Large scheduling problems with bivalent costs

N. Christofides

Department of Management Science, Imperial College of Science and Technology, Exhibition Road, London SW7 2BX

Consider *n* items to be produced on one facility in a cyclic manner, and assume the costs of resetting the facility (so that an item can follow immediately after the completion of the previous one), to be bivalent (two-valued); i.e., either resetting of the facility is necessary and the cost is *W*, or no resetting is necessary and the cost is zero. The problem considered in this paper is that of scheduling the *n* items so as to minimise the total resetting cost incurred. This problem is a special case of the well known travelling salesman problem and could be solved as such, although only small size problems could be treated in this way. The present paper describes an alternative formulation of the problem in graph-theoretic terms, which enables the solution of large size problems (involving several hundreds or thousands of items). Computational results are given for six problems varying in size from 50 to 500 items.

(Received April 1972)

In a number of industries, especially the chemical and pharmaceutical industries, the following basic scheduling problem arises: A number (say n) of items is to be manufactured using a single processing facility or reaction vessel. The facility (vessel), may or may not have to be reset (cleaned), after item p_i has been manufactured (but before production of p_j is started); depending on the item combination (p_i, p_j). The cost of resetting the facility is constant regardless of the item p_i that has just been produced or the item p_j that is to follow; and, of course, there is no cost incurred if no resetting of the facility is required. Suppose that these n items are to be manufactured in a continuous cyclic manner, so that after the production of the last of the n items the manufacture of the first item in the fixed cycle is started again. The problem is to find a sequence of manufacturing these items which incurs the least additional resetting costs.

The above problem can arise in two ways; either the resetting costs are in reality independent of the items, or, alternatively, detailed cost data is not available and an average constant value is taken as an approximation. A number of variants to the above problem exist and appear quite frequently in practice. Thus, the n items may be required to be manufactured just once and not as a continuous product cycle; or more than one facility may be available for the manufacture of the items; or, alternatively, the required quantity of some item p_i may be so large that not all of it can be manufactured in a single operation (i.e. the quantity may be greater than the capacity of the facility or vessel), and multiple operations are therefore necessary to produce the required amount.

The basic scheduling problem described above can be recognised immediately as a special case of the well known travelling salesman problem (Little *et al*, 1963; Belmore and Malone, 1971; Eilon, Watson-Gandy and Christofides, 1971; Christofides, 1970). Thus what is required is a travelling salesman tour (cycle), through the n items which incur the least total cost. A travelling salesman tour such as $p_{i_1}, p_{i_2}, p_{i_3}, \ldots, p_{i_n}$ is then taken to mean that this is the sequence in which the items are to be manufactured. The cost matrix is $C = [c_{ij}]$, where c_{ij} is the cost of resetting the facility when the production of p_j follows that of item p_i. C is therefore a bivalent matrix whose entries have values either zero or some constant quantity W (say).

Since a variety of algorithms for the solution of travelling salesman problems exists in the literature, it may at first sight be thought that the present problem need not be considered further. However, since all present-day algorithms for the optimal solution of general travelling salesman problems are limited to problems containing less than about 70 points, and since practical problems of the scheduling type discussed in this paper often involve hundreds and sometimes thousands of products, the existing algorithms are of little value. Moreover, it is a well known (although perhaps unexpected), fact (Little *et al*, 1963; Bellmore and Malone, 1971; Eilon, Watson-Gandy and Christofides, 1971) that algorithms of the branch and bound variety (which are the most efficient methods of solution of travelling salesman problems), are at their peak efficiency for problems with random asymmetrical cost matrices $[c_{ij}]$, and that their performance becomes worse as the cost matrices become more and more structured. Thus the present problem with a bivalent cost matrix may be expected to be a 'difficult' problem as far as the branch and bound methods are concerned!

The present paper formulates the bivalent cost scheduling problem as a graph-theoretic problem and gives an algorithm for the optimal solution of large scale problems, involving several hundreds or thousands of products. The increase in the computational effort of the proposed method is more or less linearly related to the problem size, (n), as compared with the exponential (Little *et al*, 1963) or high order monomial (Bellmore and Malone, 1971), increase noted for the branch and bound techniques.

Problem formulation

Consider a graph $G = (X, A)$ where X is the set of vertices of the graph, and A its set of arcs. Let the graph G be defined so that there is a vertex x_i corresponding to every item p_i to be scheduled, and that an arc (x_i, x_j) is in the set of arcs A if and only if no resetting of the facility is necessary when the manufacture of item p_j follows that of item p_i.

We will start by giving the following definition.

A hamiltonian circuit of G is a circuit which uses only arcs that are in the set A of arcs, and which passes through every vertex of G once and once only.

Now, if the graph $G = (X, A)$ possesses a hamiltonian circuit say $x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_{i_n}$ then the corresponding sequence of items $p_{i_1}, p_{i_2}, p_{i_3}, \ldots, p_{i_n}$ can be manufactured by the facility without any resetting, since by definition of the hamiltonian circuit $(x_{i_k}, x_{i_{k+1}}) \in A$ for all $k = 1, 2, \ldots, n-1$ and A has been defined as the set $\{(x_i, x_j) | c_{ij} = 0\}$.

If G possesses no hamiltonian circuit, we can construct the graph $G_1 = (X_1, A_1)$ where:

$$X_1 = X \cup \{y_1\}$$

and

$$A_1 = A \cup \{(x, y_1) | x \in X\} \cup \{(y_1, x) | x \in X\}$$

i.e. a dummy vertex y_1 is introduced into G together with arcs leading to, and from, it to every other real vertex of G.

If the graph G_1 possesses a hamiltonian circuit, this will have the form $x_{i_1}, x_{i_2}, \ldots, x_{i_r}, \boxed{y_1}, x_{i_{r+1}}, \ldots, x_{i_n}$, which can be interpreted to mean that the items can be manufactured in a sequence:

$$p_{i_{r+1}}, p_{i_{r+2}}, \ldots, p_{i_n}, p_{i_1}, p_{i_2}, \ldots, p_{i_r}$$

with a single facility resetting operation between the finishing of the last and beginning of the first item in the sequence. Thus the dummy vertex serves the purpose of a marker indicating the position in the sequence where a facility resetting is necessary. In terms of the graph, the dummy vertex y_1 and its associated arcs provide a path between any two real vertices. Thus if an item sequence with one resetting operation exists, i.e. if a hamiltonian circuit would exist in G provided that one 'arc' $(x_i, x_j) \notin A$ could be used, the addition of y_1 to G will always cause a hamiltonian circuit to exist in G_1 since the extra 'arc' (x_i, x_j) needed can be replaced by the two dummy arcs (x_i, y_1) and (y_1, x_j).

If the graph G_1 possesses no hamiltonian circuit we construct the graph $G_2 = (X_2, A_2)$ from the graph G_1, where:

$$X_2 = X_1 \cup \{y_2\}$$

and

$$A_2 = A_1 \cup \{(x, y_2) | x \in X\} \cup \{(y_2, x | x \in X\}$$

and continue in the same way.

From the above argument the following theorem immediately results.

Theorem

If the graph $G_m = (X_m, A_m)$ given by:

$$X_m = X \cup \left[\bigcup_{j=1}^{m} \{y_j\} \right] \tag{1}$$

and

$$A_m = A \cup \left[\bigcup_{j=1}^{m} \{(x, y_j) | x \in X\} \right] \cup \left[\bigcup_{j=1}^{m} \{(y_j, x) | x \in X\} \right] \tag{2}$$

contains a hamiltonian circuit, but the graph G_{m-1} defined in a similar way does not, then m is the minimum number of facility resettings that are necessary, and if the hamiltonian circuit of G_m is:

$$x_{i_1}, \ldots, x_{i_\alpha}, \boxed{y_1}, x_{i_{\alpha+1}}, \ldots, x_{i_\beta}, \boxed{y_2}, x_{i_{\beta+1}}, \ldots, x_{i_\gamma}, \boxed{y_3}, x_{i_{\gamma+1}}, \ldots, \text{etc.}, \ldots, x_{i_\delta}, \boxed{y_m}, x_{i_{\delta+1}}, \ldots, x_{i_n}$$

then the products should be produced in m sequences given by:

$p_{i_{\alpha+1}}, \ldots, p_{i_\beta}$ followed by $p_{i_{\beta+1}}, \ldots, p_{i_\gamma} \ldots$ etc. . . .
followed by $p_{i_{\delta+1}}, \ldots, p_{i_n}, p_{i_1}, \ldots, p_{i_\alpha}$

Proof

The proof follows immediately by induction from the argument preceding the theorem.

The algorithm

In the argument of the preceding section the graph G was increased by a single dummy vertex at a time. If the optimum solution to the problem involves m resettings of the facility, then $m + 1$ attempts have to be made to find hamiltonian circuits in the graphs $G, G_1, \ldots, G_m$, with only the last one of these attempts being successful and leading to a solution of the problem. Obviously m is bounded from above by n and in general for practical problems m will only be a small fraction of n. Nevertheless, a different procedure for generating and testing the graphs for hamiltonian circuits is, in fact, necessary from the computational point of view, since it turns out (see next section), that it is much faster to find a hamiltonian circuit in a graph that possesses one rather than prove that no hamiltonian circuit exists in a graph that does not possess one. This fact immediately suggests that a better algorithm is one which starts with an upper bound B for the optimal (minimal) number of facility resettings m, and sequentially forms and tests the graphs G_B, G_{B-1}, etc. until a graph G_{m-1} is found which possesses no hamiltonian circuit.

A short description of such an algorithm is given below.

```
Step 1. Find an upper bound $B$ to the optimal (minimal), number of facility resettings that may be necessary. (See Appendix.)
Step 2. Form the graph $G_B$ according to equations (1) and (2).
Step 3. Does $G_B$ possess a hamiltonian circuit? If yes store the circuit in vector $H$ overwriting any previous circuit stored there and go to Step 4, else go to Step 5.
Step 4. $B \leftarrow B - 1$, go to Step 2.
Step 5. Stop. $m = B + 1$ is the minimal number of facility resettings required and the last sequence in $H$ is the required item manufacturing sequence.
```

Steps 1 and 3 of the above algorithm require further explanation. Obviously, the tighter the initial upper bound B is, the fewer will be the number of iterations of the main algorithm. In the Appendix a procedure for calculating a good intitial upper bound is given; whereas in Selby (1970) and Christofides (1973) an efficient algorithm, which finds a hamiltonian circuit in an arbitrary directed graph, is described (corresponding to Step 3 of the main algorithm above).

Computational results

The efficiency of the algorithm in solving large scale scheduling problems of the bivalent type was tested on six problems involving 50, 100, 200, 300, 400 and 500 products to be scheduled. These were randomly generated problems, with an average of about four items being able to follow immediately after the manufacture of a given item has been completed and without resetting of the production facility being necessary.

Table 1 gives the experimental results for these problems. All times shown in this table are seconds on the CDC 6600 computer at Imperial College. The computing times are further broken down into the time expended in finding a hamiltonian circuit in the graphs which possess one (i.e. graphs $G_B, G_{B-1}, \ldots, G_m$) and that expended in order to show that G_{m-1} does not possess a hamiltonian circuit i.e. to prove that m is indeed the optimal answer to the problem. From these timings it can be seen that, if, for example, only twice the time taken to find a hamiltonian circuit in G_k is allowed for searching for a hamiltonian circuit in G_{k-1}, and if a circuit is not found during this time, assuming that G_{k-1} does not possess one,

Table 1 Computational results for six scheduling problems

NUMBER OF PRODUCTS	OPTIMAL SOLUTION* (m)	INITIAL UPPER BOUND* (B)	COMPUTING TIMES (SEC.) TOTAL	UP TO m†	FOR $(m-1)$§
50	3	4	9·51	0·31	9·2
100	8	10	15·9	0·73	15·2
200	12	14	12·8	1·36	11·4
300	15	16	25·2	1·97	23·2
400	15	17	25·9	4·42	21·5
500	16	19	40·4	3·80	36·6

*Number of facility resettings.
†Time taken to reach optimal solution.
§Time taken to prove that m is indeed the optimal solution.

then more than 90% of the computing time would be saved, and one would still have a good chance of ending with the optimal answer (although this, of course, could not be guaranteed).

Appendix

The following is an algorithm for calculating an upper bound B on the minimum number of facility resettings required.

```
Step 0.  index ← 0, $B \leftarrow 0$
Step 1.  Set labels $l(x_i) = 0 \; \forall \; x_i \in X$. Set $p = 0$.
Step 2.  $G = (X, A)$, choose any $x_0 \in X$, set $S = \{x_0\}$.
Step 3.  If index = 0 form $\bar{S} = S \cup \Gamma S$; else form $\bar{S} = S \cup \Gamma^{-1} S$
         (Where: $\Gamma x_i = \{x_j | (x_i, x_j) \in A\}$ and $\Gamma S = \bigcup_{x_i \in S} \Gamma x_i$
          and also:
          $\Gamma^{-1} x_i = \{x_j | (x_j, x_i) \in A\}$ and $\Gamma^{-1} S = \bigcup_{x_i \in S} \Gamma^{-1} x_i$)
Step 4.  If $\bar{S} = S$ go to Step 5, else $p \leftarrow p + 1$, $l(x_i) \leftarrow p \; \forall \; x_i \in \bar{S} - S$, $S \leftarrow \bar{S}$ and return to Step 3.
Step 5.  Find one $x \in \{x_i | l(x_i) = p\}$
Step 6.  If index = 0 find an $x' \in \{x_i | l(x_i) = p - 1$ and $(x', x) \in A\}$.
         else: find an $x' \in \{x_i | l(x_i) = p - 1$ and $(x, x') \in A\}$.
Step 7.  $X \leftarrow X - \{x\}$, $A \leftarrow A - \{(x, x_i) | x_i \in X\} - \{(x_i, x) | x_i \in X\}$.
         If $X = \{x_0\}$, $B \leftarrow B + 1$ and go to Step 12; else go to Step 8.
Step 8.  $x \leftarrow x'$, $p \leftarrow p - 1$. If $x' = x_0$ go to Step 9 else go to Step 5.
Step 9.  If index = 0, index ← 1 and go to Step 1; else go to Step 10.
Step 10. If $X = \{x_0\}$, $B \leftarrow B + 1$ and go to Step 12; else go to Step 11.
Step 11. index ← 0, $B \leftarrow B + 1$.
         $X \leftarrow X - \{x_0\}$, $A \leftarrow A - \{(x_0, x_i) | x_i \in X\} - \{(x_i, x_0) | x_i \in X\}$.
         Return to Step 1.
Step 12. Stop. $B$ is the required upper bound.
```

The algorithm requires some explanation. When index = 0 *forward* paths are traced through the vertices of G starting with vertex x_0. These paths are traced by labelling with p those vertices of G that require p arcs to be reached from x_0. (Steps 1, 2, 3 and 4.) When none of these paths can be extended, the algorithm proceeds to Steps 5, 6 and 7 which trace the longest of these paths backwards to the vertex x_0 erasing from the graph vertices (and their associated arcs), which lie on the longest path. Step 8 terminates the erasing process. Step 9 returns the algorithm to the beginning with index = 1 to start forming *backward* paths, i.e. paths terminating at x_0. Again the longest of these is found and erased.

The longest of the forward and backward paths (to or from x_0), can be considered together as a single long path which contains x_0. The number B (of path sequences necessary) is then increased by unity at Step 11, and the process continued by choosing another vertex x_0 from the remaining graph to form new longest forward and backward sequences until the graph is exhausted.

The final value of B, which is the number of path sequences used to erase the whole graph (i.e. cover all the vertices), is then, obviously, an upper bound on the required minimum number of such covering sequences (i.e. on the minimum number of facility resettings required).

Acknowledgements

The author wishes to acknowledge with thanks the use of a computer program, written by Dr. G. R. Selby, for determining hamiltonian circuits in undirected graphs. This program has been modified by the present author to deal with directed graphs, and forms a basic step in the algorithm given in this paper.

References

Bellmore, M., and Malone, J. C. (1971). Pathology of travelling salesman, subtour-elimination algorithms. *Ops. Res.*, Vol. 19, p. 278.
Christofides, N. (1970). The shortest hamiltonian chain of a graph, *Journal of SIAM* (Applied Maths.), Vol. 19, No. 4, p. 689.
Christofides, N. (1973). *Graph Theory—An algorithmic approach.* Academic Press. To appear.
Eilon, S., Watson-Gandy, C. D. T., and Christofides, N. (1971). *Distribution Management* (Chapter 7), Griffin, London.
Little, J. D. C., Murty, K. G., Sweeny, D., and Karel, C. (1963). An algorithm for the travelling salesman problem. *Ops. Res.*, Vol. 11, p. 979.
Selby, G. R. (1970). *The use of topological methods in computer-aided circuit layout.* Ph.D. Thesis, London University.