Efficient Optimization Algorithms for Zero Wait Scheduling of Multiproduct Batch Plants

by

Deepak B. Birewar, Ignacio E. Grossmann

EDRC 06-61-89
Carnegie Mellon University

EFFICIENT OPTIMIZATION ALGORITHMS FOR ZERO WAIT SCHEDULING OF MULTIPRODUCT BATCH PLANTS

Deepak B. Birewar

and

Ignacio E. Grossmann*

Department of Chemical Engineering
Carnegie-Mellon University
Pittsburgh. PA 15213

November 1988/Rev. June 1989

*** Author to whom correspondence should be addressed**

Abstract

This paper deals with the scheduling of multiproduct batch plants consisting of one unit per stage and involving the production of large number of batches of relatively few products (e. g. 10-50) with the Zero-Wait policy. An aggregation procedure based on a compact linear programming model is proposed for determining the minimum cycle time and the family of schedules with the optimal cycle time. It is shown that schedules can be easily derived from the LP solution using an aggregated graph representation. It is also shown that using this representation an approximate MILP model that yields very good near optimal solutions can be derived for the makespan minimization problem and for which very tight bounds can be established. The insights gained with this approximate method are used to formulate a rigorous MILP model which can be solved as an LP in most cases. The effectiveness of the proposed procedures is illustrated with several example problems.

Introduction

This paper deals with the scheduling of multlproduct batch plants consisting of a sequence of stages, where each stage involves one processing unit. The Zero Wait policy I ZW 1 for scheduling is assumed which consists of transferring a batch to the next stage as soon as the processing is completed in the current stage. This type policy is often used in batch processes. It is also assumed in this problem that the number of batches to be produced for each product that is specified is large, and that all products follow the same processing sequence. Fixed processing times and clean-up times between different products are also given. The objective is then to find a sequence of batches of products that leads to the minimum or near minimum makespan (i. e. total time).

Hie above scheduling problem is equivalent to the *Jlowshop problem* in Operations Research and has been extensively analyzed in the past; e. g. see Graham et al (1979), Graves (1981) and Baker (1975). Rigorous optimization of this problem has been shown to be NP-hard; i.e. the computational time required to obtain a globally optimal solution may increase exponentially in the number of batches to be scheduled (Garey et al, 1976). For instance, when this problem is posed as an asymmetric Travelling Salesman Problem [TSP] (Wismer, (1972), Reddi, (1972), Gupta, (1976)) its size increases exponentially with the number of batches to be scheduled due to the required addition of subtour elimination constraints which ensure that sequences without any *subtours axe* obtained. Recently, Pekny and Miller (1988) developed a parallel branch and bound method for the asymmetric TSP problem with which they were able to solve to optimality problems with up to 3000 batches. However, apart from the fact that this method is computationally intensive and requires parallel computers, it does not exploit the fact that large number of batches often belong to much fewer number of products. This fact can be quite crucial for developing an efficient method as will be seen in this paper.

On other hand there are approximate methods consisting of *heuristics* or *rules of thumb* which provide near optimal solutions with small computational expense (e. g.

see Campbell et al. 1970, Dannenbring, 1977, Panwalker and Iskander, 1977, Ku and Karimi, 1986, Kuriyan and Reklaitis, 1989). While these methods are fast, their disadvantage is that they cannot guarantee optimality. Thus, they can sometimes yield solutions which are relatively far from the global optimum. More extensive review of the scheduling problem in batch processes is provided by Ku et al(1987).

Birewar and Grossmann (1989) have recently considered the simultaneous design and scheduling of multiproduct plants, where these authors showed that the scheduling problem in this problem can be simplified to large extent by choosing an alternative objective function. In particular, it was shown that the objective of minimizing Cycle Time [CT] is a suitable alternative to the makespan minimization, especially when the time horizons concerned are long. In that paper it was shown that this leads to a 0-1 minimax assignment formulation, which was aggregated in the form linear constraints for the design problem.

This paper will show that the linear constraints for scheduling in the design problem by Birewar and Grossmann (1989), can be used as a basis to develop a very compact LP formulation for the minimization of cycle time in multiproduct batch plants. Since the key idea is to aggregate the batches into the space of products, this formulation is especially suitable for problems involving many batches belonging to relatively few products. The solution to the proposed LP model, which may require a simple procedure for eliminating subcycles, provides a *family of schedules* rather than a single solution. It is shown that one or several schedules among this family of solutions can be easily generated with a *graph* analysis which can be implemented very efficiently as an algorithmic method that is suited for automation. Using as a basis a schedule with minimum cycle time, it is shown how to derive with an MILP model a schedule with minimum or near minimum makespan for which a very tight lower bound can be computed. It is also shown that the rigorous solution to the makespan minimization problem can be obtained with a novel MILP formulation that can be solved as an LP in most instances. Several numerical examples are presented to compare the performance of the approximate and rigorous algorithms.

Problem Statement

The problem that will be initially addressed in this paper can be stated as follows:

Given is a multiproduct batch processing plant with M stages each with one processing unit per stage. A total of N batches belonging to N_p different products are to be produced using the Zero Wait [ZW 1 scheduling policy. The number of batches r^, for each product i, i = 1. . . . $.N_{pt}$ is specified, as well as fixed processing times ty for each stage j, j = 1. . . . .M, and fixed clean-up times CLN^ for successive products i, k in stage j. The goal is then to find a schedule with minimum cycle time.

As shown in the following sections, this problem can be effectively tackled with an LP model that is coupled with a graph analysis procedure. Furthermore, its optimal solution can be used as a basis to develop very good near optimal solutions for the makespan minimization problem, and to also motivate the development of a rigorous model for this problem.

Minimization of Cycle Time

As described by Birewar and Grossmann (1989), in the ZW policy some idle time may be forced to exist between any two batches due to the rigid nature of this policy. Let SL^ denote the minimum forced idle time between the batches of product i and k in stage j. These slack times can be pre-computed for each pair of products as shown in the example of Figure 1 with non-zero clean-up times. A systematic procedure for computing these slacks is presented in Appendix I, including the case when nonzero clean-up times are specified (see Birewar and Grossmann, 1989). The computing effort required to calculate these slack times for all possible combinations of the products is very small.

For the ZW policy the optimal cycle time [CT J will in general not be equal to the minimum cycle time of the Unlimited Intermediate Storage [UIS J policy (see Birewar and Grossmann, 1989):

$$CT = {}_{,}JTM \left(\sum_{I \gg 1}^{N_p} t: \; tg \right) \tag{1}$$

Unlike the UIS case, in ZW policy all the stages including the ones that define the bottleneck will typically exhibit several non-zero idle times [SL^j]. Thus, the analytical expression in (1), that defines the optimal processing time in case of the UIS policy, will provide only a lower bound for the ZW policy. To account for the impact of the slacks and to aggregate the number of batches in terms of products, the entire cyclic sequence will be viewed to be consisting of pairs of batches of various products. For example the sequence ^{M} A - A - A - B - C - " can be viewed as consisting of pairs A-A, A-A, A-B, B-C and C-A. For each possible pair of products, let NPRS^ be the actual number times that product i is followed by product k (i,k = 1.....N_p) in the entire cyclic schedule. For instance in the previous example, NPRS^ = 2, NPRS^ = 1, NPRSgc = 1, NPRS^ = 1. As every product is manufactured exactly nj times, it will appear r^ times in the first place and n_t times in the second place in the pairs (i,k) of products that are manufactured during the production run. Therefore, the following two assignment constraints apply (see Birewar and Grossmann, 1989, for more details):

$$\sum_{k=1}^{N_p} NPRS_{ik} = n_i \qquad i=1...N_p \tag{2}$$

$$\sum_{i=1} NPRS\& = n_k \qquad k=1,...,N_p \tag{3}$$

The cycle times of each stage CTj, j = 1 ... M, will consist of the batch processing times of all the products in that stage as in equation (1), plus the contribution of slack times [SLjig] existing between the batches of various products.

$$CT_j = \sum_{i=1}^{N_p} {}''i \;{}'y + \sum_{i=1}^{N_p} \sum_{k=1}^{N_p} NPRS_{*} \, SL\% \qquad J\text{-}U\text{-}M \tag{4}$$

where the overall cycle time is

$CT \geq CT_j \qquad j = 1..M$

Minimization of the cycle time CT, subject to the constraints (2) - (4), leads then to the following linear programming model LP1 :

$$\min \ CT$$

$$\text{s.t.} \quad \sum_{k=1} NPRS_{ik} = n_i \qquad i=1..N_p$$

$$\sum_{i=1}^{N_p} NPRS_{ik} = n_k \qquad k=1..N_p \qquad (LP1)$$

CT 2 ", £ «i·· 'y +, X I ^4 ^{SL}*j ST 1=1 hs\ N N> j=1..M

$NPRS_{U} \leq n_{(} - 1 \qquad i=1..N_p$

$CT \geq 0, \quad NPRS_{ik} \geq 0, \quad i, k = 1..N_p$.

where the fourth constraint is optional but has been included to reduce the possible occurrence of subcycles as will be explained later on in the paper.

The linear program LP1 is very compact as it involves only Nj*, + 1 continuous variables and 3N_P + M constraints. Thus, the size of this problem remains unchanged with the number of batches for a fixed number of products N_p.

The solution to LP1 provides the values of NPRS^ for various combinations of products i and k, but due to the aggregation in terms of products, this by itself does not define any specific schedule. Therefore the solution to LP1 corresponds to an implicit representation of a *family of schedules* that exhibit *minimum Cycle Time.* Any of these solutions can be chosen based on a secondary criterion, like for instance ease of implementation of the schedule. Also the schedule that results from this LP is in the form of a single cycle or several subcycles for the N batches (see section on subcycles). When a single cycle is obtained, there will be a total of N links among the N_p products. This cycle can be broken so as to select a sequence with minimum makespan among the N possible sequences that result from breaking one of the N links in the optimal cycle. Before discussing further this procedure, and some properties of the LP model, it is first useful to present an example of this formulation and show how it can be related to a *graph representation* for deriving a schedule.

Graph Representation

A graph representation is presented in this section that helps to visualize and select a schedule from the solution of LP1. To facilitate the understanding of this representation consider the modest sized scheduling example 1 consisting of 30 batches (N = 30) belonging to six different products A to F (N_p = 6). The multiproduct batch plant consists of four processing stages (M = 4) and a schedule is to be determined to manufacture 5 batches of A, 7 of B_f 3 of C_f 5 of D, 4 of E and 6 batches of F. The batch processing times for products A - F are given in Table I(a). There are no cleanup times and hence the slack times SL^j between the batches of product i followed by product k in stage J are given as indicated in Table I(b).

Table n(a) lists the results obtained from the LP scheduling model which corresponds to a cycle time of 140 hrs. Note that each entry in this table corresponds to a value of the variable NPRS^, i = l...N_p. k = l...N_p. This solution indicates that the optimal schedule for the production is with Mixed Product Campaigns [MPC] since several product pairs NPRS^ are greater than one for i * k.

The solution for Single Product Campaigns [SPC] can be obtained by specifying that the optimal sequence will consist of one campaign per product; in other words that there will be nj - 1 pairs of product i in the sequence. This condition can be expressed as:

$$NPRS_{U} = A; - 1 \qquad\qquad i=l,..JV, \qquad\qquad (5)$$

Imposing this constraint to problem LP1 defines LP2 :

min CT

$$
\begin{array}{llll}
S.t. & \sum_{k=1} NPRS_{ik} = n_i & i=1..N_p & \\
 & \sum_{i=1}^{N_p} NPRS_{ik} = n_k & k=1..N_p & \\
 & & & (LP2) \\
 & CT \; * \; \sum_{i=1}^{N_p} n_i \; \wedge \; + \; \sum_{i=1}^{N_p} \sum_{k=1}^{N_p} \text{AffWS}^\wedge \; 5L^\wedge. & JM\backslash\%...M & \\
 & NPRS_{U} = n_4 - 1 & /=1,...JV, & \\
\end{array}
$$

cr ^ a #/>*$* >_ o, /, * = L.JV^ .

which can be used to find the minimum cycle time for production with Single Product Campaigns. The optimal solution to LP2 is listed in Table II(b) and has a cycle time of 172 hrs which is 18.6% longer than the CT for the production with the MPC (140 hrs.). These problems were solved on MIcrovax n using the LP solver ZOOM (Marsten, 1986) through the modelling system GAMS (Kendrick and Meeraus, 1985). Problem LP1 for MPC and LP2 for SPC consisted of 37 variables and 22 constraints. They both required about 5 seconds of total computational time, respectively. (The total computational time includes the time for calculating the slacks SL^ for all possible product-pair combinations using GAMS, and then solving the problem using ZOOM).

The two solutions In Table II represent a *family of schedules* that exhibit minimum cycle time [CT]. As seen in Figures 2(a) and 2(b) these solutions can be represented with a directed graph where the nodes correspond to the products and the directed arcs are defined by the non-zero values of the variables NPRS^ in Table II. Note that In this graph the numerical values of NPRS^ can be Interpreted as the number of links between the nodes of product I and k.

Since the number above the arcs that start and end with the same product denote the number of batches of that product which will be produced in Single Product Campaigns, the actual schedule for the SPCs can be derived trivially from Figure 2(a). For example Figure 2(a) denotes a cycle A - A - A - A - A - E - E - E - E - C

- C - C - D - D - D - D - D - B - B - B - B - B - B - B - F - F - F - F - F - F - . Here the makespan will differ depending on where the cycle is *broken* to produce a sequence. For example breaking the cycle between the campaigns of products F and A results in a successive campaigns of products A followed by E, followed by C, D, B and F in that order requiring the makespan of 184 hrs. The optimal makespan for this particular cycle however results from breaking the cycle at a link between products E and C. This sequence C - C - C - D - D - D - D - D - B - B - B - B - B - B - B - F - F - F - F - F - F - - A - A - A - A - A - E - E - E - E - has a makespan of 177 hrs. and can also be shown to be the global solution for the makespan problem as shown later in the paper.

One of the cyclic schedules for the MPCs in Figure 2(b) can be derived quite easily from the graph representation as shown in Figures 3(a) to 3(e). Each directed arc in the graph shows the pair of products that will exist in the optimal schedule and the number above the arc specifies the number of times that pair will occur in the optimal sequence. The graph in Figure 203) is an aggregate graph that can be decomposed into various campaigns to yield a final schedule.

Firstly, figure 3(a) shows that 2 batches of E will be produced consecutively. Given that the remaining arcs in the graph interconnect different products, there are in general different ways to derive a schedule. One possibility is as follows. Select a node with an arc involving fewest number of successive batches and trace a path until returning to that node in order to complete a loop. The loop with the corresponding number of successive batches is removed from the graph and defines a campaign. The procedure is repeated until the nodes and the arcs in the graph are exhausted. The loops (i. e. campaigns) are then merged into a single cycle. Applying the above procedure. Figure 3(b) shows that a campaign containing 1 successive batch of products D - B - will be part of the optimal cycle. Figure 3(c) shows that a campaign of 1 successive batch of products B - F - D - will be part of the optimal cycle too. Finally. Figure 3(d) shows that a campaign of 2 successive batches of products A - E - B - F and a campaign of 3 successive batches of products A - C - D - B - F will also be part of the optimal cycle. These campaigns are then combined as shown in Figure 3(e) so that the total cycle time is still equal to the solution of LP1 (140 Hrs.). Any two campaigns can be Joined only if both contain a common product. Refer to Figure 3(e). Campaigns D - B - and F - D - B - can be Joined as shown, at a link between B - D and B - F, and so on. The final optimal cycle with CT of 140 Hrs is shown in Figure 3(e). A more systematic procedure based on matrix representation of the solution, that can be easily automated is presented in the next section before discussing some properties of the LP model and ways to overcome some potential difficulties regarding subcycles. Also, it will be later shown how to break cycles to determine an actual sequence.

An Algorithm for Generation of Cyclic Schedules

A systematic method to generate cyclic schedules from the solution of LP1 or LP2 is presented here. The scheduling problem discussed in the previous section is used to explain the implementation of this method. Tables m(a) to ni(e) show the gradual evolution of the schedule to its final form starting from the LP solution. First the rules to systematically derive the schedule from the LP solution are stated, followed by the description of their application to the above stated example problem. The method consists of 4 major rules as stated below. These rules operate on the solution represented in matrix form as shown In Table H(a) or Ilfb).

- Rule #1: Mark all the diagonal entries in the matrix. They represent the SPCs in the final schedule. Delete these entries.

- Rule #2: In the resultant matrix, choose one of the *smallest* entries. Mark the cell. Determine the product name of that column. Go to the row corresponding to that product. Pick up an entry (If there is more than one entry in this column see Rule #3). Mark the cell that is chosen. Repeat the procedure till the loop is completed, i. e. the entry that was chosen first is picked again. Now delete the smallest entry from all the marked cells. The elements of this loop represent the product-sequences that will exist in the final schedule. The smallest entiy represents the number of times this product-sequence will exist in the final schedule. Apply Rule #2 until all the entries in the matrix are deleted.

- Rule #3:
  - (a) If there is more than one nonzero entiy in the row concerned, choose the entry that will complete the present loop.
  - (b) If none of the entries in that row will complete the loop then pick

up the *smallest* entry.

- Rule #4: The application of Rules #1 and #2 will identify various campaigns that form the optimal sequence. These campaigns can be merged together to form one single cycle as follows. Any two subcycles can be joined at the point where they have a common product. If this procedure fails to merge all the campaigns into one single cyclic schedule, then it means that the solution contains subcycles similar to the subtours encountered in the Travelling Salesman Problem. Refer to the section on subcycles for schemes to eliminate such subcycles.

The final schedule for the example problem stated in the previous section was derived using these rules as follows(see also Figs. 3(a) - 3(d) for comparison):

Rule #1 is applied first. The diagonal entry in row and column of product E is marked first (Table III(a)). It represents single product campaign of E containing 2 batches. Deleting this entry results in Table III(b). As there are no more diagonal entries, Rule #2 is applied. Entry in row of product B and column D is chosen (ROW_B - COL_D). As the column belongs to product D, go to the row of product D. This row contains only one entry in column B. Pick this entry. Go to row of product B. This was the row we started with. Hence the loop is complete. The smallest entry is one. Hence delete 1 from entries ROW_B - COL_D and ROW_D - COL_B. This operation is shown in Table III(b). The same procedure is applied to the resultant table (Table III(c)). This contains a cycle F-D-B-. Now entry ROW_A - COL_E is chosen for the application of Rule #2. After picking entries ROW_E - COL_B, ROW_B - COL_F, ROW_F - COL_A we again come back to row A. Row A contains entries in columns C and E. The entry in the column E is chosen as the one that will close the loop (Rule 3(a)). The procedure is shown in Table III(d). Application of the Rule #2 again, as shown in Table III(e) exhausts all the batches.

Thus, Rules #1 through Rule #3 are used to *decompose* the solution to the LP1 into various campaigns. These campaigns can be merged into a single cycle using Rule #4. The result of applying Rule #4 is shown in Figure 3(e). The LP1, thus yields the solution:

E-E-E-B-F-A-E-B-F-A-C-D-B-F-A-C-D-B-F-A-C-D-B-F-D-B-D-B-F-A-.

with to. optimal Cycle Ttoe of 140 his. Note that this Is the same sequence that was obtained In Figure 3(e) with the graph representation.

Integrality of Solutions

There are two potential limitations in the proposed linear programming formulations for scheduling. One Is that the solution to LP1 or LP2 may In principle yield non-integer values for the variables NPPS^. However, when only one of the stages is constraining the Cycle Time, It can be proved that the values of NPRS^. will be integers by straightforward extension of the proof gjven by Birewar and Grossmann in their paper on Design and Scheduling of multiproduct batch plants (1989) . In this case the argument Is simply that when one stage limits the cycle time, problem LP1 and LP2 can be rewritten as assignment problems which have unimodular matrices. Moreover, since the rigjit hand sides in (2) and (3), the number of batches of each product, are integer, sufficient conditions are satisfied to ensure the integrality of variables NPRS^. When two or more stages are constraining, again extending the proof by Birewar and Grossmann (1989), it can be shown that the objective function value of relaxed solution UP and of the integer solution will be same. i. e. It will exhibit zero *gap*. This implies that since the optimal value of the objective function is known, then in the event that non-integer solutions arise, the computational effort with branch and bound to find the optimal solution would be minimal. Given this property of zero gap, however, one could also resort to a simple trial and error rounding scheme to obtain an integer solution that has the same cycle time. It is important to note, however, that among all the examples that we have so far solved, *none* exhibited non-integer values for NPRS^ while solving the relaxed UP. Thus, there was no need to resort to branch and bound enumeration or rounding procedure.

Subcycles

The other potential problem in LP1 and LP2 is that the LP solution might ei subcycles. As an example consider that the solution predicted by LP1 is as she Figure 4(a), where there are two subcycles (A-B-C-D and E-F). Since the soluti

subcycles does not satisfy all the constr
involvin

ete cyclic
the cyclic
which can be detected with the
Appendix II. In most of the times are due to some Single
Product Campaigns which may be present in the solutions. This situation can be avoided by adding a constraint that requires the total number of pairs containing both the elements of the same product should be less than the number of batches of that product,

$$AP_{i} \le n_i - 1 \qquad i = 1,\ldots,N_p \qquad (6)$$

It is for this reason is that constraint (6) was added to LP1 as it prevents certain types of subcycles in the solution without excluding the optimal cycle time solutions.

In the case, however, when subcycles are obtained in LP1 or with LP2, the procedure presented below can be used to obtain solutions without any subcycles. The following group of *subcycle elimination constraints* will have to be satisfied in addition to the formulation LP1 or LP2 if no subcycles are to be present in the solution to the scheduling problem. Subcycle elimination constraints are similar to the *subtour elimination constraints* used in the Travelling Salesman Problem (Garfinkel and Nemhauser, 1972) . Let Q be a subset of products such that the cardinality of Q is strictly less than N_p and it is not an empty set. Then

$$\sum_{i \in Q} \sum_{k \in Q'} NPRS_{ik} \ge 1 , \quad Q \subset Q_p, \quad Q \ne \phi, \quad |Q| + |Q'| = N_p \qquad (7)$$

where Q_p is the set of all N_p products and Q' the complement of Q. Adding (7) to LP1 or LP2 yields a linear programs that will ensure that the optimal solution does not contain any subcycles. This LP, however, contains $2^{N_p} - 2$ more constraints than LP1 and LP2. For an example consider the graph representation of a sequence containing two subcycles as shown in Figure 4(a). The corresponding sequence is shown is Figure 4(c). Use of the subtour elimination constraints of (7) yields a solution as shown in igure 4(b), that does not contain any subcycles. Note that one of the arcs between products B and C was broken and new arcs were formed between products B and F, and E and C. The new sequence is shown in Figure 4(d).

Addition of the 2^Np - 2 subcycle elimination constraints in (7) clearly represents a great reduction in size over the conventional asymmetric TSP formulation for scheduling (Gupta, 1972) where the number of subcycles elimination constraints is 2^N - 2, where N is the total number of batches. Nevertheless, the formulation can yield a large linear program if the number of products itself Is very high. To circumvent this difficulty. Instead of including all the subtour elimination constraints for all non-empty proper subsets Q of Q_p, consider only the constraints for the proper subsets Q corresponding to sets of products present in the various subcycles in the solution to LP1 or LP2. That is.

$$\sum_{i \in Q} \sum_{k \in Q'} NPRS* \geq 1 , \qquad Q = S_1, S_2, \ldots S_{N_s} \quad |Q| + |Q'| = N_p \qquad (8)$$

where S!. . . . .S^ are the sets of products belonging to the N_s subcycles. Adding (8) to LP1 or LP2 defines LP3 and LP4, respectively. As an example, in Fig 4(a), there are two subsets Q; Q = Sj = (A, B. C, D), Q = S2 = {E. F).

The linear program LP3 then ensures that the subcycles 1.2..N_S do not occur *[S_x* and S_2 in the example shown in Figure 41. If the solution to LP3 does not introduce new subcycles. the procedure is stopped. Otherwise, constraints in (8) for the new subcycles must be added to LP3 and solved again. This procedure is repeated until the optimal LP solution contains no subcycles. In our experience with the constraints (6) in LP1 the solution often does not contain subcycles (see Table VI). When this is the case, however, typically only one iteration is required with the addition of the constraints in (8) to eliminate subcycles.

Given that a solution with integer number of batches and without any subcycles is obtained, tight *lower* and *upper* bounds for the makespan minimization can be obtained as shown in the next section.

Approximate Method for Makespan Minimization

Selecting a sequence with smallest makespan within the minimum cycle solution, will often lead to an optimal or very near optimal solution of the makespan minimization problem. This result can be verified with a tight lower bound on the minimum makespan as shown below.

The optimal Cycle Time CT° obtained from the solution to LP1 or LP2. and possibly with the addition of the constraints in (7) or (8). corresponds to a lower bound of the total makespan. An improved lower bound can be obtained by choosing the smallest possible *head* and *tail* in any given schedule and by excluding the effect of largest slack time corresponding to any possible pair of products. Refer to Figure 5 for a brief explanation of the terms *head* and *tail* of a stage.

It is clear from Figure 5 that the total makespan for any sequence will consists of three parts. First is the maximum of the total processing time required in various stages (SPT^{11}), head and tail being the other two parts. Thus we can find a lower bound on the total makespan by subtracting from optimal cycle time the worst possible slack for any pair of products and adding the smallest head and the smallest tail. The worst or highest slack is subtracted from the cycle time to obtain a lower bound on SPT***. The lower bound on the makespan is then given by the following equation,**

$$MS^L = \max_{j=1..M} \left(CT^G - \max_{(i,k)\in S} (SL_{ikj}) + \min_k \left(\sum_{j'<j} t_{kj'}\right) + \min_i \left(\sum_{j'>j} t_{ij'}\right)\right) \qquad (9)$$

where S is the set of all product pairs (i,k) existing in the sequence with optimal cycle time (CT°).

An upper bound for the total makespan can be obtained by selecting a sequence with the smallest makespan contained in the minimum cycle time solution. This upper bound will be the sum of the optimal cycle time containing no subcycles (CT°) and the combination of the head and tail minus the slack of the pair corresponding to the link that is broken to form the sequence. Note that the tail corresponds to the first

product in the broken link and the head corresponds to the last (or second) product in the broken link.

The link to be broken is chosen using the binary variable YP^. Since only one of the links in the optimal cycle is to be broken to form a string, the following constraint applies :

$$\sum_{(i,k)} \sum_{\in S^f} ypit \; = \; l \qquad \text{(io)}$$

where S^f is the set of all product-pairs (i,k) present in the minimum cycle time solution.

And the makespan value for sequence will be given by.

$$MS^U \;\geq\; CT_j^G + \sum_{(i,k)\in S^f} \left(\left(\sum_{j'<j} t_{ij'} + \sum_{j'>j} t_{ij'} \right) - SL_{ikj} \right) YP_{ik} \qquad j=1..M \quad (11)$$

The upper bound of the makespan problem, which defines an actual schedule, can then be determined from the following MILP problem:

$$\begin{array}{ll} & min \;\; MS^U \\ s.t. & constraints \;\; (10) \;\; and \;\; (11) \qquad\qquad (MILPl) \\ & MS^U \;\; Z \;\; 0 \;\; ; \quad YP\& \; = \; 0,1 \;\; , \quad (ijt) \;\; e \;\; ? \end{array}$$

It should be noted that in general the percentage difference between the lower and the upper bounds as given by (9) and MILPl will be veiy small. This will be especially true when the number of batches involved is relatively large as then the difference between cycle time and the makespan will be small. This MILPl can be solved in most cases as an UP where the 0-1 variables are relaxed as continuous variables. Also note that in (MILPl) the number of 0-1 variables is always significantly smaller than Nj*. A detailed description of the scheduling algorithm for this formulation is given in Appendix III.

For the example 1 presented in the graph representation section containing 30

batches of 6 products, and with MPC's, the bounds as given by (9) and MILP1 were determined. Both the upper and lower bounds for the makespan of this problem were found out to be 145 Hrs. (Note that the cycle time CT° is 140 Hrs.), Since the upper and lower bounds coincide the solution represents the globally optimal makespan. This sequence is obtained by breaking the optimal cycle in Figure 3(e) at the link between products E and B. Figure 6(a) shows this schedule in the Gantt chart format. Figure 6(b) shows the schedule with optimal makespan for the SPCs. This again is a globally optimal solution to the SPC makespan problem as both the lower and upper bounds, calculated using (9) and MILP1 respectively, coincide (177 Hrs.). (Note that the cycle time CT° is 172 Hrs.).

Rigorous Method for Makespan Minimization

In the previous section an upper bound to the makespan was obtained by selecting from among those sequences that have minimum cycle time. In this section **this restriction will be removed so that the rigorous minimum makespan solution will be selected from among all possible sequences. The rigorous solution can be found using an MILP formulation as follows.**

Refer to LP1 presented earlier. The variables NPRS^, which must satisfy the following assignment constraints.

$$\sum_{k=1} NPRS\& = n, \qquad M..M, \qquad (12)$$

$$\sum_{i=1}^{N_p} NPRS\& = n_k \qquad k=l...N_p \qquad (13)$$

will in general define a cycle.

The optimal sequence can be obtained by breaking one of the links between the successive batches of products in the cycle. Let the binary variable Y^ define the link that is broken. If the link between batches of products i and k is broken, then the variable Y^ is equal to one, otherwise it is zero. Exactly one of the pairs that exist in the cycle is to be selected for the link to be broken. Hence the following constraint

must be satisfied:

$$\sum_{i=1}\sum_{k=1} {}^{r}» \quad '' \quad 1 \qquad\qquad 04)$$

Also a link can be broken only if the corresponding pair exists in the cycle. Therefore, the following constraint must be satisfied:

$$Y\& \quad «S \quad NPRS\& \qquad\qquad ijk = 1.JV, \qquad (15)$$

The makespan MS will be consist of head, tail and SPT***, the maximum of the total processing time required in various stages (see Figure 5). SPT*** will be equal to the the summation of the various processing times and the slacks that exist between various product-pairs minus the slack existing between the link (or pair) to be broken to form the sequence from the cycle. The tail will correspond to the first product and the head will correspond to the second (or last) product in the pair corresponding to the broken link. The makespan MS is then defined by the following constraint,

$$MS \quad 2 \quad | \quad n_i \; t_{ij} \; + \quad | \sum_{k=1}^{N_p} NPRS_{ik} \; SL_{ikj} \; +$$

$$\sum_{i=1}^{N_p} \sum_{k=1}^{N_p} \left(\left(\sum_{j'<j} t_{ij'} + \sum_{j'>j} t_{kj'} \right) - SL_{ikj} \right) YP_{ik} \qquad j=1...M \qquad (16)$$

The global optimum makespan can then be obtained from the following MILP:

$$min\ MS$$

$$s.t.\ \ Constraints\ (12),\ (13),\ (14),\ (15),\ (16),\ (6) \qquad\qquad (A//LP2)$$

$$NPRSfr \quad 2> \quad 0, \qquad Y_{ik} = 0,1 \;, \qquad i,\ k = l...N_{pf} \qquad MS \quad 2\geq \quad 0.$$

for which it is convenient to consider the addition of constraint (6) to reduce the possibility of occurrence of subcycles. Also, as in problem MDLP1$_9$ MILP2 can be solved as a relaxed LP in most cases.

Note that this formulation yields directly the information on the minimum makespan sequence since it indicates the number of product pairs ($NPRS_{ik}$) as well as the location of the link (Y_{ik}) that is to be broken in the cycle to obtain the optimal sequence. It is of course possible to obtain subcycles with this formulation in which case a procedure similar as described previously in the section on remarks can be applied.

Addition of constraint (7) to MILP2 yields an MILP problem that will ensure that the optimal solution does not contain any subcycles. However, addition of constraint (7) means 2^{N_P} - 2 more constraints than MILP2. Hence, constraints similar to (8) can be used as described previously to eliminate the subcycles. Addition of constraint (8) to MILP2 yields problem MILP3. Appendix IV summarizes the main steps for the rigorous algorithm.

As a final comment, note that while problem MILP2 has the advantage of yielding a rigorous solution, its size is larger than problem MILP1. In particular, problem MILP2 involves N_P^2 0-1 variables while problem MILP1 will usually have significantly fewer 0-1 variables.

Examples

The effectiveness of the proposed methods will be illustrated with the several examples presented in this section. It should be noted that all the models presented here have been implemented in the modelling system GAMS (Meeraus and Brooke, 1985).

Consider the rather large example 2 consisting of 1059 batches (N = 1059) belonging to a total of 20 products (N_P = 20). The multiproduct batch plant involves 8 processing stages. The batch processing times t_{ij}, cleanup times CLN_{ikj} and the number of batches n_i are given in Table IV. Table V(a) lists the results of the LP1 scheduling model for this problem. The model consisted of 48 constraints and 401 continuous variables. It required 74.87 seconds of CPU time on Microvax II to solve this problem. (This computational time includes the time required to calculate the slacks for all possible product-pairs and solve it using ZOOM via GAMS).

Each entry in Table V(a) corresponds to a value of the variable NPRS^. The optimal schedule for this problem is a Mixed Product Campaign requiring 9017.78 hrs. Figure 7(a) Is the graph representation of the solution to LP1. It can be clearly seen that there are total of two subcycles in the solution. One subcycle contains products B and D. The remaining eighteen products belong to the other subcycle. Finally the subtour elimination constraints in (8) were used in LP3 with constraints corresponding to Q = {B_f D } and Q = {A to T except for B and D }. The solution to LP3 consisted of one single cycle. LP3 required 82.57 seconds of total computational time. The solution without any subcycles exhibits the CT of 9018.92 hrs and is listed in Table V(b). The graph representation can be seen in Figure 7(b). The lower and upper bounds for the corresponding makespan problem were calculated with equation (9) and MILPl respectively. The lower bound Is 9034.17 hrs. The upper bound for the makespan (9040.64 hrs) is obtained by breaking the cycle obtained from LP3 at a link between two successive batches of D. The solution corresponding to the upper bound of 9040.64 hrs. which has a maximum deviation of 0.07% from the optimum. Calculation of the upper bounds by solving problem MILPl using the MILP solver ZOOM through GAMS required 1.16 seconds of total computational time on Microvax n. Thus, the approximate method required a total of 158.6 seconds.

Applying the rigorous algorithm to example 2 yields the global optimal makespan of 9035.92 hrs. The total CPU time required to solve the problem MILP2 on the Microvax n was 134.96 sec. It should be noted that this problem did not yield subcycles and its solution was obtained from its LP relaxation. The sequence exhibiting the global optimum makespan is shown in Figure 8.

Condensed results of examples 1, 2 and other seven examples are presented In Table VI(a). Note that the first seven examples Involve several hundred batches, while example 8 consists of a plant with 12 stages and 1,000.000 batches belonging to 15 different products. This example Is of course unrealistic In terms of size, but has been Included to explicitly demonstrate the capability of the methods to handle very large number of batches. Also note that example 9 consisted of the highest number of products, i. e. 50. The processing times and clean-up times for the examples are of the order magnitude of the data given in Tables I and IV.

As seen in Table VI(a), for the approximate method the maximum gap in *all examples* with respect to the optimal makespan is less than 0.3%. Note that example 1 has zero gap, while example 2, 4, 6. 7, 8 and 9 have a gap smaller than 0.1%. Thus, these solutions clearly correspond to very near optimal solutions. In fact, in four cases (examples 2, 4, 6, 7) they correspond to the global optimum solution. It should be noted that LPl and LP3 (subcycle eliminating constraints) yielded integer niunber of pairs of products (NPRS^) for *all* these examples. Also, only 3 out of the 9 examples needed the cycle breaking constraints of LP3. The rest of the six examples when solved with LPl yielded no subcycles. Even for the other three examples which needed application of subcycle elimination constraints, LP3 was used only once for each of them.

For the case of the rigorous formulation MILP2 only 2 out of 9 examples required the solution with cycle breaking constraints (MDLP3). The problem MILP3 had to be solved exactly once for each of them. The rest of the seven examples yielded no subcycles when solved using MILP2. Also in *all cases* integer values of NPRS^ were obtained for problems MILP1, MILP2, and MILP3 through their LP relaxation.

The comparison of the computational time required by the approximate and rigorous methods is shown in Table VI(b). It can be seen that the total computational time required for the above nine examples using the approximate method ranged approximately from 5 to 550 seconds. For the rigorous method it ranged from 7 to 1450 seconds. Also, for the first seven problems (less than or equal to 15 products) the rigorous method required nearly 50% more CPU time and for example 9 (50 products) the difference increased to 200%. Example 2 was the only one where the rigorous method required somewhat less time than the approximate method. These results are interesting in that one would expect the rigorous method to be much more expensive than the approximate method. However, it seems that aggregation of batches and strong bounds explain the good computational behaviour of the rigorous model.

Finally, note from Table VII that there is a significant difference between the global optimum makespan and the suboptimal sequence where simple single product campaigns are used (in alphabetical order of products). Hence, it is clear that the proposed methods can yield substantial improvement over intuitive or heuristic solutions with quite reasonable computational efforts.

Conclusions

This paper has presented approximate and rigorous algorithms for the minimization of makespan in multiproduct batch plants that are scheduled with the Zero-Wait policy. It was shown that for the case when many batches are involved in relatively few products, a very efficient representation can be developed that aggregates the batches in terms of products. In particular, it was shown that the problem of minimizing cycle time and its graphical representation yield very useful insights in the development and effective solution of these algorithms with models LP1, LP3, MILP1 and MILP2.

Numerical results were reported for problems with up one million batches and 50 products. These were solved to optimality with MILP2 and for near optimal solutions with LP1, LP3 and MILP1. The approximate algorithm yielded very near optimal solutions with one half or less of the computational effort of the rigorous algorithm (MILP2). The computational requirement for the latter was in fact quite reasonable.

The significance of this paper is that the proposed models constitute highly effective and efficient scheduling algorithms for a problem that in principle is highly combinatorial in nature, in fact NP-hard.

Acknowledgments

TTie authors would like to acknowledge financial support from the National Science Foundation under grant CPE-8351237 and for partial support from the Engineering Design Research Center at Carnegie Mellon University.

APPENDIX I: Determination of slack times for LPl

Formulation LPl requires the determination of the slacks of all stages j for all possible combinations of pairs of N_p products. The total number of permutations of products that is possible when two are taken at a time is given by Nj*. For example, consider the data given in Table I(a). Here the total number of combinations of product-batches is 36 (N_p = 6). They are.

A-A, ••• • A-F
B-A. ... , B-F
C-A. ... . C-F (Al)
D-A, ... , D-F
E-A, ... E-F
F-A. ... . F-F

In order to determine the slacks SL^j the following simple procedure can be applied for each pair of products 1, k. The batch processing times of product 1 in stage J (ty). the transfer times to transfer product i to stage J (try) and the clean-up times between products i and k in stage J (CLN^) are specified.

1. Set the final times Ojj for product i in the M stages *{}* = *I* ...M) by the equations:

$$\theta^{f}_{ij} = \pounds_{j=1}\ t_{ij'} + tr_{if} \qquad j=\backslash..M \qquad 042)$$

2. Set the initial times 0'^ for product k in the M stages 0 = 1 ...M) by the equations:

$$\theta^{I}_{kj} = \sum_{j=1}^{j-1} t_{kj'} + "V + "q \qquad j\sim l..M\text{-}l \qquad (A3)$$

3.a) Calculate the differences between initial and final times:

$$^{d}ikj = fy- \ \% \qquad J*1\sim M \qquad (A4)$$

b) Set the time violation $\overline{d}$^. to

$$\bar{d}_{ik} - jA (d_{ikj}) \qquad (AS)$$

c) Set the slacks without the clean-up times. $\overline{S}L$^ to

$$\overline{SL}_{ikj} = " \bar{d}_{ik} + d_{ikj} \qquad j=1..M \qquad (A6)$$

d) Set the clean-up time violation $\overline{C}L$^ to

$$- \quad - \quad) \qquad (A7)$$

e) Set the final slacks with clean-up times, the transfer time etc to

$$SL_{ikj} = \overline{SL}_{ikj} - \overline{C}L_{ikj} \qquad j=1..M \qquad (A8)$$

It can be easily verlfled that for the example In Table I(a) the above procedure yields the slacks shown In Table I(b).

APPENDIX n : Detection of Subcyclcs in the LP solution

Solution to LP1, LP2 and LP3 may contain subcycles which may result in the underestimation of the optimal cycle time- The existence of subcycles in the solution can be detected using a simple two step procedure presented below. Given are the values for the variables $NPRS_{lk}$ in the solution to LP1, LP2 or LP3.

1. Draw a directed graph indicating the existence of product pairs in the optimal sequence. This directed graph will look like the schedule in graph-representation except for the number of pairs above each arc. Thus the graph here will consist of nodes representing the products to be scheduled and arcs joining various product if $NPRS_{ik} > 0$ in the LP solution.

2. Start with any randomly chosen product 1 and merge product 1 and k to form a new super-product Ik if there is an arc connecting products i and k. Redraw the arcs accordingly. Merge into this super-product, all the other products connected to it.

Repeat this step till no more merging is possible.

3. If more than one super-products left at the end of second step, subcycles exist in the solution. Further, the number of super-products left specifies the number of subcycles in the solution.

Consider for an example, solution to a scheduling LP shown in Figure 4(a). Application the procedure described above results In two super-products ABCD and EF. Thus there are two subcycles In the solution. Using the subcycle eliminating procedure described previously In the paper, results in the LP solution as shown in Figure 4(b) in graph representation. Applying the *Subcycle Detection algorithm* described above results in *one* super-product ABCDEF, which means that the concerned solution contains no subcycles.

The procedure described above can be easily automated and identifies subcycles in very short amount of time.

Appendix III: Approximate Scheduling Algorithm

The scheduling procedure for ZW policy that is based on the models LP1 and MILP1 can be summarized as follows for the case of Mixed Product Campaigns :

- Step 1:
  - a) Given the batch processing times and cleanup times, determine the slacks SL_{ikj} as indicated in Appendix I.
  - b) Given the specific number of batches for each product n_i, $i=1 \ldots N_p$, set up and solve LP1 to determine the numbers of successive batches of various products, $NPRS_{ik}$.
  - c) If all the variables $NPRS_{ik}$ are integer go to step 2. Otherwise, use branch and bound or a trial and error rounding scheme.

- Step 2: Use the Graph representation or the algorithmic procedure to determine the campaigns that are part of the optimal schedule.

- Step 3: Combine these campaigns to form minimum number of continuous cycles using Rule #4.

- Step 4: If the solution consists of exactly one cycle then go to step 5. If there are subcycles in the solution (see Appendix II) add repeatedly the subtour elimination constraints in (8) until a solution to LP3 does not contain subcycles. This yields a family of schedules with the cycle time of CT^G.

- Step 5: Solve MILP1 to obtain the best possible solution contained within this optimal cycle to minimize the makespan. This yields an upper bound. Note that the equation in (9) can be used to compute the lower bound for the makespan problem. If both the bounds coincide the global optimum for the makespan problem is guaranteed.

For the case of Single Product Campaigns problem steps 2 and 3 are not needed.

Appendix IV: Rigorous Scheduling Algorithm

The scheduling procedure for ZW policy that Is based on the model MILP2 can be summarized as follows for the case of Mixed Product Campaigns :

- Step 1:
  - a) Given the batch processing times and cleanup times, determine the slacks SL^ as Indicated In Appendix I.
  - b) Given the specific number of batches for each product nj_f 1=1... Np, set up and solve MILP2 to determine the numbers of successive batches of various products, NPRS^.
  - c) If all the variables NPRS^ are integer go to step 2. Otherwise, use branch and bound or a trial and error rounding scheme.

- Step 2: Use the Graph representation or the algorithmic procedure to determine the campaigns that are part of the optimal schedule.

- Step 3: Combine these campaigns to form minimum number of continuous cycles using Rule #4.

- Step 4: If the solution consists of exactly one cycle then stop. If there are subcycles in the solution (see Appendix II) add repeatedly the subtour elimination constraints in (8) until a solution to MILP3 does not contain subcycles. This yields a family of schedules with the global optimum makespan.

For the case of Single Product Campaigns problem steps 2 and 3 are not needed.

References

1. Baker, K. R. Introduction to Sequencing and Scheduling; John Wiley and Sons, 1975.

2. Birewar, D. B; Grossmann, I. E. "Incorporating Scheduling In The Design Of The Optimal of Multiproduct Batch Plants". Como. and Chem. Engg. 1989, 13, 141-161.

3. Campbell, H. G.; Dudek, R. A.; Smith., M. L. "A heuristic algorithm for the n job m machine sequencing problem". Manag. Sci. 1970, 16, 630-637.

4. Dannenbring, D. G. "An Evaluation of Flowshop Sequencing Heuristics". Manag. Sci. 1977,22,1174-1182.

5. Garey, M. R.; Johnson, D. S.; R. Sethi "The Complexity of Flowshop and Jobshop Scheduling". Math. Oper. Res. 1979,1117-129.

6. Garfinkel, R. S.; Nemhauser G. L. Integer Programming: John Wiley and Sons, 1972.

7. Graham, R. L.; Lawler, E. L.; Lenstra, J. K.; Linnooy Kan A. H.G. "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey". Anna. Discrete Math 1979,5, 287-326.

8. Graves, S. C. "A Review of Production Scheduling". Oper. Res. 1981,23, 646-675.

9. Gupta, J. N. D. "Optimal Flowshop Schedules with No Intermediate Storage Space". Naval Res. Logis. Qua. 1976,22,235-243.

10. Ku, H. M.; Karimi, I. A. "Scheduling in Multistage Batch Processes with Finite Intermediate Storage. Part II: Approximate algorithms", Presented at the Annual Meeting of American Institute of Chemical Engineers, Miami, FL; AIChE : New York, NY, 1986; paper 72 E.

11. Ku, H. M.; Rajagopalan, D.; Karimi, I. "Scheduling in Batch Processes".
CEP 1987, 83(8), 35-45.

12. Kuriyan, K.; G. V. Reklaitis. "Scheduling Network Flowshops so as to Minimize Makespan".
Comp. and Chem. Engg. 1989, 12, 187-200.

13. Marsten, Roy. In ZOOM User's Manual;
University of Arizona: Tucson, AZ, 1986.

14. Meeraus, A.; Brooke, T. In GAMS;
Development Research Department, The World Bank : Washington, DC, 1985.

15. Panwalkar, S. S.; Iskander, E. "A Survey of Scheduling Rules".
Oper. Res. 1977,25, 45-61.

16. Pekny, J. S.; Miller, D. L "A Parallel Branch and Bound Algorithm for Solving Large Assymmetric Travelling Salesman Problem". Internal Report EDRC 05-27-88; Engineering Design Research Center, Carnegie Mellon University : Pittsburgh, PA, May 1988.

17. Reddi, S. S.; Ramamoorthy, C. V. "On the Optimal Flowshop Sequencing Problem with No Wait in Process".
Oper. Res. Q. 1972, *21,* 323-331.

18. Wismer, D. A. "A Solution of the Flowshop Scheduling Problem with No Intermediate Queues".
Oper. Res. 1972,2Q, 689-697.

LIST OF TABLES

Table I : Data for Example 1

(a) Batch Processing Times [tjj] Hrs.

Stage j / Prod, r s.	STG1	STG2	STG3	STG4
A	6	2	4	1
B	1	5	3	5
C	2	7	3	7
D	8	1	5	2
E	4	1	2	2
F	3	6	2	4

(b) Slack Times [SL_{jki}] Hrs.

Stage j / Pair ik	1	2	3	4	Stage j / Pair ik	1	2	3	4	Stage j / Pair ik	1	2	3	4
A.A	0	4	2	5	C.A	5	4	3	0	E.A	0	5	5	7
A.B	1	0	1	3	C.B	8	2	4	0	EB	0	0	3	4
A.C	0	0	3	5	C.C	5	0	4	0	EC	0	1	6	7
A.D	0	6	3	7	CD	3	4	2	0	ED	0	7	6	9
A.E	1	3	0	1	CE	10	7	5	0	EE	0	3	2	2
A.F	0	1	3	4	CF	6	2	5	0	EF	0	2	6	6
B.A	1	2	1	0	D.A	0	5	2	4	F.A	0	0	0	0
B.B	4	0	2	0	D.B	0	0	0	1	F.B	5	0	3	2
B.C	3	0	4	2	D.C	0	1	3	4	F.C	4	0	5	4
B.D	0	3	1	1	D.D	0	7	3	6	F.D	0	2	1	2
B.E	6	5	3	0	OE	1	4	0	0	F£	5	3	2	0
B.F	2	0	3	0	D.F	0	2	3	3	F.F	3	0	4	2

Table II : Solution to Example 1.

(a) Mixed Product Campaigns

$NPRS_{ik}$

i \ k	A	B	C	D	E	F
A			3		2	
B				1		6
C				3		
D		5				
E		2			2	
F	5			1		

(b) Single Product Campaigns

$NPRS_{ik}$

i \ k	A	B	C	D	E	F
A	4				1	
B		6				1
C			2	1		
D		1		4		
E			1		3	
F	1					5

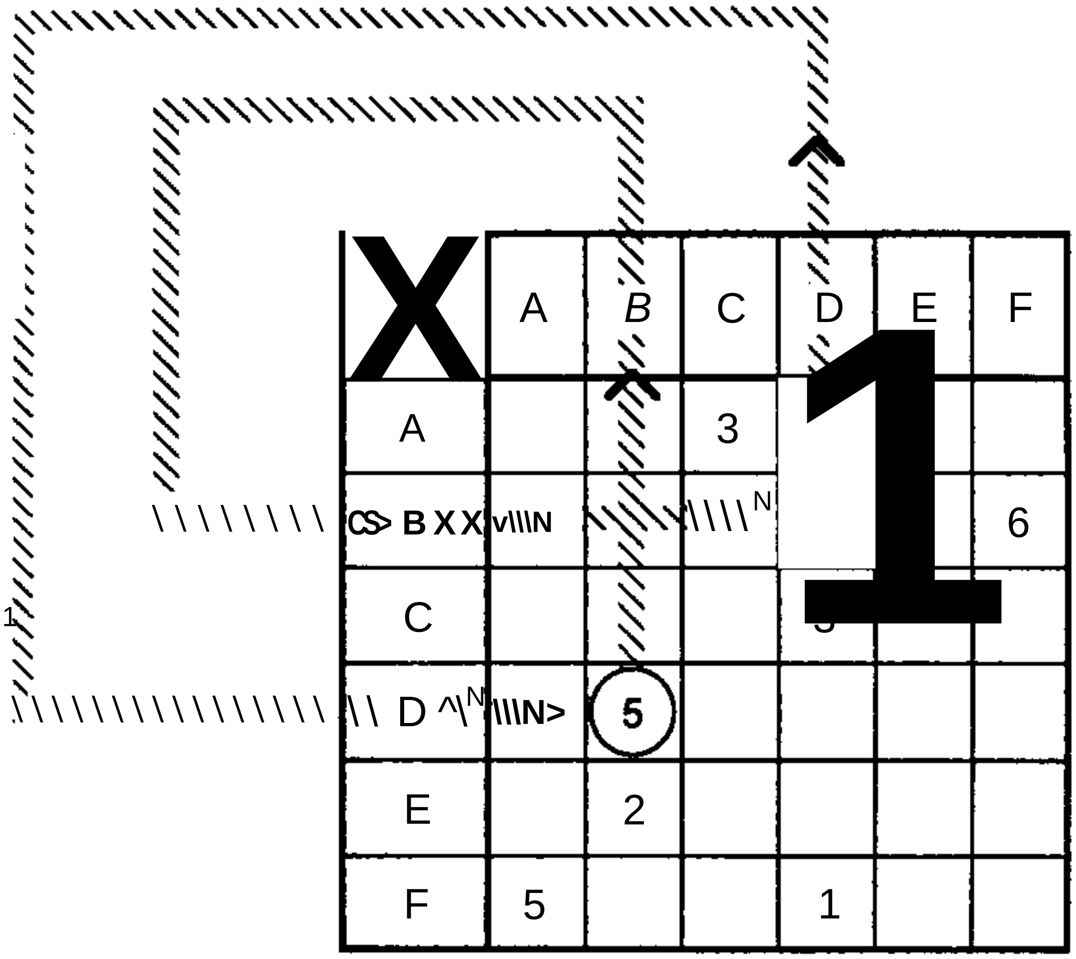
Table III : Derivation of schedule

(a) Step 1

X	A	B	C	D	E	F
A			3		2	
B				1		6.
C				3		
D		5				
E		2			②	
F	5			1		

Twice a batch of E is followed by another batch of E [Rule # 1].

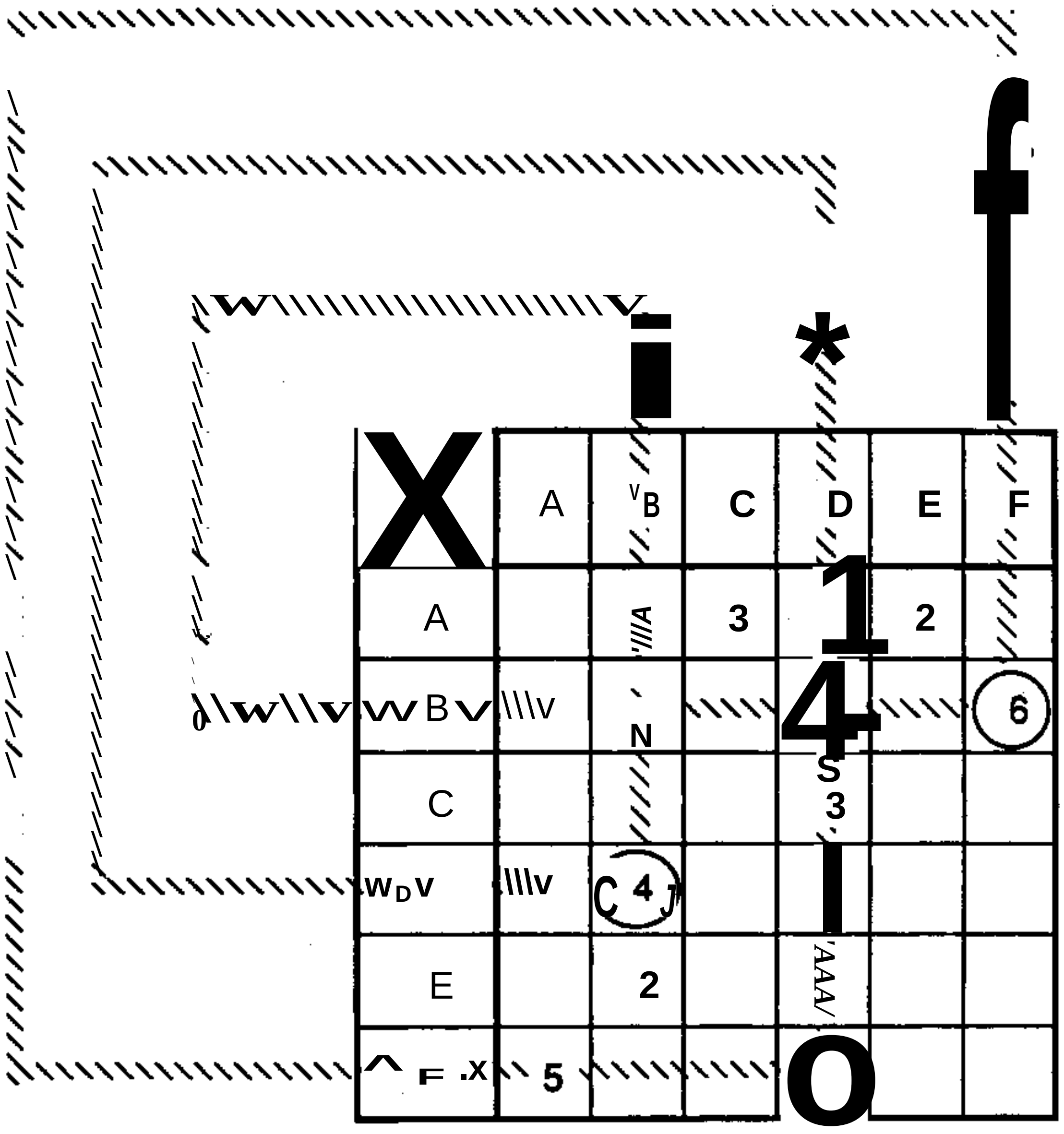
Table III(b) : Step 2



X	A	B	C	D	E	F
A			3	1		
B						6
C				[illegible]		
D		5				
E		2				
F	5			1		

One batch of D follows B and one batch of B follows D [Rule #2J.

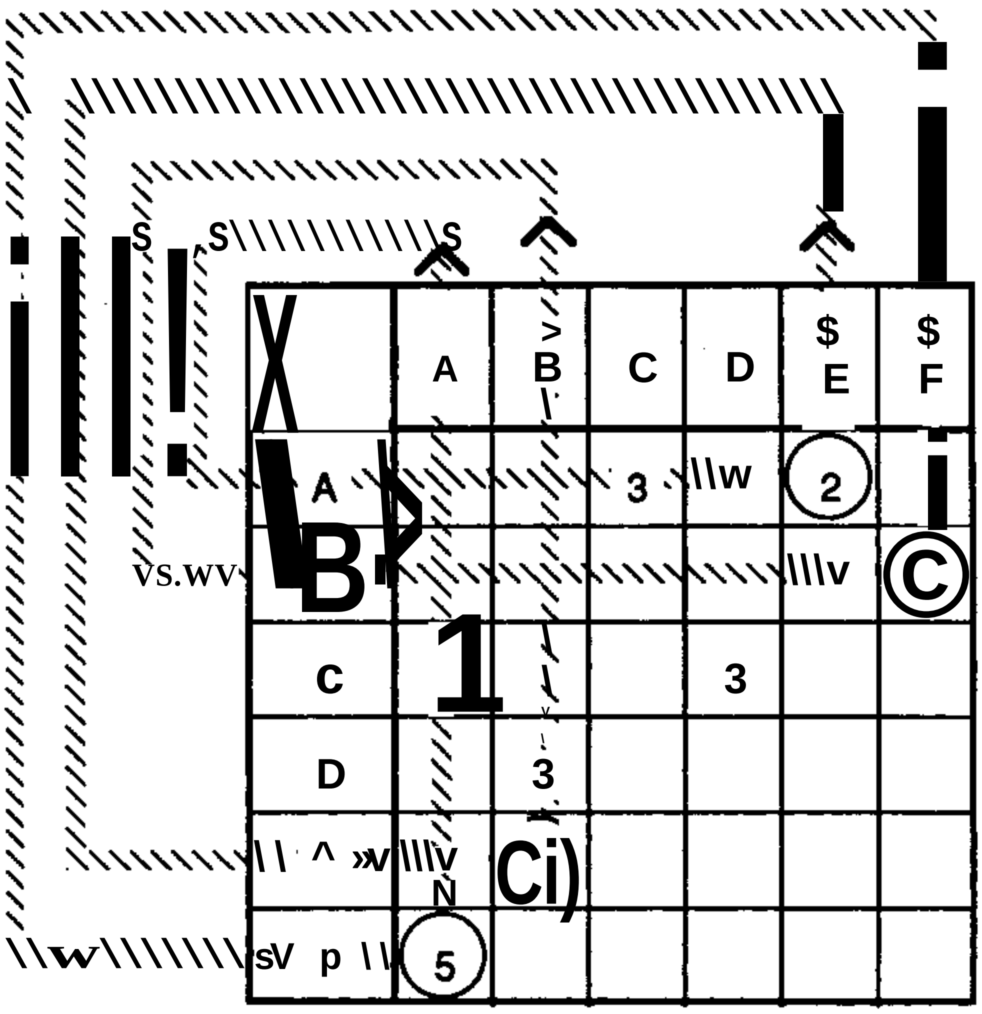
Table III(c) : Step 3



	A	B	C	D	E	F
A			3	1	2	
B				4		6
C				3		
D		4				
E		2				
F	5					

Once a batch of D follows F, B follows **D** , **F follows B** in the schedule [**Rule # 2].**

Table lll(d) :Step 4



Twice a batch of B follows E, F follows B, A follows F and E follows A [Rule #2, Rule # 3(a)].

Table IV : Data for Example 2

Batch Processing Times [t,,] (hrs)

j / i	1	2	3	4	5	6	7	8	n,
A	5.95	8.43	9.85	9.39	5.76	4.40	8.33	6.68	80
B	8.44	3.24	2.39	1.54	6.67	3.89	1.65	7.89	90
C	9.80	9.58	6.07	2.64	0.39	8.88	0.72	5.76	56
D	3.t7	5.00	4.79	4.19	2.17	0.78	3.82	2.80	27
E	4.23	6.98	2.23	8.14	0.51	5.72	6.95	2.46	90
F	4.75	6.37	3.42	6.75	6.78	2.66	6.79	9.40	45
G	2.75	9.15	5.00	6.81	7.31	3.77	8.03	4.09	35
H	4.42	3.50	0.95	0.28	9.39	5.49	2.84	7.19	38
1	0.61	6.30	2.30	6.16	5.65	2.73	6.03	3.78	96
J	9.54	7.22	4.13	8.34	2.22	0.89	9.64	2.61	84
K	8.63	8.16	0.32	8.99	4.28	9.34	8.01	3.68	03
L	7.05	4.10	7.37	8.90	1.89	4.81	8.85	7.70	13
M	8.55	6.25	3.42	6.48	4.65	3.09	4.50	7.36	09
N	8.48	4.84	4.94	7.03	7.02	8.49	4.52	2.72	47
0	2.48	9.76	7.35	5.61	3.23	9.12	4.36	0.88	14
P	9.41	9.39	2.35	5.41	8.89	2.01	8.91	9.68	92
Q	0.42	4.78	8.96	8.08	9.77	2.26	9.52	4.39	76
R	8.26	1.19	3.89	5.47	5.31	3.54	7.07	8.31	75
S	7.56	9.60	1.82	6.90	4.67	9.61	0.88	6.56	49
T	0.60	1.19	9.72	1.03	3.82	7.30	3.35	6.72	40

PRODUCT PAIRS"	CLEAN-JP TIMES {HRS] *
A.(A-T)	0.5
B.(A-T)	0.4
C.(A-T)	0.3
D.(A-T)	0.3
E.(D-T)	0.2
E.B	0.2
E.(A,C)	0
F.(A-T)	0.4
G.(A-T)	0.3
H.(A-T)	0.3
I.(A-T)	0.4
J.(A-T)	0
K.(A-T)	0
L.(A-T)	0.1
M.(A-T)	0.3
N.(A-T)	0.3
O.(A-T)	0
P.(A-T)	0.2
Q.(A-T)	0.3
R.(A-T)	0.1
S.(A-T)	0
T.(A-T)	0.1

* The clean-up times are equal to zero when the same products follow the other.

** Notation : P1.P2 => P1 to P2 ; P1.(P2-Pn) »> P1 to P2. . . . . .P1 to Pn.

Table V(a) : Solution to Example 2 without cycle - breaking constraints

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
A	79																1			
B		89		1					1											
C			47					9												
D		1		26																
E					15			29					9							37
F						10						13		22						
G																			35	
H									35											3
I									21									75		
J			9							61									14	
K														3						
L															13					
M						9														
N							35							11			1			
O																14				
P											3			11		78				
Q	1														1		74			
R					75															
S						26				23										
T									40											

Table V(b) : Solution to Example 2 with cycle - breaking constraints

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	0	P	Q	R	S	T
A	79																1			
B		88		1					1											
C			47					9												
D		1		26																
E		1			15			29					9							36
F						10						13		22						
G																			35	
H									34											4
1									21									75		
J			9							61									14	
K														3						
L															13					
M						9														
N							35							11			1			
0																14				
P											3			11		78				
Q	1														1		74			
R					75															
S						26				23										
T									40											

Table VI(a) : Results for the scheduling examples (ZW Policy)

EX	No. of Stages	No. of Products	No. of Batches	Cycle Time [Hrs] LP1	LP3	Makespan Bounds [Hrs] Lower [EQUN9J	Upper [MILP1]	% GAP[2]	Global Makespan Solution [MILP2]
1	4	6	30	140[1]	---	145	145	0	145[3]
2	8	20	1059	9017.78	9018.92	9034.17	9040.64	0.07	9035.92
3	7	10	73	585	586	604	606	0.3	605
4	6	10	274	2259.4[1]	---	2272	2273.4	0.06	2273.4[3]
5	6	10	260	2139.06[1]	...	2146.11	2150.82	0.2	2149.24
6	4	10	344	2530.05*	—	2535.33	2535.57	0.009	2535.57[3]
7	7	7	402	3472.13'	...	3488.68	3488.89	0.006	3488.89[3]
8	12	15	1 x 10*	9415259.42	9415259.50	9415286.0	9415295.68	0.001	9415293.02
9	9	50	2392	20272.5*	—	20290.78	20296.41	0.028	20295.27

[1] **Here the solution to the problem LP1 did not contain any subcycles.**

[2] **% Gap - 100 *(Upper - Lower) / Upper**

[3] **Makespan solution obtained by the approximate method matches with global optimum.**

Table VI(b) : Comparison of Computational Performances

EX	No. of Products	Total CPU "fime [sec][1]	
		Cycle Time Minimization followed by MILP1	**Globally Optimal Makespan by MILP2**
1	6	5.23	7.21
7	7	12.66	16.85
6	10	12.06	19.19
4	10	15.96	22.45
5	10	14.84	21.98
3	10	31.97	49.8
8	15	121.25	165.12
2	20	158.6	134.96
9	50	547.03	1450.4

1 **The total CPU time includes the time required to calculate the slacks and the time to solve LP1 with the constraint (6), LP3 or LP4 and MILP1 using the LP solver ZOOM through the interface GAMS on Microvax II.**

Table VII : Comparison of optimal solution and simple SPCs

EX	Global Makespan Solution [MILP2] (hrs)	Simple SPC[1] Makespan (hrs)
1	145	186
2	9035.92	9451.75
3	605	686
4	2273.4	2445.7
5	2149.24	2272.83
6	2535.57	3037.71
7	3488.89	3583.84
8	9415293.02	9549095.87
9	20295.27	21966.55

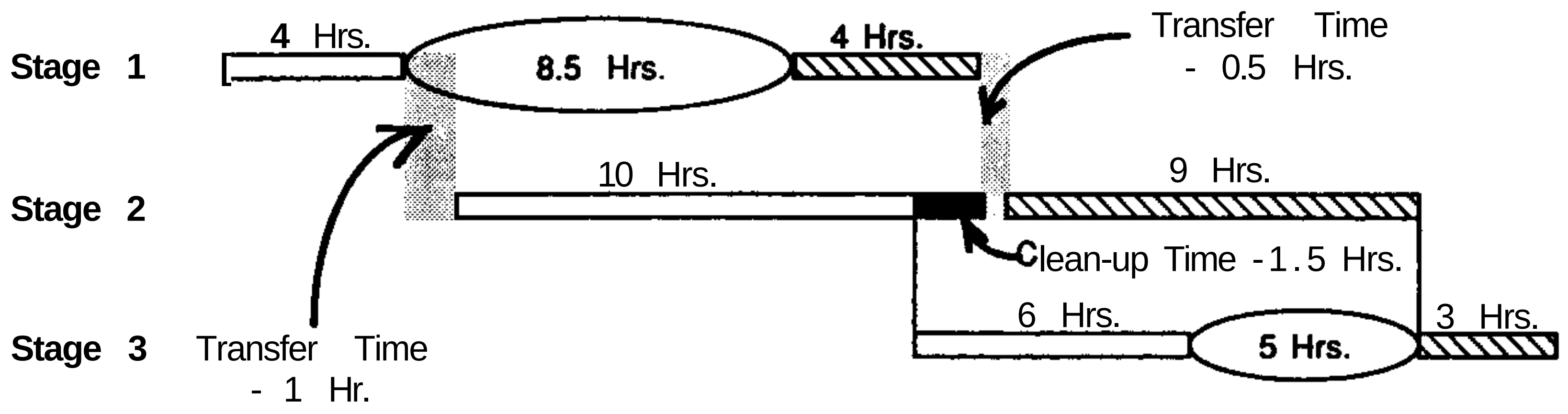
[1]Here its assumed that all batches of product A are followed by all batches of product B, followed by batches of C and so on.

LIST OF FIGURES

SL_{AB1} = 8.5 Hrs.

SL_{AB2} = 2 Hrs.

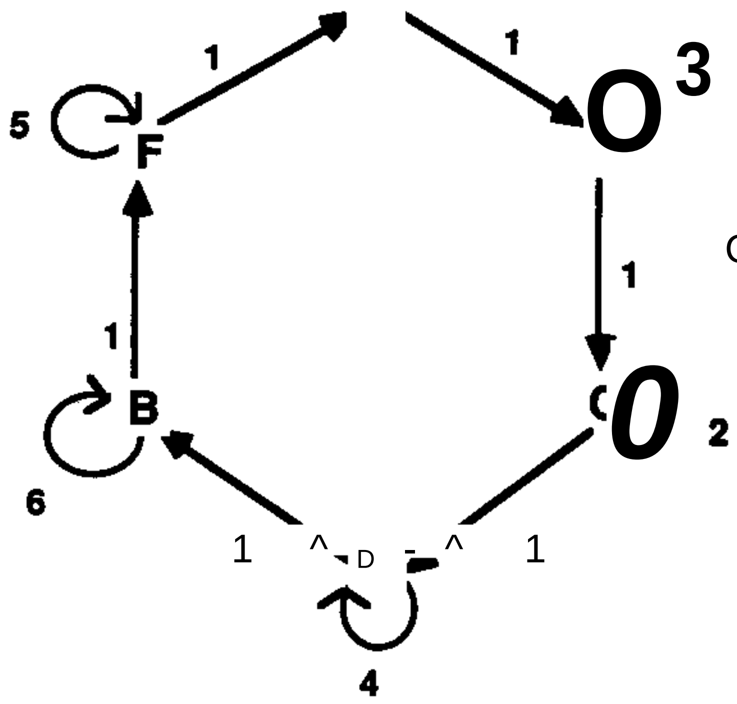
SL_{AB3} = 5 Hrs.



LEGEND : ▭ Product A ▨ Product B

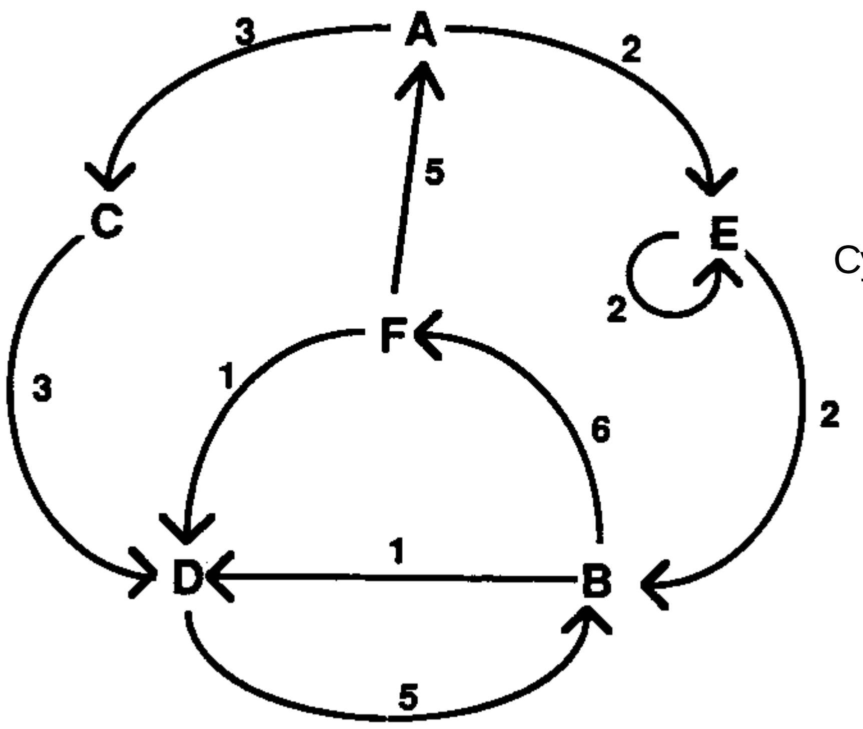
░ Transfer Time ■ Clean-up Time ⬭ Slacks / Idle Time

Figure 1 : Determining the Slacks [SL_{ikj}]



Cycle Time = 172 hrs

(a)



Cycle Time = 140 hrs

(b)

Figure 2 : Solution in the Graph Representation.
(a) Single Product Campaigns ;
(b) Mixed Product Campaigns.

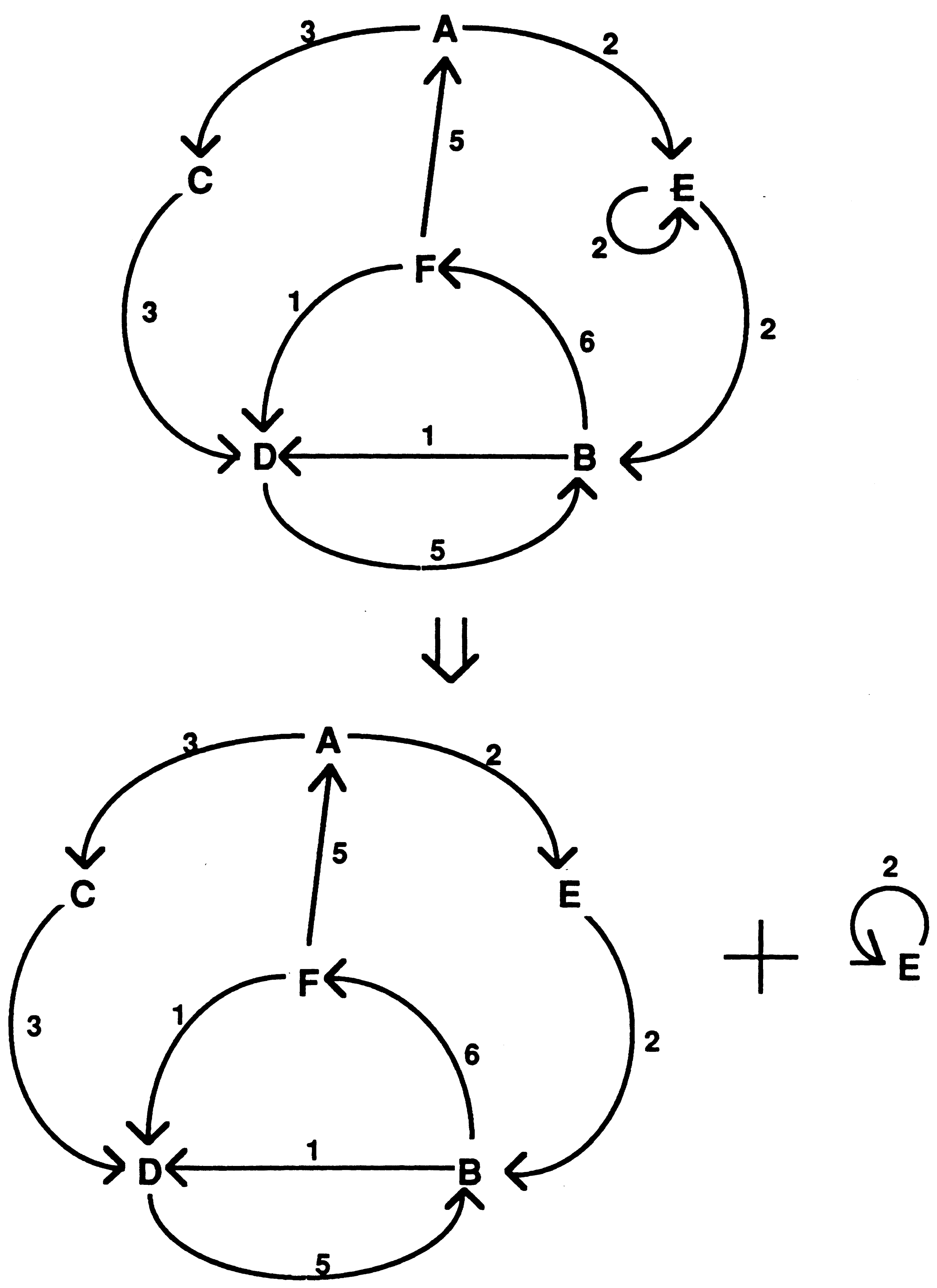


Figure 3 : Derivation of schedule.

(a) Step 1

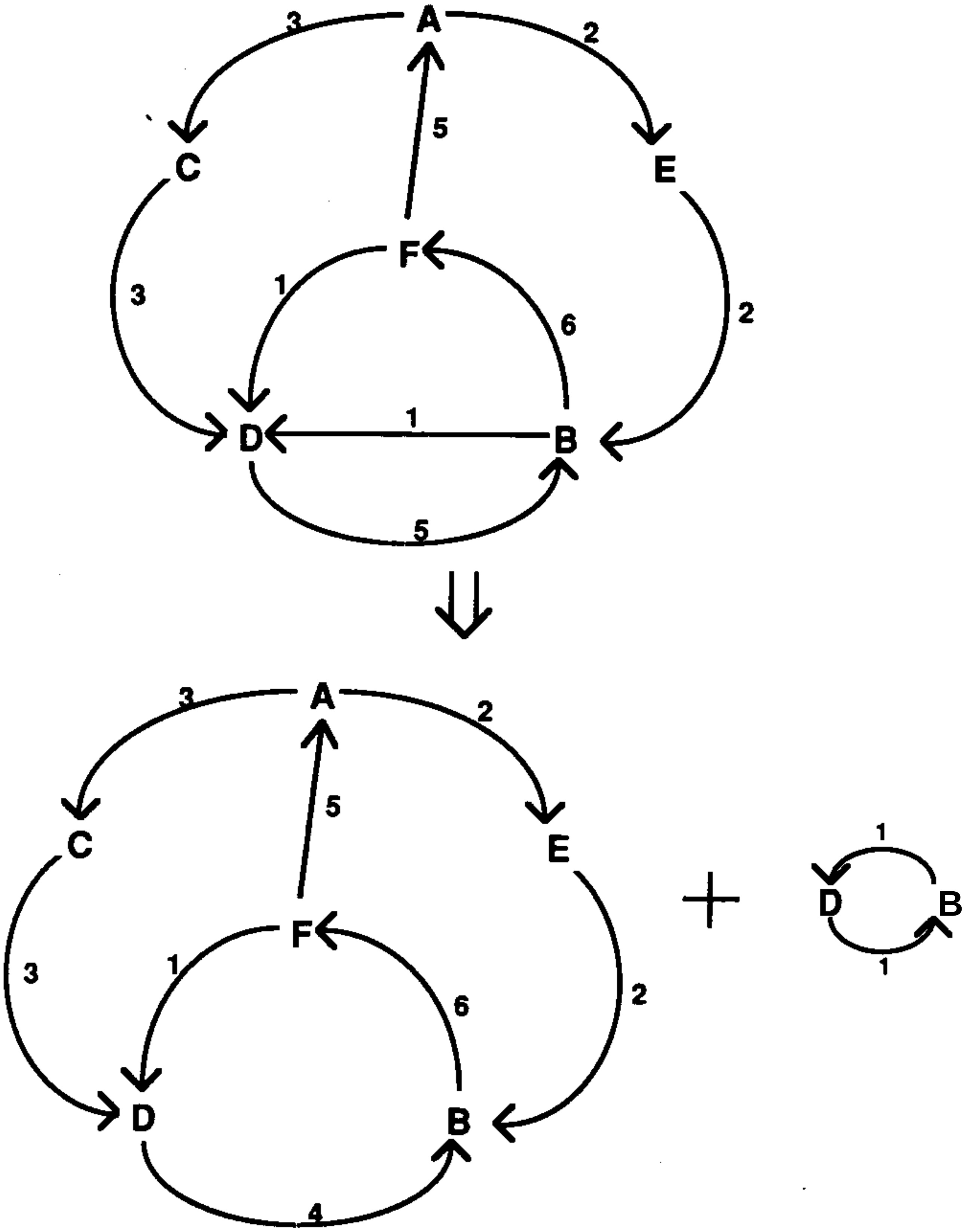


Figure 3(b) : Step 2

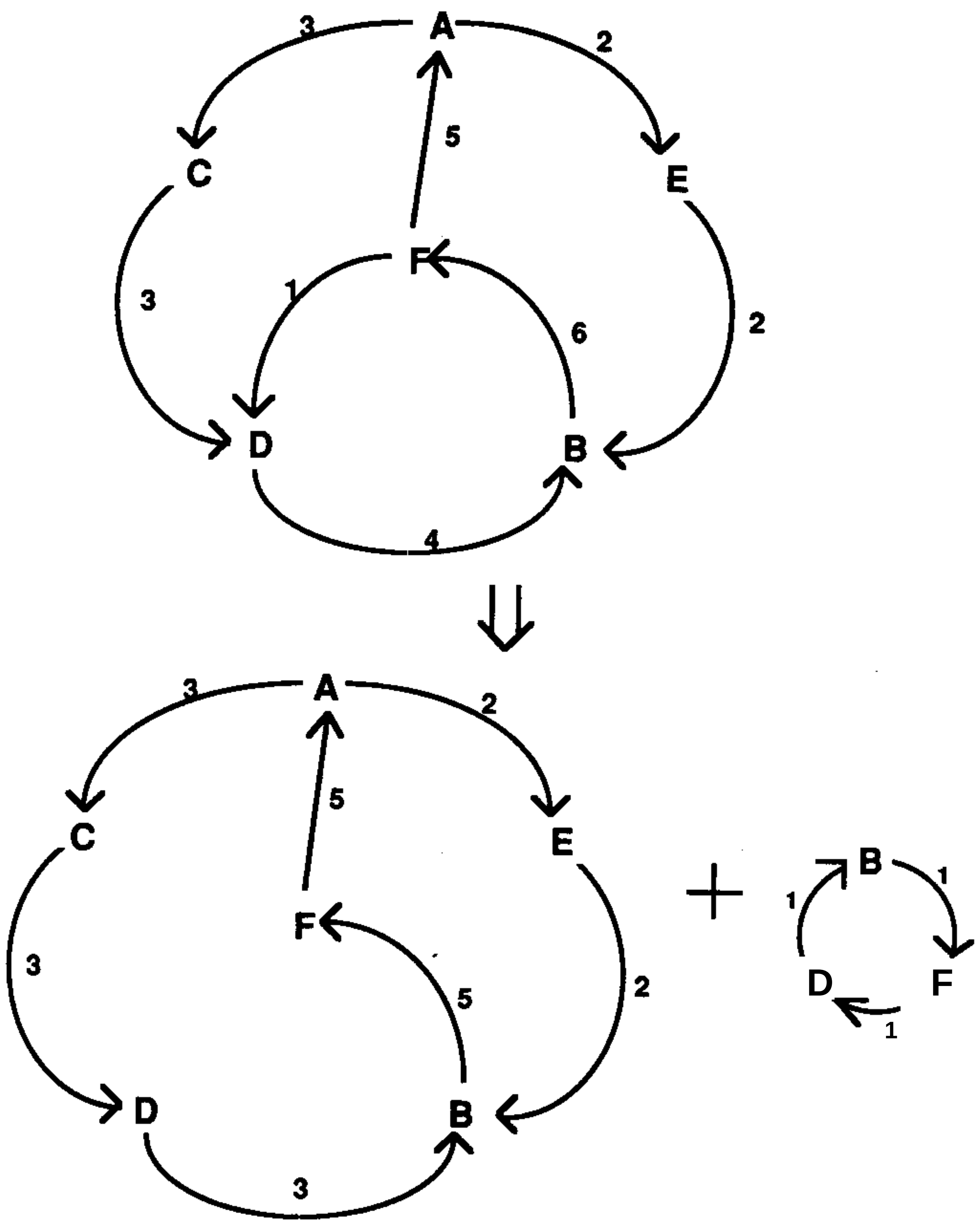


Figure 3(c) : Step 3

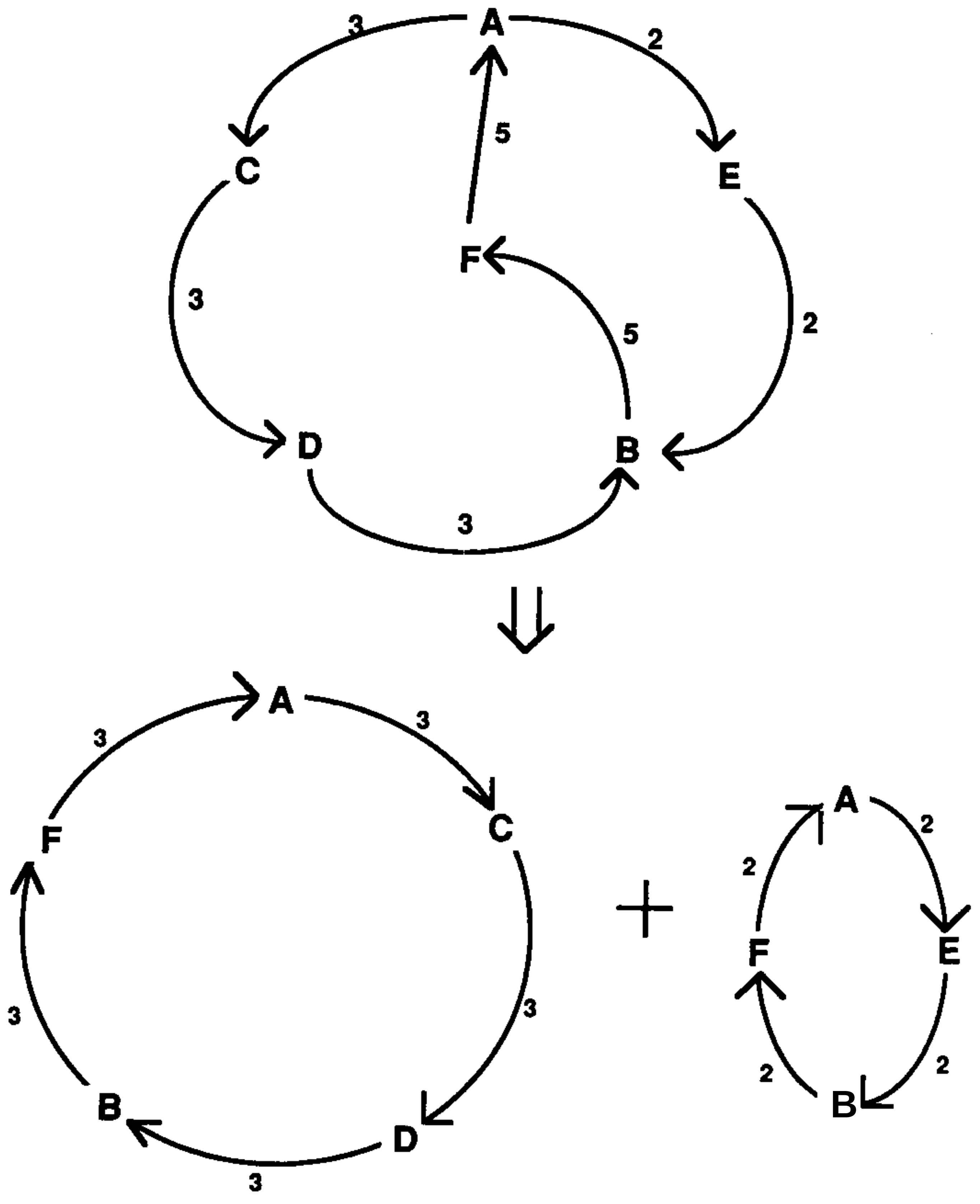
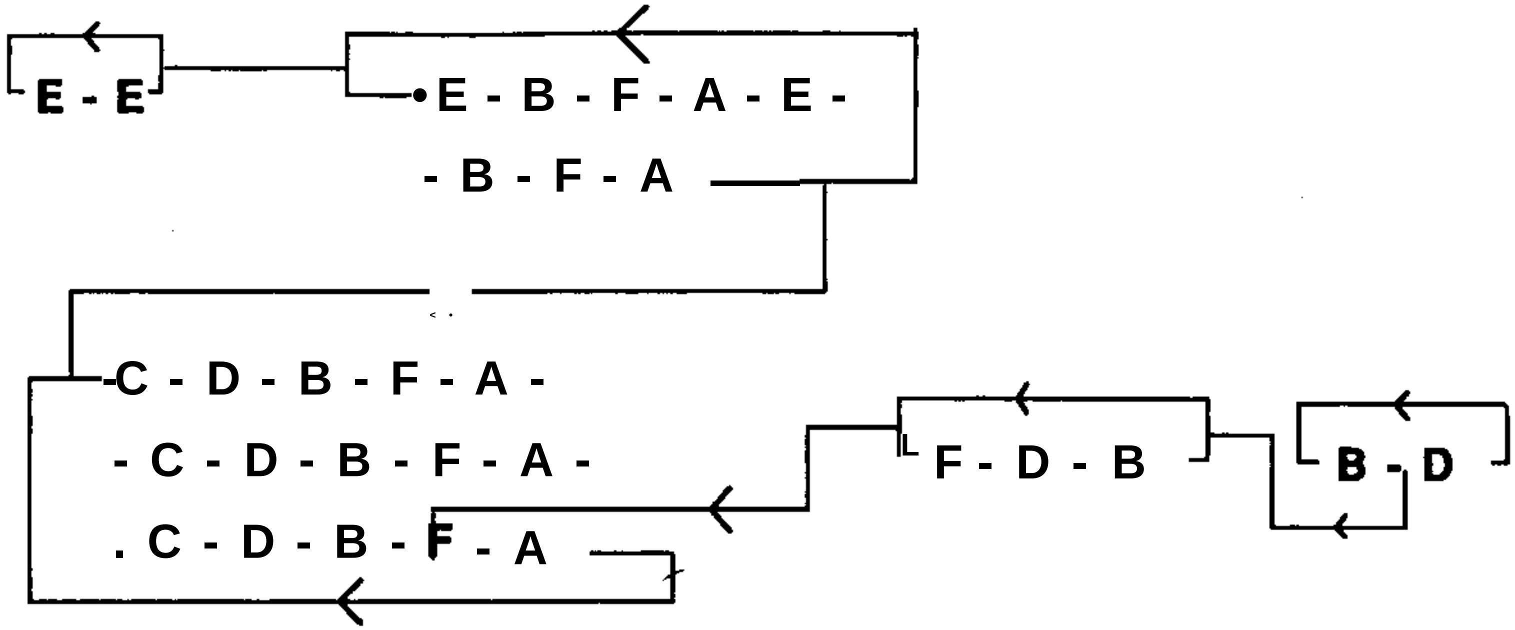
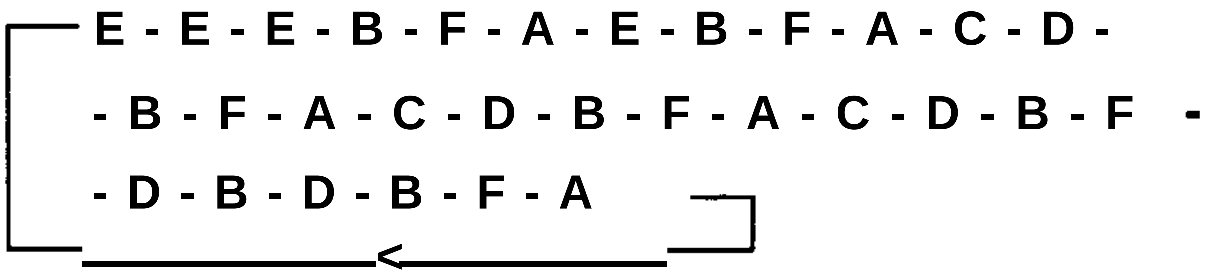


Figure 3(d) : Step 4

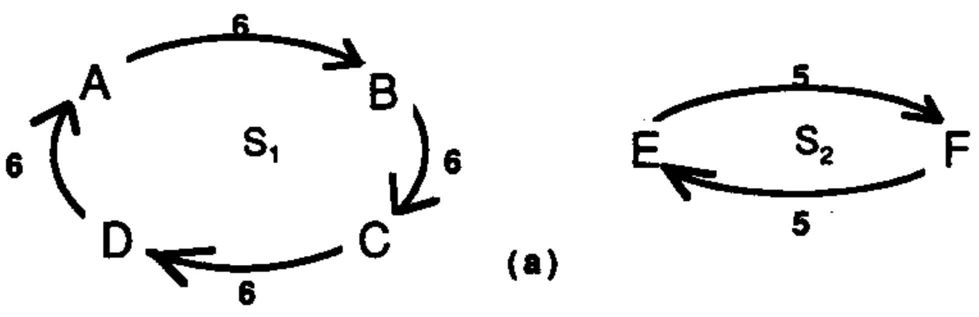


OPTIMAL CYCLE:

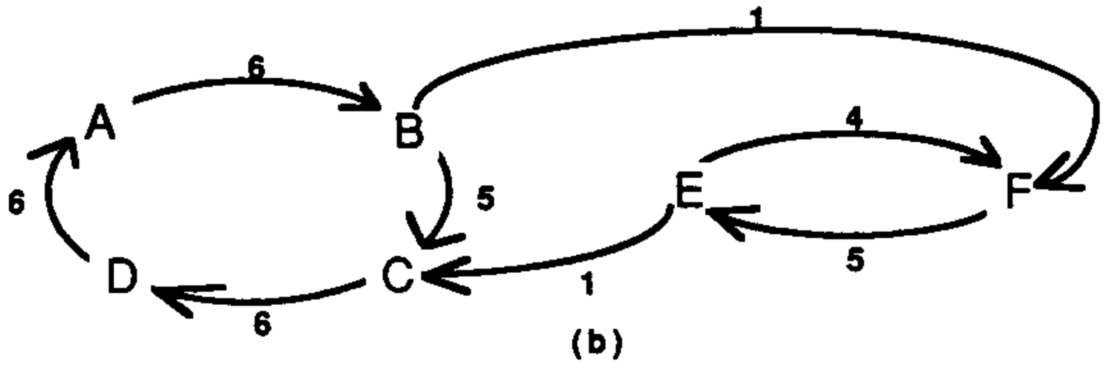


E - E - E - B - F - A - E - B - F - A - C - D -
- B - F - A - C - D - B - F - A - C - D - B - F -
- D - B - D - B - F - A

Figure 3(e) : Joining individual Campaigns - Step 5



(a)



(b)

!___^ A - B - C - D - A - B - C - D - A - B - C - D -

- A - B - C - D - A - B - C - D - A - B - C - D

E - F - E - F - E - F - E - F - E - F

(c)

A - B - C - D - A - B - C - D - A - B - C - D -

- A - B - C - D - A - B - C - D - A - B - F - E -

- F - E - F - E - F - E - F - E - C - D

(d)

Figure 4 : Eliminating the subcycles :
(a) Solution to LP1 in graph representation
(b) Solution without subcycles in graph representation
(c) Schedule according to LP1 [containing subcycles]
(d) Schedule after eliminating the subcycles

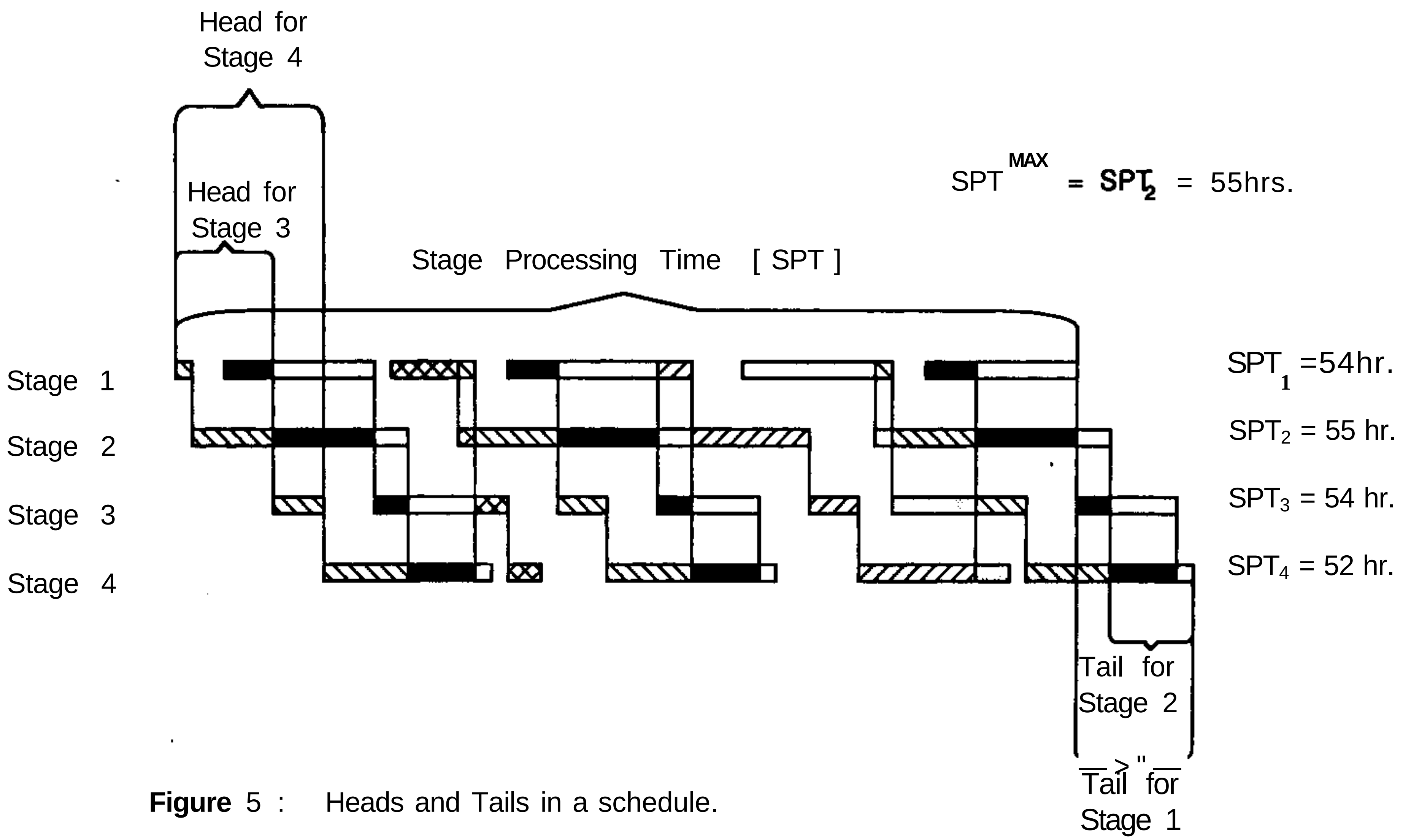


Figure 5 : Heads and Tails in a schedule.

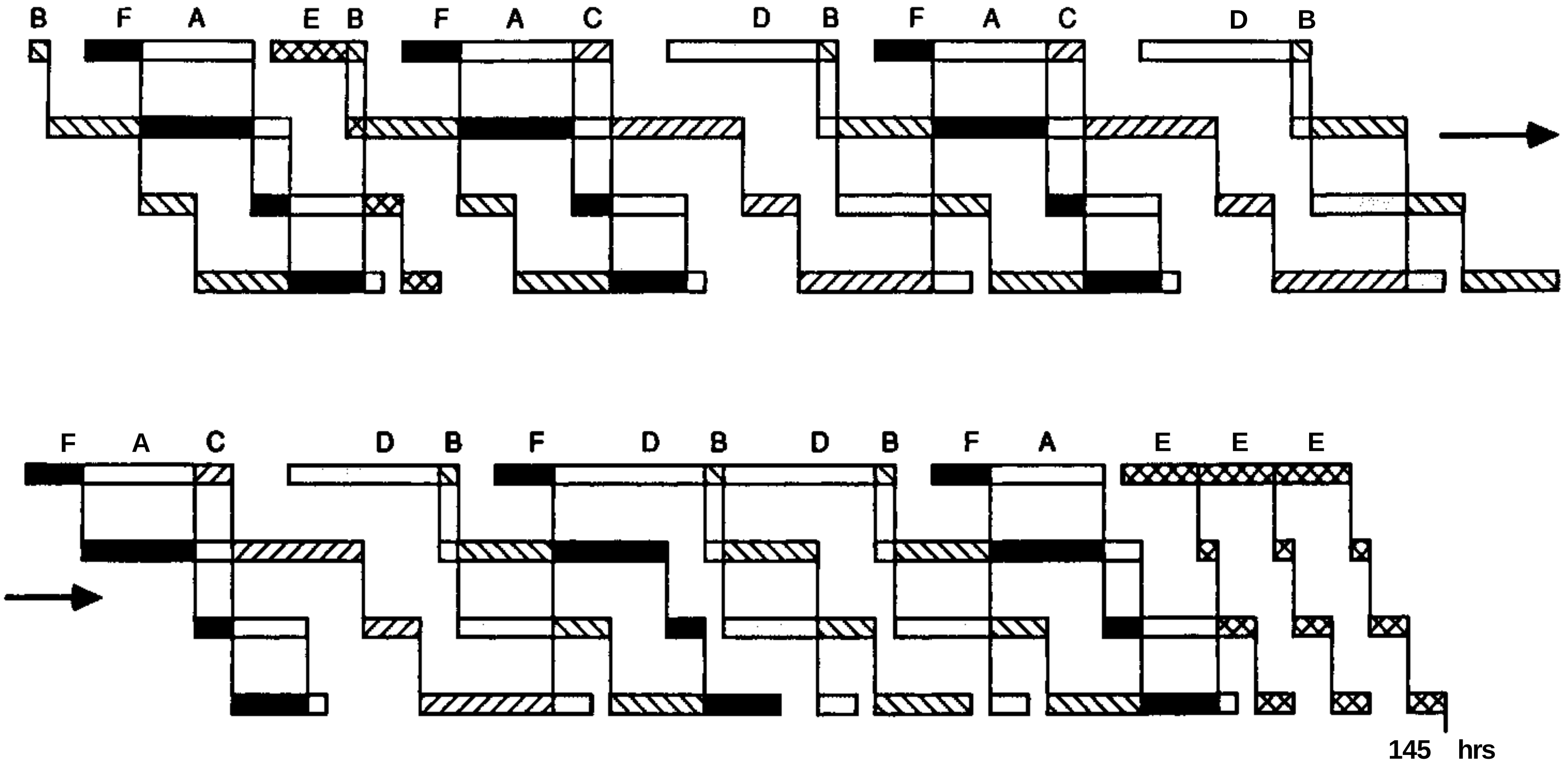


Figure 6 : Optimal Schedules for Example 1
(a) Mixed Product Campaigns.

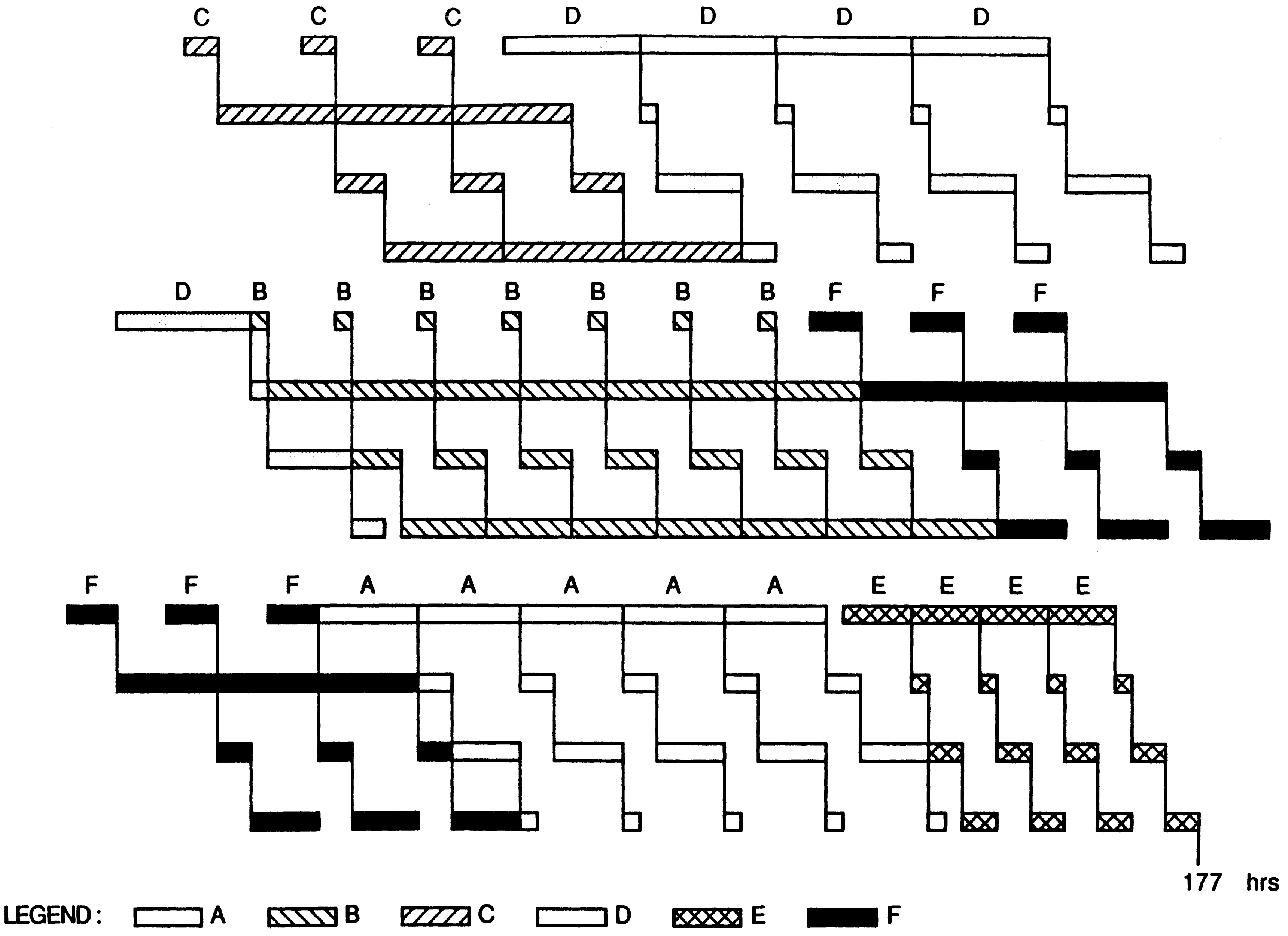
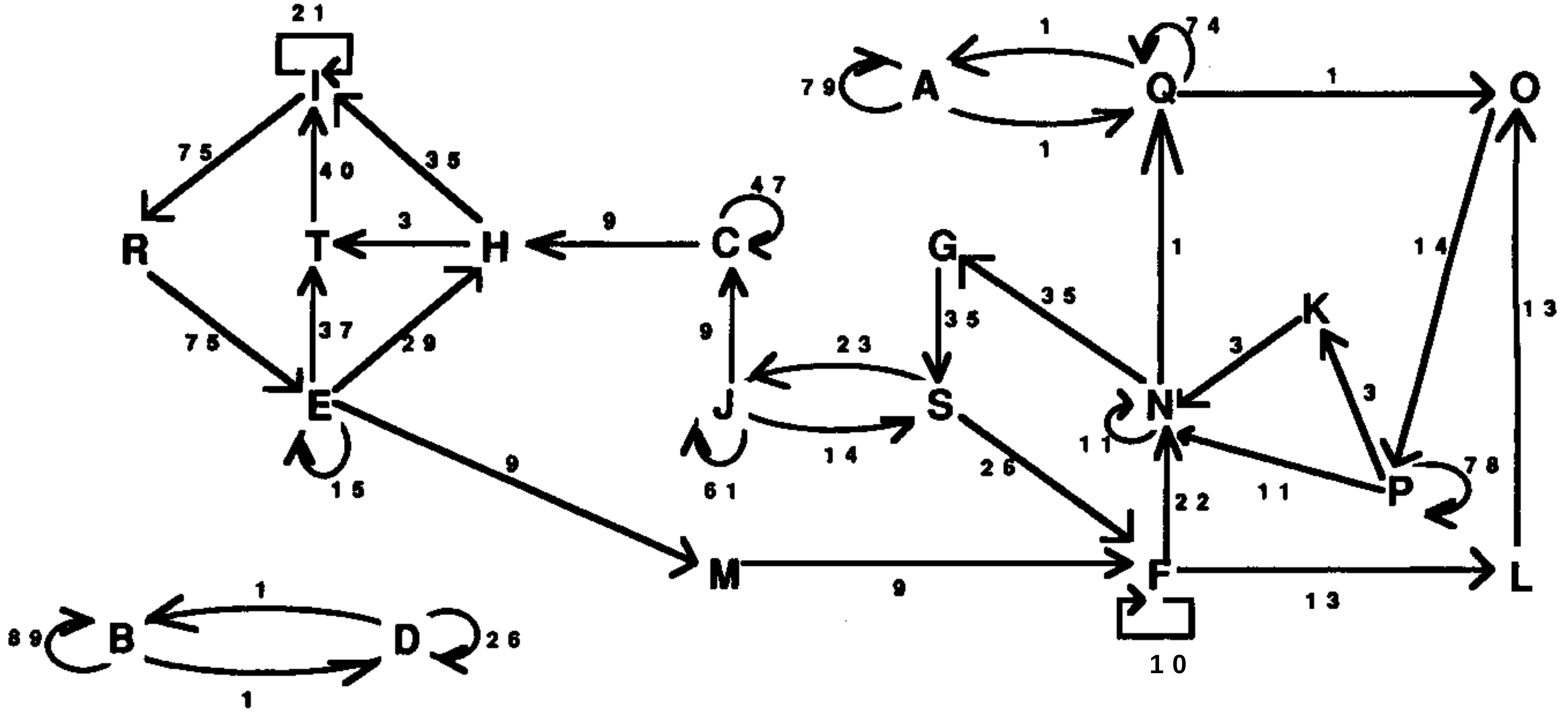
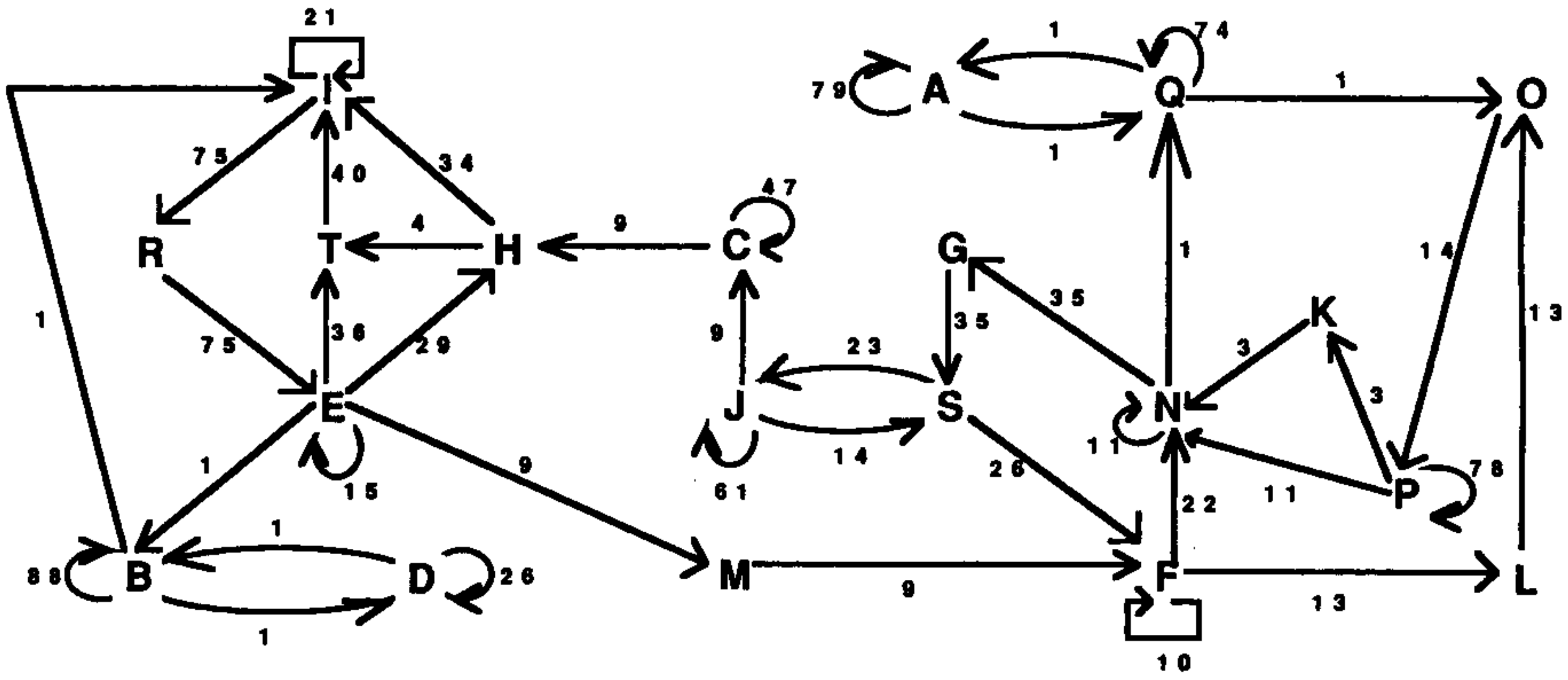


Figure 6 (b): Single Product Campaigns.



CYCLE TIME - 9017.78 Hrs.

Figure 7(a) : Solution to LP1 for example 2.



CYCLE TIME = 9018.92 Hrs.

Figure 7(b) : Optimal cycle for example 2

●— H -- 21(I) -- 28 (I-R-E-H) -- 37 (I-R-E-T) --

I - R - E - 15 (E) - M - 10 (Fi -

ll (F-L-O-P-N-G-S) - 2 (F-L-O-P-K-N-G-S) --

13 (F-N-G-S) -- F - 12 (N) - 26 (Q) - 80 (A) --

Q -- O - P - K - N - G - S -

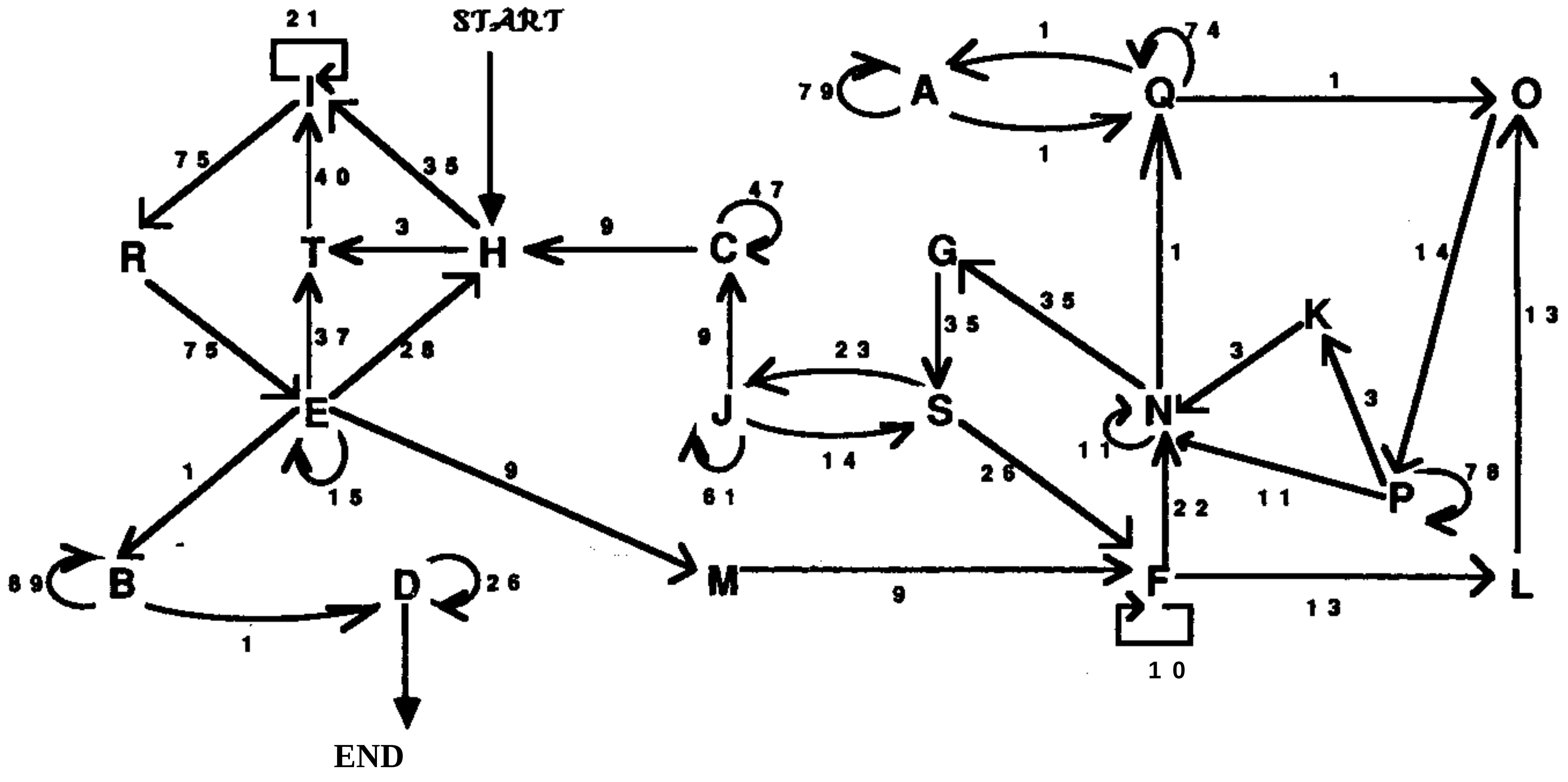
14 (J-S) -- J - 61 (J) - C - 47 (C) -- H -

5 (I-R-E-M-F-N-G-S-J-C-H) -- 3 (I-R-E-M-F-N-G-S-J-C-H-T)

- | _ R - E - - B ~ 8 9 (B) ~ D - 2 6 (D) -*

MAKESPAN = 9035.95 Hrs.

Figure 8 : Global Optimum Makespan for Example 2
(b) Detailed Schedule



MAKESPAN = 9035.95 Hrs.

Figure 8 : Global Optimum Makespan for Example 2
(a) Graph Representation