Journal of the Operational Research Society (1996) 47, 151–161

Solving the CLSP by a Tabu Search Heuristic

K. S. HINDI

University of Manchester Institute of Science and Technology (UMIST)

The multi-item, single-level, capacitated, dynamic lot-sizing problem, commonly abbreviated as CLSP, is considered. The problem is cast in a tight mixed-integer programming model (MIP); tight in the sense that the gap between the optimal value of MIP and that of its linear programming relaxation (LP) is small. The LP relaxation of MIP is then solved by column generation. The resulting feasible solution is further improved by adopting the corresponding set-up schedule and re-optimizing variable costs by solving a minimum-cost network flow (trans-shipment) problem. Subsequently, the improved solution is used as a starting solution for a tabu search procedure, with the worth of moves assessed using the same trans-shipment problem. Results of computational testing of benchmark problem instances are presented. They show that the heuristic solutions obtained are effective, in that they are extremely close to the best known solutions. The computational efficiency makes it possible to solve realistically large problem instances routinely on a personal computer; in particular, the solution procedure is most effective, in terms of solution quality, for larger problem instances.

Key words: production planning, capacitated lot-sizing, tabu search, network flows

INTRODUCTION

In many manufacturing environments, several items share a single common constrained resource. Such a case arises in the process industry when a number of items are to be made from a single common base material or compound. In other contexts, the capacity constraint may represent, in aggregate, a number of limited resources. Set-up costs and maintenance costs are aggregated into a fixed cost that is incurred whenever there is production. Variable costs are a per unit production cost and a per unit inventory holding cost. The demand for each item in each time period is known and it is required to determine the lot sizes for each item in each period, such that total cost is minimized and the capacity constraint is respected in each period. The resulting problem is known as the multi-item, single-level, capacitated, dynamic lot-sizing problem, commonly abbreviated as CLSP.

A conventional model is:

Problem P
$$\min \sum_{i=1}^{N} \sum_{t=1}^{T} (r_{it}y_{it} + c_{it}x_{it} + h_{it}I_{it}) \tag{1}$$

subject to
$$\sum_{i} a_i x_{it} \leqslant R_t \qquad \forall t \tag{2}$$

$$I_{i,t-1} + x_{it} - I_{it} = d_{it} \qquad \forall i, t \tag{3}$$

$$x_{it} \leqslant M_{it} y_{it} \qquad \forall i, t \tag{4}$$

$$y_{it} \in \{0, 1\}, \qquad x_{it} \geqslant 0, \qquad I_{it} \geqslant 0 \qquad \forall i, t \tag{5}$$

where the decision variables are

x_{it} = the amount of item i produced in period t,

$$y_{it} = \begin{cases} 1 & \text{if item } i \text{ is produced in period } t, \\ 0 & \text{otherwise,} \end{cases}$$

I_{it} = the amount of item i held in inventory from period t to period $t + 1$,

and the parameters are

Correspondence: K. S. Hindi, Department of Computation, University of Manchester Institute of Science and Technology (UMIST), PO Box 88, Manchester M60 1QD, UK

r_{it} = fixed cost for producing item i in period t;
c_{it} = per unit cost of producing item i in period t;
h_{it} = per unit cost of holding item i in inventory from period t to period $t+1$;
d_{it} = demand for item i in period t;
M_{it} = upper limit on the production of item i in period $t = \sum_{s=t}^{T} d_{is}$;
a_i = rate of resource consumption by item i;
R_t = amount of resource available in period t.

The objective function (1) minimizes total set-up costs, production costs and inventory holding costs. Constraints (2) are the capacity constraints and (3) are the flow-balance constraints. Constraints (4), along with the integrality requirement on y_{it}, enforce a set-up for item i in any period with positive production for that item. The non-negativity constraints on I_{it} mean that backlogging is not allowed.

The computational complexity of the single-item version of CLSP has been proven to be NP-hard.[1,2] Since CLSP is more general, it follows that it is also NP-Hard. It is also difficult in the sense that it is hard to solve realistically large problem instances in reasonable computation times. This has led to the development of many heuristic algorithms. Since good comparative surveys can be found in References 3 and 4, only an overview will be given here. However, two approaches with some kinship to the heuristic developed in this work, namely the Lagrangian-relaxation based heuristic approach and the set partitioning approach, will be discussed in a subsequent section.

Heuristic algorithms for solving CLSP can be divided into two major classes; 'common sense' heuristics and mathematical-programming based heuristics. The former can, in turn, be divided into single-pass constructive heuristics and improvement heuristics.

Single-pass constructive heuristics, examples of which can be found in References 5–7, proceed period by period. Priority indices are first calculated for each product/future-period combination to estimate potential savings that might accrue from producing in the current period to satisfy a future demand. Different algorithms employ different indices, but all proceed by using the indices to include future demand in the production of the current period until capacity is reached or until all future lots with positive indices are exhausted. Obviously, there is no guarantee that a feasible solution will be found if only lots with positive indices are considered, since a period may be encountered where total net demand exceeds capacity. Different algorithms differ in the way feasibility is ensured.

Improvement heuristics, examples of which can be found in References 8 and 9, start with a solution found by ignoring capacity constraints and applying either the Wagner–Whitin algorithm or some uncapacitated lot-sizing heuristic. Thereafter, improvement steps fall into two categories: feasibility enforcement steps, which attempt, on the basis of aggregate feasibility conditions to attain a feasible solution while minimizing additional cost; and cost improvement steps, which attempt to reduce cost without introducing new infeasibilities.

Since cost improvement heuristics apply to feasible schedules, there is clearly scope for combining them with single-pass constructive heuristics. This can be done either by applying improvement steps once the pass is executed, or by applying these steps to partial schedules while the pass is proceeding.

Mathematical-programming based heuristics include those based on Lagrangian relaxation and set partitioning, which will be discussed later. Other examples include the heuristic of Newson[10,11], which is based on releasing capacity in violated periods by calculating the next best Wagner–Whitin solutions. However, this approach suffers from two drawbacks: first, the search is restricted to schedules that have the Wagner–Whitin property, whereas the optimal solution may not have this property; secondly, it may end without discovering a feasible solution. The truncated branch-and-bound procedure of Reference 12, where lower bounds are calculated via Lagrangian relaxation, is also an example of mathematical-programming based heuristics.

The heuristic developed in this work belongs to the class of mathematical-programming based heuristics. An initial solution is found by solving an optimization problem. Thereafter an attempt is made to improve this solution by searching the solution space according to the rules of tabu search.

INITIAL SOLUTION AND CALCULATION OF A LOWER BOUND

As will be described in the following section, the proposed tabu search heuristic is started with an initial feasible solution. Also, one of the stopping criteria employed is to terminate the search as soon as a solution within a prespecified distance from a known lower bound is reached. In this section, finding the initial solution and calculating the lower bound are presented. Both are based on an alternative formulation of CLSP which, in turn, is based on exploring the relationship between CLSP and the problem resulting from deleting the capacity constraints, i.e. the uncapacitated problem.

Relaxing the capacity constraints (2) leads to the break up of Problem P into N single-item, uncapacitated lot-sizing problems. As is well known, the optimal (Wagner–Whitin) solution of each such problem[13] is characterized by the fact that $I_{i,t-1}x_{it} = 0$. Thus, the demand at a period is completely supplied either from inventory or production, i.e. when production occurs, it covers an integer number of consecutive periods. Schedules with the property $I_{i,t-1}x_{it} = 0$ are called dominant schedules.

For each item i, the dominant schedules can be represented as paths from node 0 to T in an acyclic, directed graph $G_i = (U_i, E_i)$, where $U_i = \{0, 1, \ldots, T\}$ is the set of nodes and $E_i \subset U_i \times U_i$ is the set of edges (st), $0 \leq s < t \leq T$. Each edge $st \in E_i$ represents production in period $s + 1$, which satisfies the demand for periods $s + 1$ through to t. An example graph is given in Figure 1.

The lot size associated with an edge is given by:

$$X_{ist} = d_{i,s+1} + \ldots + d_{it}$$

and the corresponding inventory holding cost H_{ist} is computed as

$$H_{ist} = \sum_{p=s+1}^{t} h_{ip}F_{ip}$$

where F_{ip} is calculated recursively by $F_{is} = X_{ist}$, and $F_{ip} = F_{i,p-1} - d_{ip}$, $p = s + 1, \ldots, t$. The total cost associated with edge st is then

$$\gamma_{ist} = r_{i,s+1} + c_{i,s+1}X_{ist} + H_{ist}. \tag{6}$$

Computing the optimal solution of the single-item, uncapacitated lot sizing problem corresponding to item i is then equivalent to finding the shortest path in graph G_i. From network flow theory, this is equivalent to solving the uncapacitated, minimum-cost network flow problem on G_i with a unit of demand at node T. Thus, if f_{ist} denotes the flow in edge $st \in E_i$, the shortest-path problem, for each item i, can be expressed as:

$$\min \sum_{s=0}^{T-1} \sum_{t=s+1}^{T} \gamma_{ist} f_{ist} \tag{7}$$

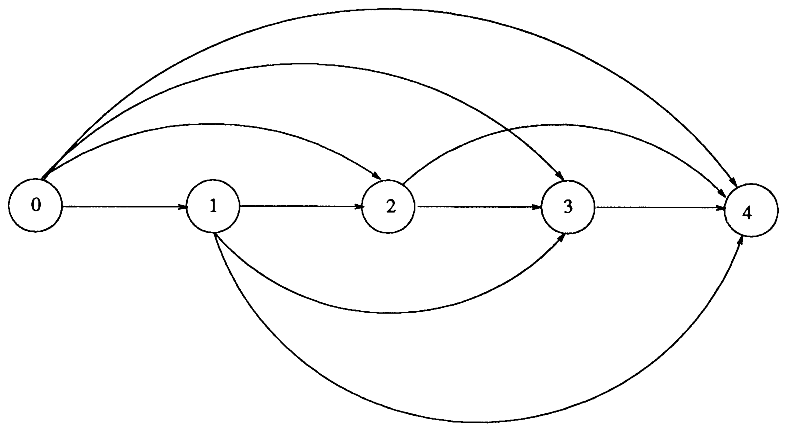

FIG. 1. *Shortest path graph corresponding to a single item uncapacitated problem for four periods.*

subject to
$$\sum_{s=0}^{t-1} f_{ist} - \sum_{s=t+1}^{T} f_{its} = 0 \quad \text{for } t = 1, \ldots, T-1 \tag{8}$$

$$\sum_{s=0}^{T-1} f_{isT} = 1 \tag{9}$$

$$f_{ist} \in \{0, 1\} \quad \forall s, t. \tag{10}$$

However, when capacity constraints are imposed, the optimal flows become fractional, i.e. $0 \leqslant f_{ist} \leqslant 1$. Under this latter condition, production of item i in period t becomes

$$x_{it} = \sum_{s=t}^{T} X_{i,t-1,s} f_{i,t-1,s}$$

and the corresponding resource consumption is equal to $\sum_{s=t}^{T} a_{it} X_{i,t-1,s} f_{i,t-1,s}$. Problem P is, thus, transformed to the following.

Problem P1
$$\min \sum_{i=1}^{N} \sum_{s=0}^{T-1} \sum_{t=s+1}^{T} \gamma_{ist} f_{ist} \tag{11}$$

$$\sum_{i} \sum_{s=t}^{T} a_{it} X_{i,t-1,s} f_{i,t-1,s} \leqslant R_t \quad \forall t \tag{12}$$

$$\sum_{s=0}^{t-1} f_{ist} - \sum_{s=t+1}^{T} f_{its} = 0 \quad \forall i, t = 1, \ldots, T-1 \tag{13}$$

$$\sum_{s=0}^{T-1} f_{isT} = 1 \quad \forall i \tag{14}$$

$$\sum_{s=t}^{T} f_{i,t-1,s} \leqslant y_{it} \quad \forall i, t \tag{15}$$

$$y_{it} \in \{0, 1\}, \quad f_{ist} \geqslant 0 \quad \forall i, s, t. \tag{16}$$

Problem P1 represents a tighter formulation of problem CLSP than Problem P, in the sense that the LP relaxation of Problem P1 gives a sharper lower bound on the optimal value of the objective function of CLSP. This is due to the fact that the linear programming relaxation of a shortest path problem yields the same solution as that of the problem itself (zero gap), which means that the linear programming relaxation of Problem P1 is equivalent to the convex hull relaxation of CLSP[14].

The transformation of Problem P to Problem P1 closely follows that suggested in Reference 15. There is, however, one crucial difference; whereas set-up costs are attached to the set-up variables in Reference 15, they are here incorporated in γ_{ist}, the costs of the edges of the shortest path graphs. This makes it possible to solve for the lower bound much more efficiently. Letting y_{it} vary continuously between 0 and 1 for all i, t, leads to:

Problem P2
$$\min \sum_{i=1}^{N} \sum_{s=0}^{T-1} \sum_{t=s+1}^{T} \gamma_{ist} f_{ist} \tag{17}$$

$$\sum_{i} \sum_{s=t}^{T} a_{it} X_{i,t-1,s} f_{i,t-1,s} \leqslant R_t \quad \forall t \tag{18}$$

$$\sum_{s=0}^{t-1} f_{ist} - \sum_{s=t+1}^{T} f_{its} = 0 \quad \forall i, t = 1, \ldots, T-1 \tag{19}$$

$$\sum_{s=0}^{T-1} f_{isT} = 1 \quad \forall i, \tag{20}$$

the optimal value of the objective function of which constitutes the desired lower bound.

Importantly, solving Problem P2, amounts to finding the cheapest convex combination of the solutions corresponding to the paths in graphs G_i, $i = 1, \ldots, N$, which satisfies the capacity constraints. Thus, Problem P2 can be solved very efficiently by a column generation strategy (Dantzig-Wolfe), with constraints (18) as the linking constraints and the subproblems separating into as many shortest path problems as there are items.

Since the shortest path problems have to be solved, within the framework of the Dantzig-Wolfe procedure a number of times, it is essential for them to be solved most efficiently. Fortunately, this is possible due to the special structure of the graph; it is without circuits and $s < t$ for all arcs (st). Thus, if G_i is constructed such that arcs occur in the arc list in the order of their tails, the shortest path can be calculated in $O(E_i)$ by the following simple algorithm, which follows from the optimality principle of dynamic programming (l_{st} = length of arc (st)):

```
for $j := 1$ to $T$   distance $[j] := \infty$;
for each arc $(st)$ in turn, if distance $[s] + l_{st} <$ distance $[t]$ then
    distance $[t] :=$ distance $[s] + l_{st}$ and predecessor $[t] := (st)$.
```

The path can then be found by tracing the predecessors back from T in $O(T)$.

Solution of Problem P2 not only provides a good lower bound, but also a feasible solution. Since the production quantities x_{it}, calculated from the solution, are such that the capacity constraints are satisfied, they can be adopted to effect a complete feasible schedule and, therefore, a starting solution. However, in solving Problem P2, the x_{it} values are calculated effectively on the basis of the 'hybrid' costs γ_{ist}, which incorporate fixed cost elements. Therefore, a better production schedule, and consequently a better overall feasible solution, may be obtained by re-optimizing excluding fixed costs. To this end, let I be the set of it indices corresponding to production, as indicated by the solution of Problem P2, and $\bar{I}$ be the complement of I, then problem CLSP can be written as (M = a sufficiently larger number):

Problem P3
$$\sum_{it \in I} r_{it} + \min \sum_{it \in I} (c_{it} x_{it}) + \sum_{it \in \bar{I}} M x_{it} + \sum_{it} h_{it} I_{it} \tag{21}$$

subject to
$$\sum_i a_i x_{it} \leqslant R_t \qquad \forall t \tag{22}$$

$$I_{i,t-1} + x_{it} - I_{it} = d_{it} \qquad \forall i, t \tag{23}$$

$$x_{it} \geqslant 0 \qquad \forall i, t. \tag{24}$$

Defining $x'_{it} = a_i x_{it}$ and substituting throughout, it can be readily seen that Problem P3 is an uncapacitated minimum-cost network flow (trans-shipment) problem,[4] which can be solved very efficiently[16–19]. An example trans-shipment problem, corresponding to two items and three periods, is given in Figure 2.

In addition to finding a starting solution, the uncapacitated trans-shipment Problem P3 constitutes the basis of the tabu search heuristic. As will be described in the next section, it will be used to evaluate the worth of a move, with the latter being defined as transferring an it from I to $\bar{I}$ or vice versa.

TABU SEARCH HEURISTIC

Although the roots of tabu search date back to the late 1960s and early 1970s, it was proposed in its 'modern form' and publicized by Glover[20,21]. Hansen[22] has also sketched out the ingredients of tabu search independently. A good introduction can be found in Reference 23.

Tabu search is an iterative method for finding, in a (finite or infinite) set X of feasible solutions, a solution s^* that minimizes a real valued objective function f. A neighbourhood $N(s)$ is defined for each solution $s \in X$. Starting from an initial feasible solution, the neighbourhood $N(s^c)$ of the current solution s^c is examined and the solution s' with the best objective function is chosen as the next solution, i.e. $s' | f(s') < f(s'')$, s', $s'' \in N(s^c)$. The

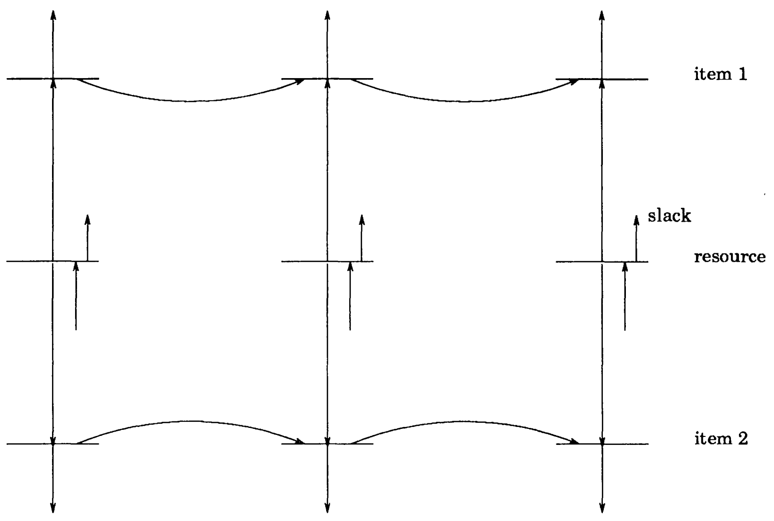

FIG. 2. *Trans-shipment network corresponding to Problem P3 for two items and three periods.*

process could be conceived of as one of steepest descent, mildest ascent (compare Reference 22). The fact that movement from an s^c to an $s' \in N(s^c)$ is allowed even if $f(s') \geq f(s^c)$ helps escape from local optima.

However, with the above scheme cycling is possible. To prevent cycling, a structure called tabu list L of length l (fixed or variable) is introduced to prevent returning to a solution visited in the last l iterations; other ingredients of tabu search, such as aspiration criteria and intensification and diversification schemes, serve to enhance its efficiency and productivity. The tabu search process stops when the solution is close enough to a lower bound of f, if known. Otherwise, it stops when no improvement occurs over the best solution for a certain number of iterations.

Apart from finding a starting solution and calculating a good lower bound, both discussed in the previous section, the elements of the tabu search heuristic as applied here can be described as follows.

Neighbourhood and moves

A solution is defined by the item-period combinations corresponding to set-ups; i.e. by the *it* indices belonging to the set I. Thus, a natural definition of a move would be transferring an *it* from I to $\bar{I}$ or vice versa, and the corresponding neighbourhood would be the set of all production-feasible schedules that can be arrived at by a move. However, the neighbourhood thus defined is too large for realistically large problem instances (e.g. 2000 possible moves for a problem instance with 200 items and 10 periods) so as to make the search impracticable. Moreover, a computational study of a tabu search application[24] indicates that exploring complete neighbourhoods may be adequate for small problem instances, but is not, in general, an efficient use of computing resources. Overall it would appear that it is better to make fairly limited explorations in the search space and exploit the best of these, rather than pursue a lengthy search before a move is chosen. For these reasons, a restricted neighbourhood was employed, for larger problem instances, by compiling dynamically a candidate list of moves in the following way: at each tabu iteration, the *it* indices, such that the production is less than local demand, i.e. $0 < x_{it} < d_{it}$, are candidates for moves. This strategy, which proved highly effective, is motivated by the reasoning that set-ups associated with production which is less than local demand are likely to be very few in a good schedule.

Move evaluation

The worth of a move is evaluated by solving the uncapacitated trans-shipment problem (Problem P3) corresponding to the set-up schedule resulting from the move.

Size of tabu list

An effective size of the tabu list was found to be equal to the number of periods T. When the list is full, it is treated as a queue, i.e. the addition of a new element causes the oldest element to leave the list.

Aspiration

Aspiration criteria are introduced to determine when tabu restrictions can be overridden in order to enable a tabu search method to achieve its best performance. In the present work, aspiration by objective was used so that a move on the tabu list is permitted if it leads to an objective function better than that of the best solution found so far. Also, aspiration by default was employed so that when the candidate list is empty, that move on the tabu list that leads to the best cost (largest improvement or least deterioration), compared with the current solution, is made.

Stopping criterion

The search is terminated when a solution is discovered such that the cost gap with respect to the lower bound found at the beginning of the solution process, calculated as ((solution cost − lower bound)/lower bound), is less than 2%. Failing that, when no improvement occurs over the best solution for a number of iterations equal to $2T$, the search is stopped.

The following points are worth noting.

- Since the search scheme is based on repeatedly solving uncapacitated trans-shipment problems, overall efficiency is contingent upon how these are solved. Fortunately, there are very efficient algorithms for solving such problems[16–18]. The algorithm used in this work is the primal network Simplex algorithm[18] as implemented in Reference 19. Moreover, the trans-shipment problem is solved only once with all other solutions obtained by postoptimizing the most recent.
- The search scheme is a basic, 'no-frills' tabu search. Recourse to more sophisticated measures (e.g. multiple starts, diversification and intensification strategies) proved unnecessary.

RELATIONSHIP WITH OTHER HEURISTICS

The starting solution for the tabu search heuristic developed here has some kinship with two classes of existing heuristics; namely, Lagrangean-relaxation based heuristics and set-partitioning based heuristics.

Lagrangean relaxation[25,26] was applied in Reference 27. Relaxing the capacity constraints (2) leads to the break up of the problem into N single-item, uncapacitated lot-sizing problems. Applying subgradient optimization, with the relaxed problem solved as N shortest-path subproblems, is used to compute an effective lower bound. However, finding, by this approach, a solution feasible with respect to the relaxed capacity constraints is often impossible, particularly when the capacity is severely limited in relation to demands. To illustrate the point, consider a single item and a horizon of two periods and assume that the demand in the second period exceeds capacity. In this case, all production schedules corresponding to the uncapacitated lot-sizing solutions are infeasible with respect to the capacity constraint. Thus, it is impossible to find a feasible solution by Lagrangean relaxation, but this does not necessarily mean that the problem is infeasible, since it may be possible to construct a feasible production schedule by producing part of the demand of the second period in the first.

Thizy and Van Wassenhove[27] attempted to generate feasible schedules at each Lagrangian relaxation iteration by adopting the set-ups corresponding to the Lagrangian solution and solving a transportation problem to optimize variable costs while enforcing capacity (see below). However, if the arcs corresponding to a period where there is no set-up are deleted, this approach may fail to find a feasible solution (for example, it would fail in the case of the single-item, two-period problem posed above). On the other hand, if these arcs are retained, but given a large cost to encourage solutions without them, feasible solutions may be produced, but their quality will suffer due to the distortion of cost structures. Thus, the heuristic developed in Reference 27 will, at best, produce a solution with a quality comparable to that of our starting solution.

It is also instructive to consider the relationship between our starting solution and set partitioning heuristics, developed as early as 1958[29]. To explore this issue, consider a set of production schedules for each item and let

$$\theta_{ij} = \begin{cases} 1 & \text{if schedule } j \text{ is selected for item } i, \\ 0 & \text{otherwise,} \end{cases}$$

x_{ijt} = production quantity required by schedule j of item i in period t,

$$Y_{ijt} = \begin{cases} 1 & \text{if } x_{ijt} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

I_{ijt} = inventory quantity associated with schedule j of item i in period t,
b_{ij} = cost of schedule j of item $i = \sum_{t=1}^{T}(r_{it}Y_{ijt} + c_{it}x_{ijt} + h_{it}I_{ijt})$,
p_{ijt} = amount of capacity required in period t by schedule j of item $i = a_i x_{ijt}$.
The problem can be written as:

Problem P4
$$\min \sum_i \sum_j b_{ij}\theta_{ij} \tag{25}$$

subject to
$$\sum_j \theta_{ij} = 1 \qquad \forall i \tag{26}$$

$$\sum_i \sum_j p_{ijt}\theta_{ij} \leq R_t \qquad \forall t \tag{27}$$

$$\theta_{ij} \in \{0, 1\} \qquad \forall i, j. \tag{28}$$

Thus, the problem is transformed to that of selecting a schedule for each item and combining these schedules into a multi-item schedule such that the total cost is minimized and the capacity constraints respected.

In Reference 29, the dominant schedules having the property $I_{i,t-1}x_{it} = 0$ are used. Since there are $N \times 2^{T-1}$ such schedules, generating them for problems of realistic size is too time consuming and the resulting set partitioning problem is far too large to solve. Methods have also been developed[30,31] in which not all $N \times 2^{T-1}$ dominant schedules are generated explicitly and many schedules are handled implicitly by column generation. Also, in other work[32,33], schedules generated by fast uncapacitated lot-sizing heuristics are used. However, there is a structural problem with using only dominant schedules or schedules generated by uncapacitated lot-sizing heuristics. As pointed out earlier, the solution to the capacitated problem does not necessarily have the $I_{i,t-1}x_{it} = 0$ property, and when the capacity constraints are tight, the set partitioning model fails. In an effort to circumvent this problem, the work reported in Reference 3 uses, in addition, solutions generated by capacitated lot-sizing heuristics to ensure that the candidate solution set contains at least one feasible solution.

In contrast, our starting solution can be thought of as generating a series of multi-item schedules—each of which consists of dominant sub-schedules, one for each item—and then finding the optimal schedule by effecting a convex combination of the multi-item schedules such that the total cost is minimized and the capacity constraints respected. This is equivalent to solving the linear programming relaxation of the set partitioning Problem P4 above, achieved by replacing $\theta_{ij} \in \{0, 1\}$ by $\theta_{ij} \geq 0$, $\forall i, j$. Indeed both Dzielinski and Gomory[30] and Lasdon and Terjung[31] suggest solving the linear programming relaxation of the set partitioning problem rather than the set partitioning problem itself. Thus, it could be argued that our starting solution, although arrived at through redefinition of variables, is equivalent to that proposed in Reference 30 and 31.

COMPUTATIONAL RESULTS

A series of 17 benchmark problem instances have been used in computational testing. These are the same problem instances tackled by Eppen and Kipp Martin[15]. Table 1 shows the results obtained on a 33 MHz, 80486 IBM-compatible PC. The results shown under the heading 'MIP' are those obtained by formulating the problem as a mixed-interger programming problem using the variable redefinition method of Eppen and Kipp Martin, believed to be the most effective optimum seeking approach. The resulting MIP model was then solved using GAMS/OSL (OSL is IBM's state-of-the-art, general-purpose linear and mixed integer programming package). The calculations were stopped either when a solution guaranteed to be within a distance of 0.5% from the best possible solution was found or when CPU time exceeded 90 minutes (indicated in the table by 90+). The 'best cost' cited is either that quoted in Reference 15 or obtained using OSL, as just described, whichever is the smaller. The percentage gap between the solution obtained by tabu search and the best known cost is calculated as ((tabu cost − best cost)/best cost).

Tabu search was implemented as described above, except that for smaller problem instances (defined as those instances for which the product of items by periods is less than 64), the complete neighbourhood was searched.

The tabu search scheme achieves very good solutions indeed in very short times. In seven, out of the 17 problem instances, the gap between the tabu search solution and the best known is less than 0.5%; in four, it is less than 1%; in four, less than 2%. Only in the first two problem instances, which are unrealistically small (see the third remark below), did the gap exceed 2%. Moreover, solution times were all less than half a minute. This is in comparison with the solution times by OSL, which exceeded 1.5 hours in two instances and was 89 minutes in a third. Also, the memory requirement of the tabu search heuristic proved to be quite modest.

It is worth noting that in many instances, the need for search is obviated, since the starting solution satisfies the proximity to the lower bound stopping criterion. This was the case for problem instances 10 thorough 15 and for problem instance 17. Also, when the problem is infeasible, as was the case for problem 16, the fact is speedily discovered.

The following can be drawn from the results of computational testing.

1. The computational efficiency of the tabu search heuristic is such that large-scale problem instances can be solved routinely on a PC.
2. When the capacity constraints are tightened, solution time tends to increase, but not appreciably. This can be seen by comparing problem instances 1 through 4, where problem

TABLE 1. *Results for test problem instances*

Problem No.	Items	Periods	Best cost	MIP		Tabu		
				Solution cost	CPU (min)	Solution cost	Gap (%)	CPU (min)
1	8	8	8430.0	8430.0	6.78	8730.0	3.56	0.19
2	8	8	7910.0	7910.0	3.0	8250.00	4.30	0.29
3	8	8	7610.0	7610.0	0.91	7690.00	1.05	0.14
4	8	8	7520.0	7520.0	0.78	7520.00	0.0	0.004
5	50	8	48 690.0	48 690.0	16.65	49 001.9	0.64	0.1
6	100	8	94 379.7	97 130.0	90+	95 681.3	1.38	0.21
7	150	8	137 819.0	138 772.3	90+	139 755.20	1.41	0.37
8	200	8	188 122.0	189 646.3	89	191 274.40	1.68	0.44
9	20	13	5807.66	5812.0	3.05	5860.31	0.91	0.357
10	100	10	20 911.2	20 911.2	3.01	20 929.30	0.08	0.1
11	100	10	20 962.0	20 990.07	3.51	21 069.85	0.52	0.13
12	100	10	21 059.59	21 059.59	4.75	21 267.65	0.99	0.18
13	100	10	21 257.0	21 257.0	4.83	21 319.82	0.3	0.22
14	100	10	21 320.11	21 320.11	6.54	21 368.47	0.23	0.25
15	100	10	21 369.86	21 369.86	5.93	21 463.34	0.44	0.27
16	100	10	infeasible	infeasible	3.23	infeasible	0.0	0.23
17	200	10	43 650.0	43 650.0	12.52	43 809.36	0.23	0.39

instance 1 is the most stringent, and problem instances 10 through 15, where problem instance 15 is the most stringent.

3. When the number of items becomes large, the problem becomes easier, in the sense that the gap between the solution found and the optimum tends to become smaller. This phenomenon is due to the fact that when the number of items becomes large compared with the number of time periods, the problem becomes nearly a linear program[34]. The phenomenon of problem instances becoming easier as the number of items increases was observed quite early in References 29–31. It is also interesting to note that this same phenomenon occurs for a different lot sizing problem tackled by Trigeiro *et al.*[35]. In offering a rationale, they make the analogy that with few items, the scheduling problem has the character of trying to pack a few boulders of varying size into a set of containers, but scheduling many items is like filling containers with sand. They also observe that this phenomenon casts doubt on the validity of the common practice by researchers, where algorithms are tested on small problem instances and results extrapolated to larger ones, particularly when, from a practical viewpoint, interesting problem instances are the large ones.

SUMMARY AND CONCLUSIONS

The multi-item, single-level, capacitated, dynamic lot-sizing problem (CLSP) is NP-hard. It is also empirically difficult, in the sense that solving realistically large problem instances often requires long computation times. These facts justify seeking effective heuristic methods for its solution.

One such method has been presented in this work. The problem is cast in a tight mixed-integer programming model. The linear programming relaxation of this model is then solved by column generation. The resulting feasible solution is further improved by adopting the corresponding set-up schedule and reoptimizing variable costs, by solving a minimum-cost network flow (trans-shipment) problem. The improved solution is then used as a starting solution for a tabu search procedure, with the worth of moves assessed using the same trans-shipment problem.

The results of computational testing of benchmark problem instances show that the heuristic developed is effective; being capable of producing solutions of very high quality in very short computation times with modest memory requirements, making it possible to solve large problem instances routinely on a personal computer. In contrast with most other heuristic methods, the solution procedure guarantees finding a good feasible solution if the problem is feasible, and discovering infeasibility quickly if the problem is infeasible.

Acknowledgement—The author is grateful to Professor R. Kipp Martin of the University of Chicago for supplying the data set used for testing.

REFERENCES

1. G. R. Bitran and H. H. Yanasse (1982) Computational complexity of the capacitated lot size problem. *Mgmt Sci.* **28**, 1174–1186.
2. M. Florian, J. K. Lenstra and A. H. G. Rinnooy Kan (1980) Deterministic production planning: algorithms and complexity. *Mgmt Sci.* **26**, 669–679.
3. D. Cattrysse, J. Maes and L. Wassenhove (1990) Set partitioning and column generation heuristic for capacitated dynamic lotsizing. *Eur. J. Opl Res.* **46**, 38–47.
4. J. Maes and L. N. Van Wassenhove (1988) Multi-item single-level capacitated dynamic lotsizing heuristic: a general review. *J. Opl Res. Soc.* **39**, 991–1004.
5. P. S. Dixon and E. A. Silver (1981) A heuristic solution procedure for the multi-item, single level, limited capacity, lot sizing problem. *J. Opns Mgmt* **2**, 23–39.
6. P. S. Eisenhut (1975) A dynamic lotsizing algorithm with capacity constraints. *AIIE Trans.* **7**, 170–176.
7. M. R. Lambrecht and H. Vanderveken (1979) Heuristic procedures for the single operation, multi-item loading problem. *AIIE Trans.* **11**, 319–326.
8. A Dogramaci, J. E. Pansyiotopoulos and N. R. Adam (1981) The dynamic lot-sizing problem for multiple items under limited capacity. *AIIE Trans.* **13**, 294–303.

9. R. Karni and Y. Roll (1982) A heuristic algorithm for the multi-item lot sizing problem with capacity constraints. *IIE Trans.* **14**, 249–256.
10. E. F. P. Newson (1975) Multi-item lot-size scheduling by heuristic, part I: with fixed resources. *Mgmt Sci.* **21**, 1186–1193.
11. E. F. P. Newson (1975) Multi-item lot size scheduling by heuristic, part II: with variable resources. *Mgmt Sci.* **21**, 1194–1203
12. L. F. Gelders, J. Maes and L. N. Van Wassenhove (1986) A branch and bound algorithm for the multi item single level capacitated dynamic lot sizing. In *Multi Stage Production Planning and Inventory Control.* (S. Axsater *et al.*, Eds) pp. 92–108. Springer Verlag, Berlin.
13. H. M. Wagner and T. M. Whitin (1958) Dynamic version of the economic lot-sizing model. *Mgmt Sci.* **5**, 89–96.
14. R. Kipp Martin (1987) Generating alternative mixed-integer programming models using variable redefinition. *Opns Res.* **35**, 820–831.
15. G. D. Eppen and R. Kipp Martin (1987) Solving multi-item capacitated lot sizing problems using variable redefinition. *Opns Res.* **35**, 832–848.
16. A. I. Ali, R. Padman and H. Thiagarajan (1989) Dual algorithms for pure network problems. *Opns Res.* **37**, 158–171.
17. D. Bertsekas and P. Tseng (1988) Relaxation methods for minimum cost ordinary and generalized network flow problems. *Opns Res.* **36**, 93–114.
18. M. D. Grigoriadis (1986) An efficient implementation of the network simplex method. *Math. Prog. Study* **26**, 83–111.
19. K. S. Hindi (1993) Efficient implementation of network simplex algorithms. DTG Report 93:3 (update of DTG report 87:7), Department of Computation, UMIST.
20. F. Glover (1989) Tabu search—part 1. *ORSA J. Computing* **1**, 190–206.
21. F. Glover (1990) Tabu search—part 2. *ORSA J. Computing* **2**, 4–32.
22. P. Hansen (1986) The steepest ascent mildest descent heuristic for combinational programming. Presented at the *Congress on Numerical Methods in Combinatorial Programming*, Capri, Italy.
23. F. Glover and M. Laguna (1993) Tabu search. In *Modern Heuristic Techniques in Combinatorial Problems.* (C. R. Reeves, Ed.) pp. 70–150. Blackwell, Oxford.
24. C. R. Reeves (1993) Improving the efficiency of tabu search for machine sequencing problems. *J. Opl. Res. Soc.* **44**, 375–382.
25. M. L. Fisher (1981) The Lagrangean relaxation method for solving integer programming problems. *Mgmt Sci.* **27**, 1–18.
26. A. M. Geoffrion (1974) Lagrangean relaxation of integer programming. *Math. Prog. Study* **2**, 82–114.
27. J. M. Thizy and L. N. Van Wassenhove (1985) Lagrangean relaxation for the multi-item capacitated lotsizing problem: a heuristic approach. *AIIE Trans.* **17**, 308–313.
28. E. H. Bowman (1965) Production scheduling by the transportation method of linear programming. *Opns Res.* **4**, 100–103.
29. A. S. Manne (1958) Programming of economic lot sizes. *Mgmt Sci.* **4**, 115–135.
30. B. P. Dzielinski and R. E. Gomory (1965) Optimal programming of lot sizes, inventory and labor allocations. *Mgmt Sci.* **11**, 874–890.
31. L. S. Lasdon and R. C. Terjung (1971) An efficient algorithm for multi-item scheduling. *Opns Res.* **19**, 946–969.
32. H. C. Bahl (1983) Column generation based heuristic algorithm for multi-item scheduling. *IIE Trans.* **15**, 136–141.
33. H. C. Bahl and L. P. Ritzman (1984) A cyclical scheduling heuristic for lotsizing with capacity constraints. *Int. J. Prod. Res.* **22**, 791–800.
34. G. R. Bitran and H. Matsuo (1991) The multi-item capacitated lot size problem: error bounds of Manne's formulations. *Mgmt Sci.* **32**, 350–358.
35. W. Trigeiro, J. Thomas and J. McClain (1989) Capacitated lot sizing with set-up times. *Mgmt Sci.* **35**, 353–366.

Received July 1994; accepted April 1995 after two revisions