be performed by multiple equipment units with identical processing times.

(P2) All the preceding tasks satisfy property (P1).

If a task can be performed by a single equipment unit, the related auxiliary-arc is transformed as shown in *Figs.8* and *9*.

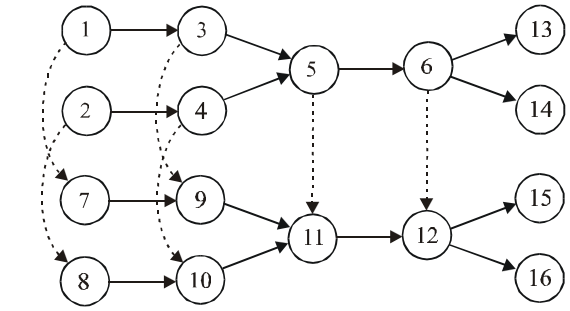


*Fig.11* Recipe-graph with auxiliary-arcs for a complex recipe: multiple equipment units are available for a task; their processing times are identical

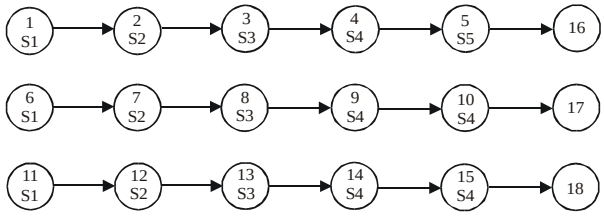


*Fig.12* Recipe-graph of Example 2

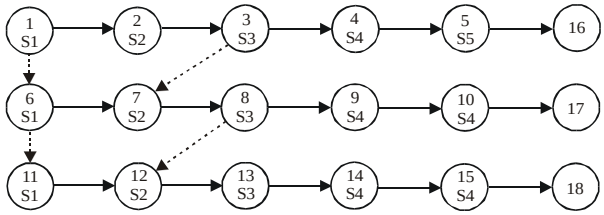


*Fig.13* Recipe-graph of Example 2 with auxiliary-arcs

Example 2

For the recipe-graph shown in *Fig.12*, suppose that S1={E1, E2}, S2={E3}, S3={E4, E5}, S4={E1}, and S5={E2}. Furthermore, the processing time for the first task, i.e., the processing times related to task nodes 1, 6, and 11, is independent of the selection of E1 or E2 while the processing time of the third task depends on the selection of the equipment unit. The first, the second, the fourth, and the fifth tasks satisfy property (P1); moreover, the first, the second, and the third tasks satisfy property (P2). Therefore, the task nodes of the first task is to be chained by auxiliary-arcs and the task nodes of the second task is to be chained by transformed auxiliary-arcs (see *Fig.13*).

## Program realizations and applications

The basic framework algorithm that has been published in [17] has two main steps: the generation of the recipe-graph and the branch-and-bound type scheduling algorithm based on S-graph (see steps 1 and 3 in *Fig.14*). The present work accelerates the algorithm for multiple batches by an additional step, step 2 (see

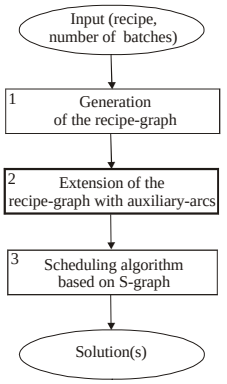


*Fig.14* Solution procedure

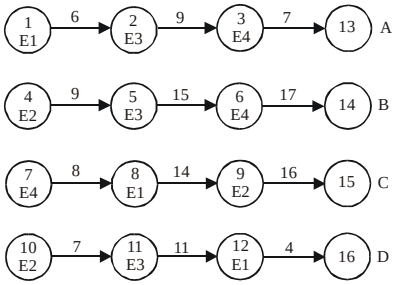


*Fig.15* Recipe-graph of Example 3 for four batches

*Fig.14*). The algorithm has been realized in C++, its demonstration version is available at http://www.dcs.vein.hu/demo/hjic together with illustrative examples. Although the acceleration tool of the present work has been developed for the S-graph framework, it is not specific to S-graphs and, therefore, can be fitted to other scheduling algorithms.

Example 3

This example is introduced by Sanmartí *et al.* [17]; four equipment units, E1, E2, E3, and E4, are available to generate four products, A, B, C, and D. The recipes of the products are given in *Fig.15*. The basic algorithm and its acceleration have been compared by solving this example with different number of batches per products; the results are shown in *Table 1*.

Example 4

This example is introduced by Voudouris and Grossmann [18]. Five equipment units (stages), E1, E2, E3, E4, and E5, are available to generate four products, A, B, C, and D. The recipes of the products are given in *Fig.16*.

The basic algorithm and its acceleration have been compared by solving this example with different number of batches per products, the results are shown in *Table 2* (solved by PC-Pentium 667 MHz). For more than eleven batches the basic algorithm did not arrive at the optimal solution in reasonable time (in several hours).

*Table 1* Comparison of the algorithms in generating an optimal solution

| Number of batches | | | | Basic algorithm [Sanmartí et al. [17]] CPU time (s)* | Accelerated algorithm [present work] CPU time (s)* | Acceleration ratio |
|---|---|---|---|---|---|---|
| A | B | C | D | | | |
| 1 | 1 | 1 | 1 | 0.17 | 0.17 | 1 |
| 2 | 1 | 1 | 1 | 2.42 | 1.26 | 1.92 |
| 2 | 2 | 1 | 1 | 11.54 | 3.02 | 3.82 |
| 2 | 2 | 2 | 1 | 142.11 | 20.76 | 6.85 |
| 2 | 2 | 2 | 2 | 3019.86 | 81.72 | 36.95 |
| 3 | 2 | 2 | 2 | N/A | 329.5 | N/A |
| 3 | 3 | 2 | 2 | N/A | 1169.04 | N/A |
| 3 | 3 | 3 | 2 | N/A | 6577.11 | N/A |

*PC-Pentium 667 MHz

*Table 2* Comparison of the algorithms in generating an optimal solution

| Number of batches | | | | Base algorithm [Sanmartí et al. [17]] CPU time (s)* | Accelerated algorithm [present work] CPU time (s)* | Acceleration ratio |
|---|---|---|---|---|---|---|
| A | B | C | D | | | |
| 1 | 1 | 1 | 1 | 0.17 | 0.17 | 1 |
| 2 | 1 | 1 | 1 | 0.55 | 0.45 | 1.22 |
| 2 | 2 | 1 | 1 | 0.55 | 0.22 | 2.5 |
| 2 | 2 | 2 | 1 | 15.71 | 3.63 | 4.32 |
| 2 | 2 | 2 | 2 | 72.22 | 6.59 | 10.96 |
| 3 | 2 | 2 | 2 | 309.06 | 12.32 | 23.2 |
| 3 | 3 | 2 | 2 | 72.83 | 6.71 | 10.85** |
| 3 | 3 | 3 | 2 | 26827.26 | 72.56 | 369.73 |
| 3 | 3 | 3 | 3 | N/A | 233.61 | N/A |
| 4 | 3 | 3 | 3 | N/A | 390.08 | N/A |
| 4 | 4 | 3 | 3 | N/A | 161.59 | N/A |
| 4 | 4 | 4 | 3 | N/A | 2530.26 | N/A |
| 4 | 4 | 4 | 4 | N/A | 7579.39 | N/A |

*PC-Pentium 667 MHz
**Instance solved by Voudouris and Grossmann [18]

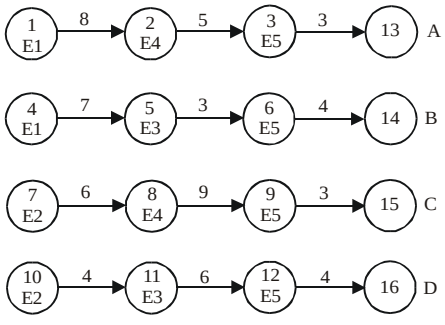


*Fig.16* Recipe-graph of Example 4 for four batches

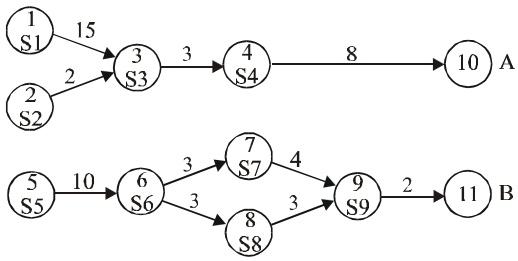


*Fig.17* Recipe-graph of Example 5 for two batches

Note that, the running time of the algorithm is not necessarily monotonic with the number of batches, as shown in *Table 2*, since the number of batches may affect the structure of the problem that may make a smaller problem more complex in some cases.

Voudouris and Grossmann [18] published the result of the case 3, 3, 2, and 2 batches of products A, B, C, and D, respectively with 293 second running time using GAMS 2.25/Sciconic 2.11 on an IBM/R6000/Power 530 workstation (see the highlighted row of *Table 2* for comparison).

Table 3 Comparison of the algorithms in generating an optimal solution

| Number of batches | | Base algorithm [Sanmartí et al. [17]] CPU time (s)* | Accelerated algorithm [present work] CPU time (s)* | Acceleration ratio |
|---|---|---|---|---|
| A | B | | | |
| 1 | 1 | 0.06 | 0.06 | 1 |
| 2 | 1 | 0.16 | 0.11 | 1.45 |
| 2 | 2 | 0.38 | 0.27 | 1.4 |
| 3 | 2 | 1.26 | 0.49 | 2.57 |
| 3 | 3 | 6.76 | 0.93 | 7.27 |
| 4 | 3 | 46.09 | 1.59 | 28.99 |
| 4 | 4 | 386.51 | 2.96 | 130.58 |
| 5 | 4 | 3632.72 | 5.11 | 710.9 |
| 5 | 5 | 41667.74 | 9.94 | 4191.93 |
| 6 | 5 | N/A | 17.81 | N/A |
| 6 | 6 | N/A | 36.74 | N/A |
| 7 | 6 | N/A | 68.01 | N/A |
| 7 | 7 | N/A | 144.46 | N/A |
| 8 | 7 | N/A | 273.48 | N/A |
| 8 | 8 | N/A | 565.68 | N/A |

*PC-Pentium 667 MHz

Example 5

Two products (product A and B) are to be produced according to recipe given in *Fig.17*. *Si* (i=1,2,..., 9) denotes the set of those equipment units that can perform task *i*. The sets are specified as S1={E1}, S2={E2}, S3={E3}, S4={E4}, S5={E1}, S6={E2}, S7={E3}, S8={E4}, and S9={E5}. *Table 3* shows the result for different number of batches.

## Concluding remarks

By eliminating redundant solutions, additional combinatorial constraints may drastically reduce the search space of optimal scheduling of batch processes for generating multiple batches of the products. This type of scheduling problems can be solved more effectively with these additional constraints as its applications show for the S-graph based scheduling methodology.

## Acknowledgement

This project has been financially supported in part by the Hungarian National Science Foundation OTKA T-029-309 and by the MCyT (Project No. OCCASSION: DPI2002-00856).

## REFERENCES

1. PINTO J. M. and GROSSMANN I. E.: Ind. Eng. Chem. Res., *1995*, 34, 3037
2. SHAH N.: Single-and Multisite Planning and Scheduling: Current Status and Future Challenges, Foundations of Computer-Aided Process Operations. AIChE Symposium Series No.320. (Eds. Joseph F. Pekny and Gary E. Blau), American Institute of Chemical Engineers (AIChE), New York, 94, 75, 1998
3. PEKNY J. and REKLAITIS G. V.: Towards the Convergence of Theory and Practice: A Technology Guide for Scheduling /Planning Methodology, Foundations of Computer-Aided Process Operations. AIChE Symposium Series No.320. (Eds. Joseph F. Pekny and Gary E. Blau), American Institute of Chemical Engineers (AIChE), New York, 94, 91, 1998
4. PUIGJANER L.: Computers Chem. Engng., *1999*, 23S, S929
5. KONDILI E. C., PANTELIDES C. C. and SARGENT R. W. H.: A General Algorithm for Scheduling of Batch Operations, Proc. 3rd Intl. Symp. On Process Systems Engng., Sydney, Australia, 62-75, 1988
6. SHAH N., PANTELIDES C. C. and SARGENT R. W. H.: Computers Chem. Engng., *1993*, 17, 229
7. ELKAMEL A.: Scheduling of Process Operations using Mathematical Programming Techniques, PhD Thesis, Purdue University, 1993
8. SAHINIDIS N. V., GROSSMANN I. E., FORNARI R. E. and CHATHRATHI M.: Computer Chem. Engng., *1991*, 15, 255
9. PANTELIDES C. C.: Unified Frameworks for Optimal Process Planning and Scheduling, Proceedings of

the Second Conference on Foundations of Computer Aided Operations (FOCAPOII), 235-274, 1994
10. ZANG, X and SARGENT R. W. H.: Computers Chem. Engng., *1996*, S20, S1287-S1292
11. SCHILLING G. and PANTELIDES C. C.: Computers Chem. Engng., *1996*, S20, S1221
12. ADAMS J., BALAS E. and ZAWACK D.: Management Science, *1998*, 34, 391-401
13. CARLIER J. and PINSON E.: Management Science, *1989*, 35, 164-176
14. CORMEN T. H., LEISERSON C. E. and RIVEST R. L.: Introduction to algorithms, The MIT Press, 1997
15. APPLEGATE D. and COOKE W.: ORSA Journal of Computing, Spring, *1991*, 3(2), 149-156
16. SANMARTÍ E., FRIEDLER F. and PUIGJANER L.: Computers chem. Engng., *1998*, 22, S847-S850
17. SANMARTÍ E., HOLCZINGER T., PUIGJANER L. and, FRIEDLER F.: AIChE Journal, *2002*, 48(11), 2557-2570
18. VOUDOURIS V. T. and GROSSMANN I. E.: Computers chem. Engng., *1994*, 20(11), 1335-1360