Chemical Engineering Science 64 (2009) 605 – 610

Contents lists available at ScienceDirect

Chemical Engineering Science

journal homepage: www.elsevier.com/locate/ces

Practical infeasibility of cross-transfer in batch plants with complex recipes: S-graph vs MILP methods

M. Hegyháti[a], T. Majozi[a,b], T. Holczinger[a], F. Friedler[a,*]

[a]Department of Computer Science, University of Pannonia, Egyetem u. 10, Veszprém, H-8200, Hungary
[b]Department of Chemical Engineering, University of Pretoria, Lynnwood Road, Pretoria 0002, South Africa

ARTICLE INFO

Article history:
Received 9 May 2008
Received in revised form 4 October 2008
Accepted 10 October 2008
Available online 1 November 2008

Keywords:
Batch
Chemical process
Optimization
Systems engineering
S-graph
Scheduling

ABSTRACT

Multipurpose batch processes entail various operational policies that have been widely investigated in published literature. In this paper, no-intermediate-storage (NIS), zero-wait (ZW) and common-intermediate-storage (CIS) operational policies are of particular interest. In all these policies, no dedicated storage facility is available between two consecutive units. Unlike the other operational policies, these particular policies bear some subtle practical infeasibility that has gone unnoticed in literature. In essence, this infeasibility has been reported as optimal, thus assumed to be feasible, by various authors using mathematical programming techniques. It pertains to a unit transferring product to one or more units whilst simultaneously receiving feed from another, which is practically infeasible and as such need not be considered as a possible solution. This feature is particularly conspicuous in batch processes with complex recipes wherein production paths can be in opposite directions. This paper presents the unique feature of the S-graph framework to isolate cross-transfer during optimization, whereas the available mathematical programming methods inherently fail neither to detect nor to eliminate this infeasibility. A few examples taken from published literature are presented for demonstration purposes.

1. Introduction

The problem of batch process scheduling has been under investigation for the last three decades following the emergence of high value added specialty chemicals as key contributors to the economy. Contrary to their continuous counterparts, batch processes have an intrinsic ability to adapt to sudden changes in market conditions which is the characteristic that has made them attractive in low volume manufacturing. In particular, scheduling involves the allocation of tasks to limited resources with the intention of either minimizing the makespan or maximizing throughput. Minimizing makespan is encountered where the production profile is known *a priori* and the objective is to realize it in the shortest possible time. On the other hand, the maximization of throughput problem relates to a presupposed time horizon over which maximum production should be achieved. The latter problem is also ideally suited for economical objectives like maximization of revenue or profit.

Recent advances in the field of batch process scheduling have aimed to address the issue of lengthy computational times emanating from the combinatorial nature of scheduling problems. In Kondili et al. (1993) developed a mixed integer linear programming (MILP) formulation for throughput maximization together with a recipe representation known as the state task network (STN). The mathematical formulation was based on even discretization of the time horizon which resulted in computational difficulties due to the excessive number of binary variables. Subsequent developments, therefore, were aimed at reducing the binary dimension through uneven discretization of the time horizon using a presupposed number of time points. This technique was initially proposed by Schilling and Pantelides (1996). Notable extensions to this contribution involve the work of Castro et al. (2001), Castro and Grossmann (2006), Ierapetritou and Floudas (1998), and Majozi and Zhu (2001). A detailed review on uneven discretization of time horizon formulations is given by Floudas and Lin (2004). The major limitation of all these methods is that the degree of optimality depends on the presupposed number of time points.

Added to the issue of computational times, much research effort has been directed towards scheduling of batch plants that are characterized by complex recipes with the aim of makespan minimization. The recipe complexity provides another dimension of combinatorial difficulty in scheduling. In this class of batch operations, different batches not only follow different routes, but they also involve production paths in opposite directions. The contributions of

* Corresponding author. Tel.: +36 88 424 483; fax: +36 88 428 275.
E-mail address: friedler@dcs.uni-pannon.hu (F. Friedler).

0009-2509/$ - see front matter © 2008 Published by Elsevier Ltd.
doi:10.1016/j.ces.2008.10.018

Kim et al. (2000) and Méndez and Cerdá (2003) are noticeable in this regard. In both of these contributions the subtle infeasibility associated with cross-transfer is seriously overlooked and reported as an optimum, hence this paper.

Recently, a graph theoretic framework known as the S-graph has been introduced with great capabilities in combinatorial problems (Sanmartí et al., 1998, 2002). The strength of this framework essentially lies in its capability to directly exploit the problem structure with drastic reduction in computational intensity. Once all the processing tasks have been represented in the recipe, an optimal schedule can be generated using the S-graph framework. Recent comparisons in performance with respect to CPU time have demonstrated that the S-graph tends to obtain the optimal solutions relatively faster than the mathematical techniques (Holczinger et al., 2007). The S-graph framework has been successfully applied to both minimization of makespan (Romero et al., 2004; Sanmartí et al., 2002) and maximization of throughput problems (Majozi and Friedler, 2006).

In this paper, the inherent capability of the S-graph framework to isolate and exclude infeasible candidate solutions during the course of optimization is highlighted through comparison with published mathematical programming techniques. This contribution was motivated by the fact that published mathematical programming methods frequently generate infeasible schedules due to the existence of cross-transfer which is inherent in model formulation. Of particular interest to this paper are the no-intermediate-storage (NIS), zero-wait (ZW) and common-intermediate-storage (CIS) operational policies. In the NIS policy a product is allowed to stay in the processing unit until the next unit is available to conduct the subsequent task in the recipe, whilst in ZW operational policy the next task has to commence immediately after the preceding task the recipe. The CIS policy involves a storage facility that is shared by various products within the schedule.

Overall, the paper is organized as follows. Section 2 gives the elaborate description of the problem at hand followed by Section 3 with a focus on the drawback of mathematical techniques. Section 4 highlights the capabilities of the S-graph in isolating and excluding the cross-transfer infeasibility, whilst Section 5 gives the comparison of solutions obtained from mathematical programming techniques and S-graph based algorithms. Lastly, conclusions are given in Section 6.

2. Problem description

Fig. 1 shows the Gantt chart of the production of two products, P1 and P2, that share processing units U1 and U2. According to the depicted schedule, P1 is transferring its intermediate from U1 to U2 at time t_1', whilst P2 is doing the same, albeit in the opposite direction, hence the cross-transfer.

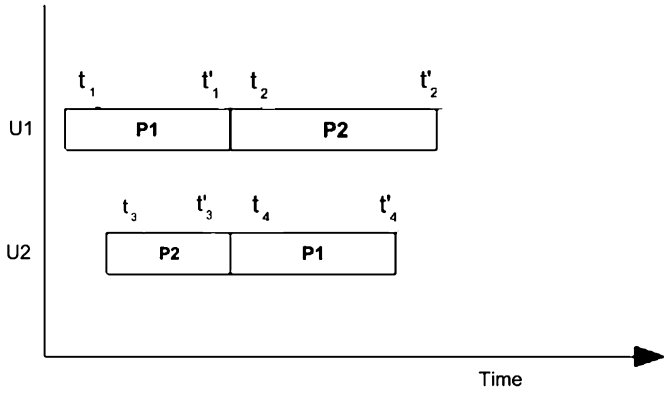

Fig. 1. Gantt chart showing cross-transfer.

In physical terms, the schedule in Fig. 1 stipulates that U1 is discharging the intermediate corresponding to P1 whilst simultaneously receiving the intermediate corresponding to P2. Similarly, U2 is receiving the intermediate corresponding to P1 whilst discharging the intermediate corresponding to P2. In the case of NIS, CIS and ZW policies this occurrence is certainly infeasible from the practical standpoint and as such cannot be considered as a possible solution. Practically, each unit has to be discharged completely before any feed can be introduced. Although only two units and two products have been used in the foregoing description, similar infeasibility could be encountered where more than two units and products are involved. Regardless, of its practical infeasibility, this occurrence has been reported widely in literature as an optimal solution.

Given the obvious nature of this practical infeasibility and the fact that it has not been isolated by the available mathematical programming techniques (see for example Kim et al., 2000; Méndez and Cerdá, 2003), close scrutiny into this drawback of mathematical techniques is necessary. Section 3 focuses on why known mathematical techniques inherently fail to detect and exclude this type of infeasibility.

3. Limitations of the MILP approach in eliminating cross-transfer

There exists various mathematical formulations of a scheduling problem. Most of these formulations exhibit a MILP structure, but differ in the representation of the time horizon of interest. Overall, some are based on uniform time discretization and others on non-uniform time discretization. A detailed account on these has been given by Floudas and Lin (2004).

Regardless of their fundamental distinction, mathematical methods behave similarly at the optimal point for the same objective function, e.g. minimization of makespan. In the solution, the binary variables are fixed and the sequential constraints are similar. Two types of sequential constraints are inherent in the description of a scheduling problem. These are the sequence constraints of two consecutive tasks for a particular product as defined by the recipe and sequence constraints of two consecutive tasks in the same unit as defined by the production schedule. Of particular interest in this paper is when the first type of sequence constraints involves separate units and the second type involves different products.

The subtle infeasibility that occurs frequently in the solutions of mathematical formulations is a consequence of the following analysis. Let t_i denote the starting time and t_i' the finishing time of task i, where $t_i \leqslant t_i'$. The difference of the starting and finishing time, which signifies the duration of a task in a unit, can be different for different mathematical models. Nevertheless, it always contains the processing time of the task and waiting time of the material in the current equipment unit in the case of NIS policy. For the ZW operational policy the waiting time is always zero. It can also contain transfer time.

In a situation where consecutive tasks of a recipe take place in different units, the sequence constraint can be expressed as $t^f_{ij} \leqslant t^s_{i'j'}$, where t^f_{ij} is the finishing time of task i in unit j and $t^s_{i'j'}$ is the starting time of task i' in unit j'. Both i and i' belong to the same recipe. In the NIS, CIS and ZW cases $t^f_{ij} = t^s_{i'j'}$. On the other hand, if two consecutive tasks involve the same unit, and not necessarily the same recipe, the sequence constraint can be expressed as $t^f_{ij} \leqslant t^s_{i'j}$, where i and i' are two consecutive tasks in unit j. In the situation depicted in Fig. 1, wherein two products and two equipment units are involved, the following set of constraints hold. It should be noted that this

analysis pertains to zero transfer times.

$t'_1 = t_4$

$t'_3 = t_2$

$t'_1 \leqslant t_2$

$t'_3 \leqslant t_4$

This sequencing is an obvious choice in makespan minimization problem, and the only feasible solution to this set of constraints is $t'_1 = t_4 = t'_3 = t_2$. In essence, Fig. 1 represents this apparent solution. However, this is infeasible in any practical setting because it means cross-transfer. This infeasibility is easily detected and eliminated in the S-graph framework as detailed in the next section.

4. Inherent capabilities of the S-graph framework in addressing the problem

The S-graph framework provides a robust representation of the problem structure in scheduling problems. The detailed description of the S-graph framework has been presented by Sanmartí et al. (2002). However, a brief description of the S-graph is given in this paper to facilitate understanding.

A recipe of a scheduling problem defines the order of tasks, the material transfer between them, and the set of plausible equipment units for each task. The S-graph framework represents this type of information. In an S-graph, a node is assigned to each task (task node) and a node is introduced for each product (product node). It is supposed that at least one equipment unit is available to execute each task. Set *Si* denotes the set of equipment units that can perform task i (represented by task node i). The processing orders of two consecutive tasks are given by a weighted arc (recipe-arc) established between the nodes of tasks. Furthermore, additional recipe-arc is established from the nodes of tasks generating the products to the corresponding product node. The weight of a recipe-arc is specified by the processing time of the task, i.e. the weight of a recipe-arc is positive and not equal to zero. The processing time of a task may be different for different equipment units. In this case, the weight of the recipe-arc is the shortest processing time. The resultant graph is called task network. In order to generate 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. In multiple batch case the recipe-graph can be extended by auxiliary-arcs between the appropriate task nodes of different batches (see details in Holczinger et al., 2002) which accelerate the solution procedure.

An S-graph can represent NIS, ZW and CIS policies appropriately. Let τ_j denote the set of those tasks that follow task j according to the recipe and an equipment unit is assigned to task j and to task k in the order of processing, then, a zero weighted arc (or an arc whose weight is equal to the length of the changeover time if applicable) is established from each element of τ_j to k. This type of arc called schedule-arc. In Fig. 2, the arcs connecting node T2 to node T4 and node T3 to node T4 are the schedule-arcs representing the order of processing in U1. These arcs suggest that task T4 follows task T1 in U1.

In both NIS and ZW cases no intermediate storage is available which implies that an equipment unit is not free after executing a task until the material stored in it has been transferred to the equipment unit of the next task. Schedule-arcs express these constraints properly. In case of the CIS policy, treating common storage as an ordinary processing unit the task of which is simply storage, reduces the problem to a typical NIS case.

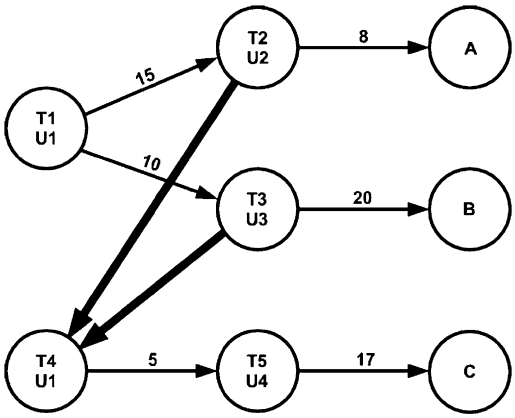

Fig. 2. Schedule-arcs expressing, that U1 is assigned to T1 and then T4.

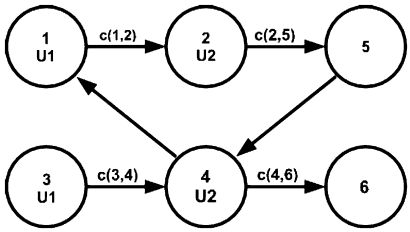

Fig. 3. Cyclic S-graph: corresponding schedule infeasible.

A specific S-graph, schedule-graph, represents a single solution of a scheduling problem. A schedule-graph exists for each feasible schedule. S-graph is called a schedule-graph of recipe-graph if all tasks represented in the recipe-graph have been scheduled by taking into account equipment-task assignments.

As aforementioned, if an arc (recipe-arc, schedule-arc or auxiliary-arc) is established from node i to j, then it is supposed that the task corresponding to node j must start its activity at least $c(i, j)$ time later than the task corresponding to node i started, where $c(i, j)$ denotes the weight of the arc (i, j) and $c(i, j) \geqslant 0$. This is also represented in Fig. 2.

In an S-graph, arcs signify precedence in time, the violation of which always results in infeasibility. Whenever this precedence is not violated, the schedule related to this S-graph is not only feasible from the modelling perspective, but also practically implementable. The unique property of the S-graph that deserves emphasis at this point is that there exists clear congruence between S-graph results and practice. This congruence is not necessarily true for the mathematical programming approaches. The presence of any cycle in an S-graph invariably signifies practical infeasibility. If a cycle contains only schedule-arcs, then it means cross-transfer. If recipe-arcs are also encountered in a cycle, this might mean other form of practical infeasibility, i.e. no corresponding schedule or Gantt chart exists. These two cases are depicted in Figs. 3 and 4. In Fig. 3, the arcs connecting node 5 to 4 and node 4 to 1 are the schedule-arcs. The rest of the arcs in the cycle are recipe-arcs. The existence of a cycle in the S-graph as appears in Fig. 3 suggests an infeasible solution with no corresponding Gantt chart. According to Fig. 3, task 1 should start in U1 at least $c(2,5)$ time units after the commencement of task 2 in U2. This contradicts the recipe which requires that task 1 should precede task 2 by at least $c(1,2)$ time units. Consequently, no schedule can be derived without violating time precedence from the cyclic graph shown in Fig. 3.

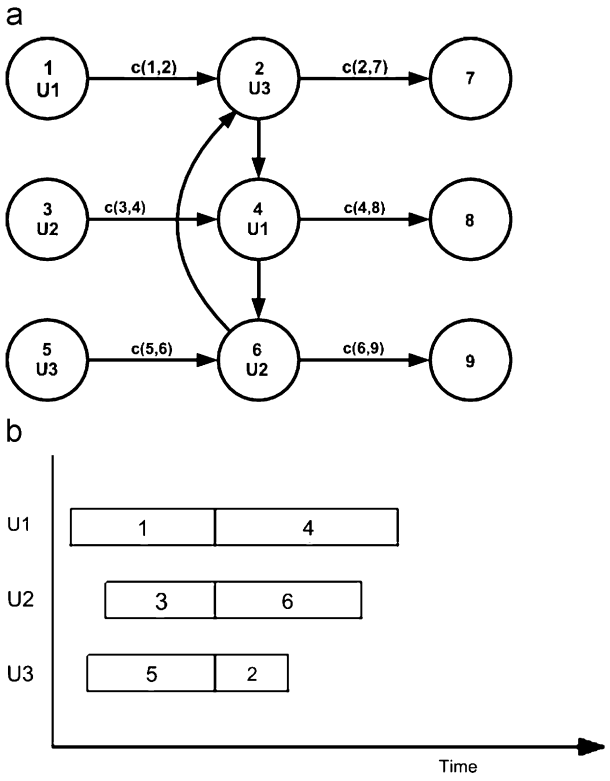

Fig. 4. Cross-transfer for three units and three products: (a) cyclic S-graph, (b) Gantt chart.

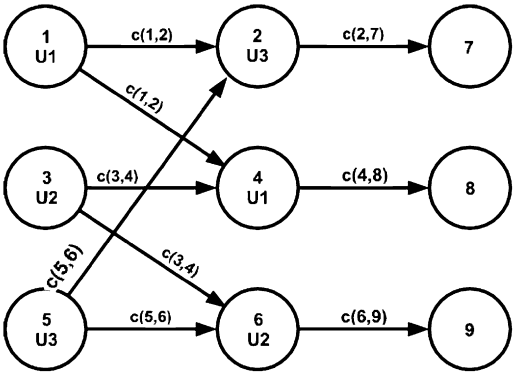

Fig. 5. Traditional graph representation of task sequencing: cross-transfer not detected.

Table 1
Processing times for literature example (Kim et al., 2000; Méndez and Cerdá, 2003)

Products	Units			
	U1	U2	U3	U4
A	15		8	12
B	10	20	5	13
C	20	7	9	
D		7	17	5

In Fig. 4(a), the schedule-arcs connect node 2 to node 4, node 4 to node 6, and node 6 to node 2, thereby forming a cycle as shown. The Gantt chart corresponding to the S-graph in Fig. 4(a) is shown in Fig. 4(b). This Gantt entails cross-transfer as elaborated in Section 2 of this paper, which constitutes infeasibility.

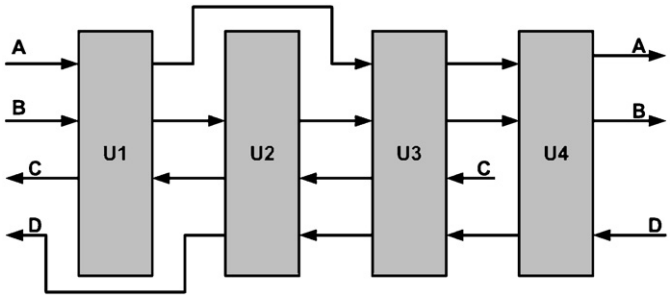

Fig. 6. Sequence of processing units for the literature example.

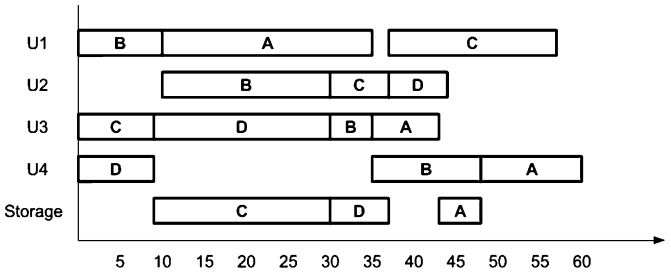

Fig. 7. Supposedly optimal solution obtained by Kim et al. (2000).

The feasibility test of a scheduled (or partially scheduled) S-graph is very effective. Cycles of a directed graph, can be detected easily with a simple cycle search algorithm, which in return is an effective way of recognizing if an S-graph represents an infeasible schedule. Conversely, a non-cyclic S-graph in which the order of precedence of all the tasks is obeyed represents a feasible solution.

It should be emphasized that the ability of the S-graph framework to detect cycles or infeasible schedules is not an inherent feature of all graph-theoretic frameworks, but a specific property of the S-graph. Using a typical graphical representation in which the task sequence in every equipment unit is defined by a set of conjunctive arcs connecting the tasks assigned to the same unit, the schedule shown in Fig. 4(b) would correspond to the graphical representation depicted in Fig. 5. Clearly, there are no cycles in Fig. 5 for recognizing infeasibilities.

Because of its combinatorial characteristics, a branch-and-bound (B&B) procedure may generate the optimal schedule of a scheduling problem, i.e. the S-graph that corresponds to the minimal makespan. The recipe-graph serves as the root of the enumeration tree of the B&B procedure. At any partial problem, one equipment unit is selected and then all child partial problems are generated through the possible assignments of this equipment unit to unscheduled nodes. Every partial problem can be represented by an S-graph and its feasibility can be checked by a cycle search algorithm as mentioned before. The detailed solution procedure and an acceleration technique have been presented by Holczinger et al. (2002) and Sanmartí et al. (2002), respectively.

5. Comparison of MILP and S-graph solutions

In this section of the paper, comparisons between the solutions of mathematical programming methods and the solutions obtained using S-graph framework are presented. Table 1 shows data for the

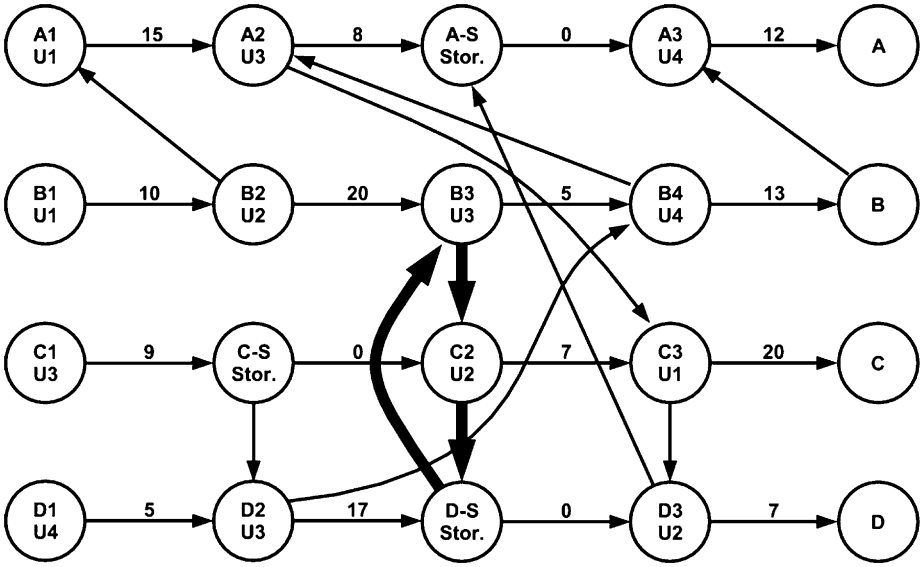

Fig. 8. Cyclic S-graph corresponding to the infeasible schedule in Fig. 7.

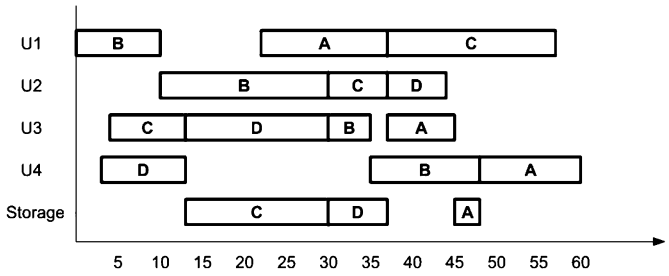

Fig. 9. Supposedly optimal schedule obtained by Méndez and Cerdá (2003).

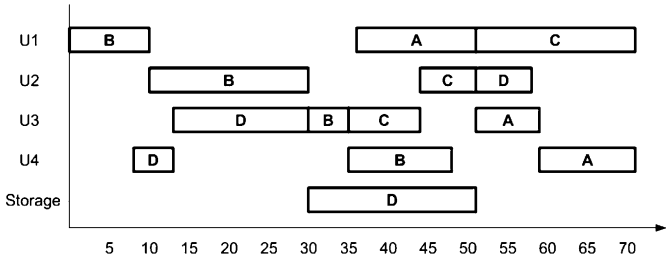

Fig. 10. Globally optimal schedule obtained using the S-graph framework.

literature example taken from Kim et al. (2000). The example involves four products, A, B, C and D that are processed in units U1, U2, U3 and U4 as required by the recipe. The sequence of processing units for each product is shown in Fig. 6.

Depicted in Fig. 7 is the schedule for the literature example obtained using the MILP formulation of Kim et al. (2000). The authors presented this schedule as a globally optimal solution to the problem. In their case, a common storage facility for all the products was introduced into the problem to cast it as a finite intermediate storage (FIS) operational policy. However, it is evident that if the CIS facility is treated as one of the operational units, the task of which is simply storage, the problem reduces into the NIS case. Close scrutiny of Fig. 7 with this understanding in mind immediately confirms that the schedule shown is infeasible from the practical standpoint. In particular, 30 h into the makespan, product B is transferred from U2 to U3 whilst product C is transferred from storage to U2 and product D from U3 to storage at the same time, which is infeasible due to cross-transfer. This observation implies that the reported solution is, in essence, not correct. Fig. 8 shows the S-graph corresponding to Fig. 7 with the cross-transfer, i.e. infeasibility, highlighted in bold. A similar solution, as shown in Fig. 9, was also reported by Méndez and Cerdá (2003) using the MILP formulation.

On the other hand, Fig. 10 shows an optimum schedule obtained using the S-graph framework, whilst Fig. 11 shows the S-graph corresponding to the schedule. No occurrences of cross-transfer are observed in both the schedule and the S-graph, hence a true optimum. Overall, the minimum makespan for this problem is 71 h and not 60 h.

6. Conclusions

The unique strength of the S-graph framework in eliminating the infeasibility, termed cross-transfer, has been justified in detail. In addition, the inherent weakness of the MILP approaches in isolating this infeasibility, which occurs in NIS and ZW operational policies, has been fully demonstrated. This drawback of the MILP approaches has gone unnoticed in the past, with infeasible solutions reported as globally optimal.

Acknowledgment

This work is supported by the South African/Hungarian bilateral agreement on science and technology under Grant Number 2072845 (South Africa) and ZA-13/2006 (Hungary).

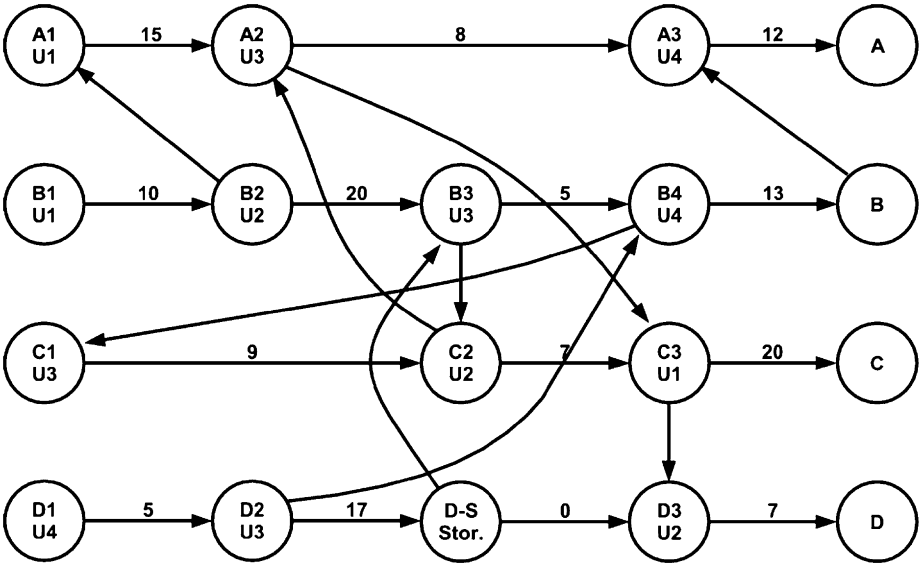

Fig. 11. S-graph without cycle corresponding to the optimum schedule in Fig. 10.

References

Castro, P.M., Grossmann, I.E., 2006. An efficient MILP model for the short-term scheduling of single stage batch plants. Computers and Chemical Engineering 30, 1003–1018.

Castro, P.M, Barbosa-Póvoa, A.P.F.D., Matos, H., 2001. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. Industrial and Engineering Chemistry Research 40, 2059–2068.

Floudas, C.A., Lin, X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers and Chemical Engineering 28, 2109–2129.

Holczinger, T., Romero, J., Puigjaner, L., Friedler, F., 2002. Scheduling of multipurpose of batch processes with multiple batches of the products. Hungarian Journal of Industrial Chemistry 30, 305–312.

Holczinger, T., Majozi, T., Hegyhati, M., Friedler, F., 2007. An automated algorithm for throughput maximization under fixed time horizon in multipurpose batch plants: S-graph approach. European Symposium on Computer Aided Process Engineering (ESCAPE), Bucharest, Romania, May 27–30 2007. ISBN 978-0-444-53157-5.

Ierapetritou, M.G., Floudas, C.A., 1998. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Industrial and Engineering Chemistry Research 37, 4341–4359.

Kim, S.B., Lee, H.-K., Lee, I.-B., Lee, E.S., Lee, B., 2000. Scheduling of non-sequential multipurpose batch processes under finite intermediate storage policy. Computers and Chemical Engineering 24, 1603–1610.

Kondili, E., Pantelides, C.C., Sargent, R.W.H., 1993. A general algorithm for short-term scheduling of batch operations. I. MILP formulation. Computers and Chemical Engineering 17 (2), 211–227.

Majozi, T., Friedler, F., 2006. Maximization of throughput in a multipurpose batch punder fixed time horizon: S-graph approach. Industrial and Engineering Chemistry Research 45 (20), 6713–6720.

Majozi, T., Zhu, X.X., 2001. A novel continuous-time MILP formulation for multipurpose batch plants. 1. Short-term scheduling. Industrial and Engineering Chemistry Research 40 (25), 5935–5949.

Méndez, C.A., Cerdá, J., 2003. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. Optical Engineering 4, 7–22.

Romero, J., Puigjaner, L., Holczinger, T., Friedler, F., 2004. Scheduling intermediate storage multipurpose batch plants using the S-graph. A.I.Ch.E. Journal 50 (2), 403–417.

Sanmartí, E., Friedler, F., Puigjaner, L., 1998. Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation. Computers and Chemical Engineering 22 (Suppl.), S847–S850.

Sanmartí, E., Holczinger, T., Puigjaner, L., Friedler, F., 2002. Combinatorial framework for effective scheduling of multipurpose batch plants. A.I.Ch.E Journal 48 (11), 2557.

Schilling, G., Pantelides, C.C., 1996. A simple continuous-time process scheduling formulation and a novel Solution algorithm. Computers and Chemical Engineering 20 (Suppl.), S1221–S1226.