A Local Search Approach to Lot Sequencing and Sizing

Alistair R. Clark,
Faculty of Computer Studies and Mathematics,
University of the West of England, Bristol, BS16 1QY, England.
E-mail: alistair.clark@uwe.ac.uk, fax: +44 (0) 117 344 3155

February 1, 2000

To be presented at the FIP WG5.7 Special Interest Group on
Advanced Techniques in Production Planning and Control,
24-25 February 2000, Florence, Italy.

Abstract

This working paper reports ongoing research into the simultaneous sequencing and sizing of production lots on a set of parallel machines in the presence of sequence-dependent setup times. A flexible mixed integer programming (MIP) model is presented which is optimally solvable only for very small instances. As an alternative, a local search approach is discussed and outlined, making use at each search iteration of an assignment patching algorithm to determine efficient sequences of setups, followed by the dual simplex method to determine optimal lot sizes for that sequence. A solution and varying-sized neighbourhood structure is proposed that will hopefully permit the search to efficiently explore a wide range of solutions and yet eventually home in on a good solution. Computational tests have yet to be carried out.

1 Introduction

The ongoing research reported here is concerned with the simultaneous sequencing and sizing of production lots on a set of parallel machines when a sequence-dependent setup time is incurred to change production between lots of different types. Production is organised in lots in order to meet varying periodic demand under conditions of tight capacity. If lot sequencing and sizing is badly planned and managed, then inventory will be larger than necessary and, worse, setups times will consume scarce machine time, causing excessive unmet demand and backorders.

Many companies face the challenges of managing setup times, particularly in industries where capital investments in production capacity are large and product variety is diverse. Simple approaches such as dispatching rules sometimes function well where the sequencing of fixed-sized orders is concerned, but when the sizing of production lots is also part of the planning decision, we are left to confront a very tough problem indeed.

Little research has been carried out into the problem, perhaps because it is so difficult to solve optimally for anything other than very small sized instances. It is NP-hard, so that it is very unlikely that an optimal solution method exists which is efficient for medium sized problems upwards. Consequently, rather than pursue the fruitless goal of trying to develop a fast optimal procedure, this research explores heuristic approaches that permit the identification of good solutions in a reasonable amount of time. To the industrial user, an optimal solution is an invisible yardstick whose value is not known. Furthermore the often dubious quality of production data and frequent updating of demand forecasts means that a supposedly optimal production plan would in fact be optimal for only one sample point among many millions. A possible approach to overcoming this objection might be a sophisticated stochastic model, but this would have impossible data needs, requiring an knowledge of the probability distributions of the quality of production data and demand forecasts. Instead of following such a hopelessly complex road, a more useful approach would be the development of a lot sizing and sequencing method whose results are generally good, robust and quickly obtainable for industrially sized problems. The purpose of this research is to follow just such a path.

Previous research in this area is rare. A recent survey into lot-sizing [1] noted very few papers that studied the problem of lot sequencing as well as sizing. The general problem that interests us includes representation of setup times that are sequence-dependent and permits multiple setups within a planning period. In addition, it does not require that all or none of the production capacity on a machine be utilised within a period. As such, it is related to the capacitated lot-sizing problem (CLSP) as defined in [1], although the CLSP does not include sequencing decisions.

Clark & Clark [2] developed a mixed integer linear programming (MIP) model that allowed an arbitrary number of setups in each period and showed how it could be simplified when applied under a rolling horizon. The resulting model, presented below, was still only practicable for small to medium sized problems, even when solved merely approximately within a branch-and-bound search. It motivated the application of the more flexible approach presented in this research where the sequencing of lots will be tackled through local search rather than mixed integer programming. Once a sequence is defined, the sizing of lots can be quickly optimised via linear programming.

2 A MIP Model for Lot Sizing and Sequencing

Our aim is to satisfy the demand for P products over the T planning periods with minimal backorders or inventory carried over from one period to the next. The products may be manufactured in batches of varying size on any one of M machines in parallel to eachother. A changeover from one product to another requires a setup time during which the machine cannot process any products. The machines are not identical and may have different rates of production and setup times. There is no restriction on the number of setups in each planning period, but since it makes no sense to produce a product in more than one batch on the same machine in the same period, there will in practice be at most one setup per product per period and so we can limit the number of setups on machine to P per period.

Suppose we permit up to $N \leq P$ setups per period on each machine. Then a MIP formulation of the problem is:

Model MIP:

minimise

$$\sum_{i,t} \left[h_i I_{it}^{+} + g_i I_{it}^{-} \right] \tag{1}$$

such that

$$I_{i,t-1}^{+} - I_{i,t-1}^{-} + \sum_{m,n} x_{imt}^{n} - I_{it}^{+} + I_{it}^{-} = d_{it} \qquad \forall\ i,t \tag{2}$$

$$\sum_{j,n}\left[\sum_i s_{ijm}y^n_{ijmt} + u_{jm}x^n_{jmt}\right] \;\leq\; A_{mt} \qquad \forall\; m,t \tag{3}$$

$$y^1_{ijm1} \;=\; 0 \qquad \forall\; i,m,j\neq i_{0m} \tag{4}$$

$$\sum_j y^1_{i_{0m}jm1} \;=\; 1 \qquad \forall\; m \tag{5}$$

$$\sum_i y^n_{ijmt} \;=\; \sum_k y^{n+1}_{jkmt} \qquad \forall\; j,m,t,n{=}1,\ldots,N{-}1 \tag{6}$$

$$\sum_i y^N_{ijm,t-1} \;=\; \sum_k y^1_{jkmt} \qquad \forall\; j,m,t{=}2,\ldots,T \tag{7}$$

$$x^n_{jmt} \;\leq\; M_{jmt}\sum_i y^n_{ijmt} \qquad \forall\; j,m,n,t \tag{8}$$

$$y^n_{ijmt} \;=\; 0 \text{ or } 1 \qquad \forall\; i,j,m,n,t \tag{9}$$

$$x^n_{imt} \;\geq\; 0 \qquad \forall\; i,m,n,t \tag{10}$$

$$I^+_{it} \;\geq\; 0; \qquad I^-_{it} \;\geq\; 0 \qquad \forall\; i,t \tag{11}$$

where the decision variables are:

$$y^n_{ijmt} = \begin{cases} 1 & \text{if the } n\text{th setup on machine } m \text{ in period } t \text{ is from product} \\ & i \text{ to product } j \\ 0 & \text{otherwise.} \end{cases}$$

x^n_{imt} = Quantity of product i produced between the nth and $n+1$th setups on machine m in period t. (It is non-zero only if the nth setup on machine m is to product i)

I^+_{it} = Stock of product i at the end of period t.

I^-_{it} = backlog of product i at the end of period t.

The parameters and data inputs are:

d_{it} = Demand for product i at the end of period t.

A_{mt} = Available time on machine m in period t.

s_{ijm} = Time needed to setup from product i to product j on machine m.

u_{im} = Time needed to produce one unit of product i on machine m.

h_i = Cost of holding one unit of product i from one period to the next.

g_i = Penalty cost of a carrying over a backorder of one unit of product i from one period to the next.

j_{0m} = The product produced on machine m at the end of period 0, i.e., the starting setup configuration on machine m.

and where

$$M_{imt} \;=\; \min\left\{\frac{A_{mt}}{u_{im}},\; \sum_{t=1}^{T} d_{it} + I_{i0}^{-} - I_{i0}^{+}\right\} \tag{12}$$

is an upper bound on x_{imt}^{n}, since all backlogs and future production of product i might in theory be produced on machine m.

Note that, unlike many formulations in the literature, model MIP allows backlogs. To prohibit them is unrealistic - many companies face occasional or frequent capacity overloads and they often have no immediate choice but to backlog demand. The question of what penalties g_i should be allocated to backlogs can only be answered for each individual context, depending on market conditions and the importance of the customers for particular product. Since the value of such penalties will partly be based on judgement and imprecise information, there is little added value in investing the often huge extra effort needed to solve the model optimally rather than approximately. Only holding and backlog penalty costs are included in the objective function (1) since these are our major concerns. We assume that the total costs of the provision of the production capacity A_{mt} are fixed and do not depend on the production decision variables x_{imt}^{n} and y_{ijmt}^{n}. Additional setup and direct production costs are not included in the objective function as they are likely to vary little or be negligible in comparison to the penalty costs of the additional backlogs provoked by the lost machine time that an inefficient sequence of setups would cause. However, if need be, such costs can be incorporated into the objective function without difficulty by the inclusion of a term such as $\sum_{j,t,n}\left[\sum_i SetupCost_{ijm} y_{ijmt}^{n} + UnitCost_{jm} x_{jmt}^{n}\right]$

Constraints (2) are the standard equations linking inventory, production and demand while constraints (3) represent the limited availability of capacity. Note that since the machine time capacity parameter $A_m t$ is indexed on t, the production periods t in the model may be of different lengths. For example, weekend working may be combined into a single planning period.

Constraints (4) to (7) ensure that a setup on a machine must and may only occur between a single pair of products, possibly both the same product, and that if a certain product is changed to, then it must be changed from in the following setup. The equals sign =, rather than the $\leq$ sign, is necessary in constraints (4) to (7) so that we always know for which product a machine is configured, especially when it is not producing. Thus the combination $y_{jimt}^{n} = 1$ and $x_{imt}^{n} = 0$ must be allowed. Note that constraints (4) to (7) require that for each triple (n, m, t) there is exactly one pair (i, j) for which $y_{jimt}^{n} = 1$, i.e., there must be precisely N setups in each period on each machine, even if a setup $y_{iimt}^{n} = 1$ is just from a product i to itself. Since a setup time s_{iim} from a product i to itself is zero, the model does not force a machine to have exactly N positive-time setups but rather up to N such setups. The remaining zero-time setups are modelling phantoms and do not exist in reality.

Constraints (8) ensure that there must be a setup if a product is produced on a machine in a period, even if it is just a phantom one from a product to itself.

The first setup in a period on a machine must occur at the beginning of the period, but the subsequent N-1 setups may occur at any time within the period. If the constraints $y_{iimt}^{1} = 1\;\forall\, i, m, t$ and $y_{jimt}^{1} = 0\;\forall\, i, m, j \mid i{\neq}j$ are imposed, then model MIP is restricted to just N-1 setups per period, but these may occur at any time within the period. Thus the model is related to (and more general than) the proportional lot-sizing and scheduling problem (PLSP) which allows the single permitted setup in each period to occur at any time within the period, if at all [3, 1]. Model MIP does not, however, permit a setup to begin in one period and finish in the next, so it is not totally flexible.

In [2] it was shown that there are limits to the size of problem that model MIP can solve in practical time. A limit on the time spent searching for a solution can be imposed, an approach which is easy to implement in most MIP solvers, but the computational results suggested that for medium to large problems impractical amounts of time will be spent just identifying a feasible solution.

3 A Local Search Approach to Lot Sizing and Sequencing

An alternative approach is to use a solution method where lot sequencing is solved by local search [4] and lot-sizing by linear programming (LP). This means that a local search solution is uniquely identified by a sequence of a subset of distinct numbers from the set $1, \ldots, P$ for each pair (m, t) of machines and periods. The optimal lot-sizes associated with a given setup sequence $\{y^n_{ijmt} \ \forall\ i, j, m, n, t\}$ obeying constraints (4) to (7) are found by solving the following LP:

Model LotSizes:

minimise

$$\sum_{i,t} \left[h_i I^+_{it} + g_i I^-_{it} \right] \tag{13}$$

such that

$$I^+_{i,t-1} - I^-_{i,t-1} + \sum_m x_{imt} - I^+_{it} + I^-_{it} \ = \ d_{it} \qquad \forall\ i, t \tag{14}$$

$$\sum_i u_{im} x_{imt} \ \leq \ A_{mt} - \sum_{i,j,n} s_{ijm} y^n_{ijmt} \qquad \forall\ m, t \tag{15}$$

where

$$x_{imt} \begin{cases} \geq 0 & \text{if type } i \text{ must be setup on machine } m \text{ in period } t \\ = 0 & \text{otherwise.} \end{cases}$$

Thus, between successive sequences in a local search, the LP will retain the same objective function, but the right hand sides of only some constraints will change and, in addition, the upper bounds of some x_{imt} variables will have to be set to zero or infinity. This means that objective function updating between successive sequences in a local search can be done (quickly we hope) by reoptimising the LP using the dual simplex method.

Furthermore, on each machine there will in practice be at most one setup per product per period and so we can limit the number of setups on a machine to P per period. Thus for a given machine-period pair (m, t) and unordered subset B_{mt} of all P products, we can minimise the value of the total time lost to setups:

$$\sum_{i,j,n} s_{ijm} y^n_{ijmt} \tag{16}$$

by an optimal sequencing of the setup times of the products in the subset. This will in turn give the best possible value of the model LotSizes for the given pair (m, t) and subset B_{mt}. Finding an optimal sequence of products to minimise expression (16) is equivalent to solving an Asymmetric Travelling Salesman Problem (ATSP). This is only viable for very small subsets of products if (16) is to be minimised many times (up to PT) at every local search iteration. However, since we are focusing on finding good (rather than optimal) solutions for model A, a fast ATSP heuristic may be used to minimise (16). The usual symmetric TSP heuristics such as the *k-opt* [5] or the Lin-Kernughan [6] tour improvement methods are not necessarily the best heuristics to use for the asymmetric TSP. Instead, a fast procedure that gives good and often near-optimal results is to solve an assignment problem and then use a patching heuristic to convert assignment subtours into a single salesman tour [7, 8, 9].

If a machine m is already setup for producing type i_0 at the end of period t-1 and must produce type j_{99} after the first setup of period t+1, then the assignment problem to be solved is:

Model AP:

minimise

$$\sum_{i,j \in B_{mt}} s_{ij} z_{ij} \tag{17}$$

such that

$$\sum_{j} z_{i_0 j} = 1 \tag{18}$$

$$\sum_{j} z_{ij} = 1 \qquad \forall\ j \in B_{mt} \tag{19}$$

$$\sum_{i} z_{ij} = 1 \qquad \forall\ i \in B_{mt} \tag{20}$$

$$\sum_{i} z_{ij_{99}} = 1 \tag{21}$$

where the assignment decision variables are:

$$z_{ij} = \begin{cases} 1 & \text{if product } i \text{ is assigned to product } j\text{, i.e., if there is} \\ & \text{a setup from product } i \text{ product } j \\ 0 & \text{otherwise.} \end{cases}$$

Thus at each local search iteration, an assignment patching algorithm is used to determine efficient sequences of setups, followed by the dual simplex method to determine optimal lot sizes for that sequence.

Solving just model AP (without applying the patching heuristic) would be fast and could well provide a good approximate reflection of the optimal value of the ATSP optimal solution that could be used in (15)

Various local search strategies such as Simulated Annealing [10, 11] and Tabu Search [12] have been proposed to encourage a search not to get stuck in a local optimum, but rather to get very near a global optimal solution, and often involve considerable tuning and effort. However, is such precision appropriate in the messy world of production planning and scheduling where the input data is often imprecise and upsets such as rush orders or machine failure are common, necessitating frequent replanning ? A more useful outcome would be a quickly-obtained solution of reasonable quality, especially for medium to large sized problems where optimal seeking methods would take an impractically long time to converge. What implications does this have for the use of local search and the neighbourhood structure ?

At one extreme, a small structure would mean a very slow convergence rate, coupled with the risk of getting trapped in a local optimum. At the other extreme, a very wide ranging neighbourhood structure would lean in the direction of random sampling among all possible solutions. Random sampling has the advantage that it would avoid entrapment in a local optimum, but is possibly inefficient in identifying good solutions by nature of its randomness. A very wide range of neighbours at the start of a local search might also to some extent avoid the worst local optima, and could then be narrowed later in the search to home in to a good local optimum. The challenge here is to find out how to do this efficiently and map the trade-offs between quality of solution and speed of computation.

In order to start the local search with a large neighbourhood and then gradually reduce it, we propose to generate a neighbour by carrying out a large number of random insertions and deletions of products in the sets $B_{mt} \forall\ (b, t)$. As the search progresses, we will slowly reduce the number of such insertions and deletions. Specifically, the procedure is:

1. Identify a starting solution $\{B_{mt} \forall\ (m, t)\}$
2. Let N be a large integer.
3. For $n = 1$ to N do
   (a) Randomly select a pair (m, t).
   (b) Randomly choose whether the next change is an insertion or a deletion.
   (c) If an insertion, then randomly select $b \notin B_{mt}$ and insert b into B_{mt}.
   (d) If a deletion, then randomly select $b \in B_{mt}$ and delete b from B_{mt}.
4. Solve model AP/ATSP and then model LotSizes.
5. Adopt and record the solution if it is the best so far.
6. Reduce N occasionally. Stop if you wish.
7. Go to step 3.

As there are MT sets B_{mt}, each of maximum size P, all solutions are reachable if the search is started with an initial value of $N = PMT$. A fast rate of reduction of N may well not produce as good a final solution as a slower rate, but we cannot make firm statements about this without further experimental investigation.

At the time of writing [February 2000], experimental tests are being designed to test the comparative effectiveness of the following procedures:

1. Random Sampling of a set B_{mt} for each pair (m, t), to serve as a benchmark.
2. Biased Sampling of a set B_{mt} for each pair (m, t).
3. Local Search among all possible solutions $\{B_{mt} \forall\ (m, t)\}$ using the N random insertions/deletions neighbourhood structure described above, with varying intial values and reduction rates of N.
4. Other Local Search procedures among all possible solutions $\{B_{mt} \forall\ (m, t)\}$.

The results will be reported in the future.

References

[1] A. Drexl and A. Kimms, "Lot sizing and scheduling - survey and extensions," *European Journal of Operational Research*, vol. 99, pp. 221–235, 1997.

[2] A. R. Clark and S. J. Clark, "Rolling-horizon lot-sizing when setup times are sequence-dependent," *International Journal of Production Research*, forthcoming.

[3] A. Drexl and K. Haase, "Proportional lotsizing and scheduling," *International Journal of Production Economics*, vol. 40, pp. 73–87, 1995.

[4] E. H. L. Aarts and J. K. Lenstra, eds., *Local Search in Optimization*. New York: John Wiley and Sons, 1997.

[5] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*. Wiley Interscience, 1998.

[6] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Operations Research*, vol. 21, pp. 498–516, 1973.

[7] R. M. Karp, "A patching algorithm for the nonsymmetric traveling-salesman problem," *SIAM Journal on Computing*, vol. 8, pp. 561–573, Nov. 1979.

[8] R. M. Karp and J. M. Steele, "Probabilistic analysis of heuristics," in *The Traveling Salesman Problem - A Guided Tour of Combinatorial Optimization* (E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, eds.), Chichester: Wiley, 1985.

[9] A. M. Frieze and M. E. Dyer, "On a patching algorithm for the random asymmetric travelling saleman problem," *Mathematical Programming*, vol. 46, pp. 361–378, 1990.

[10] R. W. Eglese, "Simulated annealing: a tool for operational research," *European Journal of Operational Research*, vol. 46, pp. 271–281, 1990.

[11] K. A. Dowsland, "Simulated annealing," in *Modern Heuristic Techniques for Combinatorial Problems* (C. R. Reeves, ed.), ch. 2, pp. 20–69, London: Blackwell Scientific, 1993.

[12] F. Glover and M. Laguna, "Tabu search," in *Modern Heuristic Techniques for Combinatorial Problems* (C. R. Reeves, ed.), ch. 3, pp. 70–150, London: Blackwell Scientific, 1993.