Mathematical Programming-based Heuristics for Production Planning

Ross J.W. James
Department of Management
University of Canterbury, New Zealand
ross.james@canterbury.ac.nz

Bernardo Almada-Lobo
Faculdade de Engenharia
da Universidade do Porto, Portugal
almada.lobo@fe.up.pt

Abstract

We develop construction and improvement heuristics to solve the single and parallel-machine capacitated-lotsizing and scheduling problem with sequence dependent setup times and costs. We compare the performance of these heuristics to metaheuristics and other MIP-based heuristics that have been proposed in the literature as well as a state-of-the-art commercial solver. Computational experiments show the effectiveness and efficiency of the proposed approach in solving medium size problems.

Key words: CLSD, Mixed Integer Programming, relax-and-fix, heuristics

1 Introduction

The aim of this paper is to develop a range of mathematical programming-based heuristics to solve complex lotsizing and scheduling problems. The techniques developed here could be easily generalised to other variants of lotsizing and scheduling problems and even other problems outside this domain.

Our motivation, ultimately, is to solve parallel-machine capacitated lotsizing and scheduling problems with sequence dependent setup times and costs (CLSD-PM). This single-stage problem is an extension of the pure capacitated lotsizing problem (CLSP – Bitran and Yanasse (1982)) and of the CLSD proposed by Haase (1996), which only addresses one machine and no setup times, as we also need to consider an allocation dimension with multiple parallel machines and a sequencing dimension with sequence-dependent setups.

CLSD-PM is considered to be a big-bucket problem, as several products/setups may be produced/performed per period. Despite its applicability, there is a lack of research on these problems and correlated variants, mainly due to its inherent complexity (Clark, Almada-Lobo, and Almeder (2010)).

We consider N products to be manufactured on M different capacitated machines over a discrete planning horizon of T periods. Due to the sequence-dependency of setups in a product changeover, lot sizing and sequencing are simultaneously tackled. The objective is to find a strategy that satisfies demands without backlogging and minimizes both setup and holding costs. The computational complexity of the resulting mixed integer programming (MIP) model makes the use of efficient heuristic strategies mandatory for solving large real-world instances. State-of-the-art optimisation engines either fail to generate feasible solutions to this problem or take a prohibitively large amount of computation time, even for the single-machine setting (see Almada-Lobo and James (2010)).

We focus on new mathematical programming-based heuristics that are flexible and could easily be adapted to cope with model extensions or to address different optimisation problems that arise in practice, as they are problem and constraint independent.

2 CLSD-PM Model Formulation

In this paper we study the CLSD-PM of which the single machine problem (CLSD) is a special case. We consider a planning interval with $t = 1, \ldots, T$ periods and $i, j = 1, \ldots, N$ products processed on $m = 1, \ldots, M$ machines. We use the notation $[K]$ to refer to the set $\{1, 2, \ldots, K\}$. As usual, d_{it} denotes the demand of product i in period t, s_{mij} and c_{mij} the time and cost incurred when a setup occurs from product i to product j on machine m respectively, h_i the cost of carrying one unit of inventory of product i from one period to the next, p_{mi} the processing time of one unit of product i on m and C_{mt} the capacity of machine m available in period t. In addition, G_{mit} is an upper bound on the production quantity of product i in period t on machine m. A_{mi} indicates which machines are able to produce which products. A_{mi} is set to one if machine m is able to produce product i, or zero otherwise. The decision variables used are: X_{mit} is the quantity of product i produced in period t on machine m, I_{it} the inventory level of product i at the end of period t and V_{mit} an auxiliary variable that assigns product i on machine m in period t. The larger V_{mit} is, the later the product i is scheduled in period t on machine m, assuring that each machine is only set up for one product at any given time. In addition, the following 0/1 decision variables are defined: T_{mijt} equals one if a setup occurs from product i to product j on machine m in period t, and Y_{mit} equals one if the machine m is set up for product i at the beginning of period t. The CLSD-PM model below is a generalization of the CLSP (single-machine) introduced in Almada-Lobo et al. (2007):

$$\min \sum_m \sum_i \sum_j \sum_t c_{mij} \cdot T_{mijt} + \sum_i \sum_t h_i \cdot I_{it} \tag{1}$$

$$I_{i(t-1)} + \sum_m X_{mit} - d_{it} = I_{it} \qquad \forall i, \forall t \tag{2}$$

$$I_{i0} = 0 \qquad \forall i \tag{3}$$

$$\sum_i p_{mi} \cdot X_{mit} + \sum_i \sum_j s_{mij} \cdot T_{mijt} \leq C_{mt} \qquad \forall m, \forall t \tag{4}$$

$$G_{mit} \cdot (\sum_j T_{mjit} + Y_{mit}) \geq X_{mit} \qquad \forall m, \forall i, \forall t \tag{5}$$

$$Y_{mi(t+1)} + \sum_j T_{mijt} = Y_{mit} + \sum_j T_{mjit} \forall m, \forall i, \forall t \tag{6}$$

$$\sum_i Y_{mit} = 1 \qquad \forall m, \forall t \tag{7}$$

$$V_{mit} + N \cdot T_{mijt} - (N-1) - N \cdot Y_{mit} \leq V_{mjt} \qquad \begin{array}{l} \forall m, \forall i, \forall t \\ \forall j \in [N] \setminus \{i\} \end{array} \tag{8}$$

$$\sum_t X_{mit} \leq G_{mit} A_{mi} \qquad \forall m, \forall i \tag{9}$$

$$(X_{mit}, I_{it}) \geq 0, (T_{mijt}, Y_{mit}) \in \{0,1\}, X_{mit} \in \mathbb{Z}, V_{mit} \in \mathbb{R} \tag{10}$$

Objective function (1) minimizes the sum of setup and inventory holding costs. Constraints (2) balance production and inventories with demand. Constraints (3) set the initial inventory levels. Constraints (4) ensure that production and setups do not exceed the available capacity. The setup forcing constraints are provided by (5). Constraints (6) keep track of the setup carryover information, whilst constraints (7) ensure that each machine is set up for one product at the beginning of each time period. Note that requirements (6) link two consecutive periods. Disconnected subtours are eliminated by constraints (8). These constraints apply whenever one subtour occurs in a period, forcing the respective machine to be set up at the beginning of that period to one of the products that are part of the subtour. Constraints (9) ensure production of a product can only occur on machines that are able to produce that product. Finally, there are the non-negativity and integrality constraints (10).

3 Decomposition Scheme

The techniques we use to solve the CLSD-PM problem are based on decomposing the original problem into subsets that can be solved more easily by a MIP in an iterative fashion. As the number of integer variables in each subproblem are significant smaller than those of the original problem, the solution times to solve each one to optimality is very small. In order to do this we assume we already have a partial or feasible solution, whose decision variables T, X, Y, and I have the values T', X', Y' and I', respectively, obtained in previous iterations. In each iteration, the set of decision variables to be released and the set of those to be frozen (fixed) need to be given. We formulate a subMIP that will solve specific combinations of periods, products and machines, while also bringing into the solution elements of the problem that have been determined at earlier stages. Let β_t denote a parameter that is set to 1 if the period t is to be optimized (i.e. binary variables related to period t are released), $\psi_i = 1$ to represent the products to be optimized and $\delta_m = 1$ to represent machines that are to be optimized. The periods, products and machines that are to remain the same as our incumbent solution have $\beta_t = 0$, $\psi_i = 0$ and $\delta_m = 0$, respectively. We refer to this as the $subMIP\,(T', X', Y', \beta_t, \psi_i, \delta_m)$ model which is formulated as follows:

$$\min \sum_m \sum_i \sum_j \sum_t c_{ij} \cdot T_{ijt} + \sum_i \sum_t h_i \cdot I_{it}$$
$$(2) - (9)$$

$$T_{mijt}(1 - \beta_t \psi_i \psi_j \delta_m) = T'_{mijt}(1 - \beta_t \psi_i \psi_j \delta_m) \forall m, \forall i, \forall j, \forall t \quad (11)$$
$$X_{mit}(1 - \beta_t \psi_i \delta_m) = X'_{mit}(1 - \beta_t \psi_i \delta_m) \quad \forall m, \forall i, \forall t \quad (12)$$
$$Y_{mit}(1 - \beta_{t-1} \psi_i \delta_m) = Y'_{mit}(1 - \beta_{t-1} \psi_i \delta_m) \quad \forall m, \forall i, \forall t \quad (13)$$
$$I_{it}(1 - \beta_{t-1} \psi_i) = I'_{it}(1 - \beta_{t-1} \psi_i) \quad \forall i, \forall t \quad (14)$$

$$(X_{mit}, I_{it}) \geq 0, (T_{mijt}, Y_{mit}) \in \{0, 1\}, X_{mit} \in \mathbb{Z}, V_{mit} \in \mathbb{R}$$

Constraints (11)-(13) define the binary and integer decision variables for periods that have been solved previously, while constraints (14) the inventory which will be naturally integer due to production and demand being integer.

4 MIP-Based Construction Heuristics

4.1 Myopic Fix: Period construction heuristic

The principle of Myopic Fix period construction heuristic is to construct a solution to the problem instance by breaking it down into a given number of period blocks and solving them separately using a MIP solver. The larger the number of periods, the more CPU time it will take to create a solution and also the higher the quality of the solution. The blocks of periods are solved starting from the earliest time period and working through to the latest time period. We assume that the periods where $\beta_t = 1$ are contiguous and that all periods before the first period where $\beta_t = 1$ have been solved. β_t equals one for every t in $[k..min(T, k + w - 1)]$. We set $\psi_i = 1 \forall i$ and $\delta_m = 1 \forall m$.

4.2 Relax-and-Fix

The relax-and-fix (RF) framework decomposes a large-scale MIP into a number of smaller partially relaxed MIP subproblems, that are solved in sequence. Due to the structure of the original MIP, we rely on a time-stage partition, which is a rolling horizon approach. The planning horizon is typically partitioned into non-overlapping intervals. A shift forward strategy solves a sequence of sub-MIPs (one per time interval), each dealing with a sub-set of the integer variable set of the overall problem. The other sub-sets are either relaxed or removed depending on the simplification strategy. As this heuristic progresses, the integer variables (and, depending on the freezing strategy, also the continuous variables) are permanently fixed to their current values. The schedule is completed at the last iteration.

The RF can be parameterized in many ways. Our design is based on Merce and Fontan (2003) and Absi and Kedad-Sidhoum (2007). Let γ be the number of overlapping periods and w the width of each interval. It is implied here that both parameters are constant throughout the algorithm. Note that γ allows us to smooth the heuristic solution by creating some overlap between successive planning

intervals (Pochet and Wolsey (2006)). In addition, t_k^i and t_k^f denote the initial and final periods of the interval at step k. At each iteration, k, of the heuristic, the sub-set of variables T and Y up to period t_{k-1}^i are fixed, the sub-set from period t_k^i up to t_k^f are restricted to be integer, and other sub-sets being relaxed. Note that $w - \gamma$ defines the length of the anticipation horizon at each iteration and, consequently, the number of iterations of the heuristic. Recall that our formulation keeps track of the setup carryover information, knowing the product that the machine is ready to process at the beginning of each time period. The motivation for using $w > 1$ and $\gamma > 0$ (with $\gamma < w$) comes from the setup carryover information. With a $\gamma = 0$, blocks do not overlap and therefore the product the machine is set up for at the beginning of the block is fixed at the previous iteration. This therefore affects the production sequence of the current block to be optimised.

5 MIP-Based Improvement Heuristic

The principle behind the improvement heuristic is to reoptimise iteratively a contiguous set of periods and reoptimise the jobs in these periods, fixing all the variables of other periods. The reoptimisation is again done via a subMIP similar to that outine previously except that constraints (14) is removed. Each subproblem is derived from the best solution found so far, fixing the binary variables of the periods where $\beta_t = 0$ at the optimal values from the incumbent solution, thus dealing with a reduced number of "free binary" variables (related to periods where $\beta_t = 1$).

We do not relax the integrality restriction for any binary variable. In order to improve the speed of the heuristic, the improvement heuristic is designed to intelligently select the periods. Initially the probability of selecting the set of periods starting at any given periods will be the same. However as periods are selected, the probability of that period being selected is reduced. Two factors determine a period's probability of selection, the frequency of the period being selected and the recency of the period being selected. The more frequently a period has been used (given by the number of runs the respective period was selected), and the more recently it has been used (given by the number of iterations passed since the last selection of that period), the lower the probability of selection. The probability of selection of a period is proportional to a weighing average of these two factors. The search also keeps track of which period combinations have been solved without any improvement in the solution quality. A period will not be selected if it already has been used and the current objective value was obtained (providing a form of short-memory which speeds up the solution procedure). If all period combinations have been tried and there has been no improvement in the solution, then we are in a local minima and the heuristic will terminate. The heuristic will also terminate if the maximum CPU time is exceeded. We refer to the heuristic using the above formulation as period heuristic (PH).

One of the main causes of local minima in the above formulation is that the quantity produced in any one period is only optimised to the periods under consideration. The formulation can quite easily be generalized by removing constraints (12) to allow for the quantities produced in each period (X_{mit}), and subsequently the inventory held in each period (I_{it}), to be optimised for all periods, while the sequence is optimised only for selected periods, hence overcoming this source of local minima. The disadvantage of this approach will be that the solution time to solve

an iteration will increase. Nevertheless, it is well known that given a fixed set of the setup variables, the remaining linear subproblem can be solved optimally through a network flow reformulation that can be solved in polynomial time, thus each node of the branch-and-cut tree can be evaluated quickly. The cost of this extra computational time compared to the extra quality achieved will be assessed in Section 6. We refer to the heuristic using this formulation as X (production quantity) and P (period) heuristic (XPH). The pseudo code for PH and XPH is identical, except for the MIP model used.

6 Computational Experiments

6.1 Single Machine Experiments

For the single-machine problem we test different algorithms on random data sets generated using the approach of Almada-Lobo et al. (2007). Elements of the problem were generated from a uniform distribution and then rounded to the nearest integer, or calculated from elements that were generated this way. The ranges used for the elements are: Setup Times between 5 and 10 time units; Setup Costs are proportional to the setup time by a specified parameter (Cost of Setup per unit of time, θ); Holding costs between 2 and 9 penalty units per period; Demand between 40 and 59 per period; Period Capacity is proportional to the total demand in that period as defined by a parameter (Capacity Utilization per period, Cut): $C_t = \sum_i d_{it}/Cut$; Processing time for one unit = one unit of time.

Twenty four different problems types were created from the combinations of the following problem parameters: Number of Products $N \in 15, 25$; Number of Periods $T \in 5, 10, 15$; Capacity Utilization per period $Cut \in 0.6, 0.8$; Cost of Setup per unit of time $\theta \in 50, 100$

In each case 10 different instances were generated, creating a total of 240 problem instances to be solved. Each type of instance can be characterized by the quadruple N, T, Cut and θ.

The number of periods to solve at a time to solve for the Myopic-Fix MIP construction heuristic was tested with 2 and 3 period solutions. The Relax-and-Fix heuristic requires two parameters, namely the width of the interval w and the number of overlapping periods γ. Preliminary empirical analysis pointed out the following choice of values: $w = 2$ and $\gamma = 1$. For each subMIP$_k$ $\left(t_k^i, t_k^f\right)$, we have set a time limit for each of $3600/K$ seconds, where $K = \lceil (T - w)/(w - \gamma) \rceil + 1$ denotes the total number of iterations, and a relative MIP gap tolerance of 0.5%. Besides these two MIP approaches, we implement the constructive heuristic of Almada-Lobo et al. (2007) to generate an initial solution. It contains several forward and backward steps that place feasibility over optimality to find feasible solutions efficiently. The aim is to compare afterwards the effectiveness of the improvement heuristic given different initial solutions.

To evaluate the quality of these heuristics, we use the lower bound generation procedure described in Almada-Lobo et al. (2007), which essentially strengthens the formulation with the (l, S) cuts that are introduced with the separation algorithm of Barany, Vanroy, and Wolsey (1984). These computational experiments were performed on a computer with a Pentium T7700 CPU running at 2.4 GHz with 2GB of random access memory. On this system, Parallel CPLEX 11.1 was used as the

mixed integer programming solver, while the algorithms were coded in Visual C++ .NET 2005.

The problems solved here are a subset of the problems that were solved in Almada-Lobo and James (2010) using Tabu Search and Variable Neighbourhood Search. Almada-Lobo and James (2010) used similar hardware to what we used here, and therefore we can compare the quality of the solutions between the different techniques and the CPU times required to obtain these solutions.

Tables 1 and 2 present the computational results of the different algorithms proposed in this paper. The algorithms presented in these tables are as follows:

- PHOC/XPHOC: five-step construction heuristic of Almada-Lobo et al. (2007) then a two period improvement heuristic or two period and quantity improvement heuristic respectively;
- PHRF/XPHRF: relax-and-fix construction heuristic then a two period improvement heuristic or a two period and quantity improvement heuristic respectively;
- PH2/XPH2: two period construction heuristic then a two period improvement heuristic or a two period and quantity improvement heuristic respectively;
- PH3/XPH3: three period construction heuristic then a two period improvement heuristic or a two period and quantity improvement heuristic respectively;
- TS/VNS : the Tabu search or Variable Neighbourhood Search of Almada-Lobo and James (2010) respectively;
- FOHRF: relax-and-fix construction heuristic then a fix-and-optimise improvement heuristic with a MIP solution tolerance of 0.005;
- CPLEX: branch-and-cut performed by Parallel CPLEX 11.1 on the original model;

Of the two different types of LP heuristics, not surprisingly XPHOC, XPHRF, XPH2 and XPH3 dominate PHOC, PHRF, PH2 and PH3 respectively. XPHRF is clearly the best heuristic in terms of solution quality, being on average 1.40% off the lower bound, compared to 1.50%, 1.66%, 3.55%, 3.59% for XPH2, XPH3, PH2 and PH3 respectively. More surprisingly are the solution times for XPH2, which average 88.4 seconds for the test data. This was considerably faster than XPH3 and PH3, at 151.1 and 107.1 seconds respectively, but slower than PH2 at 57.8 seconds, as expected.

It appears that regardless of the initial solution, the extra quality of the final solutions obtained from the XPH heuristic are worth the extra computational time involved. Furthermore, the results show that the XPH improvement heuristic is not too-dependent upon the starting solution, as proven by the small differences on the gaps reported by XPHOC, XPHRF, XPH2 and XPH3.

The results clearly show that all LP-based heuristics outperformed the TS and VNS metaheuristics in almost every problem type. The only exception being where there are fifteen product types, five periods, $Cut = 0.6$ and $\theta = 100$, where VNS outperforms the PH2 and PH3 heuristics at the expense of significantly larger computational time. In this case, XPHRF, XPH2 and XPH3 performed better than VNS. In all other cases all the LP heuristics not only obtain higher quality solutions but also do so in less time. Given the same initial solution, it is clear that XPH (XPHOC) produces significantly better results with shorter CPU times than VNS

or TS, being 1.59% off the lower bound (needing only 126.4 seconds), compared to 4.75% (2310.3 seconds) and 5.98% (727.7 seconds) for the other two, respectively. There are no single test instances where FOHRF dominates XPHRF. If we look at Table 1 we see that there are no problem classes where FOHRF has a better average than XPHRF. In terms of solution quality CPLEX, not surprisingly, provided the best solution when it could find them. Unfortunately CPLEX could only find solutions to the smaller problem instances and could not find even a feasible solution to the larger problems within the one hour time limit.

Table 1: Average % Deviation from the Lower Bound

Problem Type	Average % Deviation from the Lower Bound											
N-T-Cut-θ	PHOC	XPHOC	PH2	XPH2	PH3	XPH3	PHRF	XPHRF	TS	VNS	FOHRF	CPLEX
15-5-0.6-50	1.13%	0.57%	1.43%	0.35%	0.64%	0.44%	0.47%	0.46%	2.48%	1.05%	0.53%	**0.22%**
15-5-0.6-100	4.57%	2.39%	7.25%	1.97%	5.28%	2.80%	1.86%	1.82%	6.40%	3.85%	1.93%	**1.37%**
15-5-0.8-50	1.31%	0.73%	1.47%	0.66%	1.04%	0.76%	0.72%	0.64%	3.11%	1.42%	0.83%	**0.40%**
15-5-0.8-100	4.27%	2.33%	4.59%	2.45%	3.75%	2.50%	2.31%	2.08%	6.56%	3.83%	2.36%	**1.73%**
15-10-0.6-50	1.92%	0.67%	0.56%	0.46%	1.92%	0.76%	0.86%	0.64%	3.27%	2.33%	1.11%	**0.40%**
15-10-0.6-100	6.09%	3.24%	5.65%	3.47%	9.01%	3.65%	4.24%	2.95%	7.52%	5.85%	4.16%	bf2.70%
15-10-0.8-50	1.71%	0.84%	1.24%	0.92%	2.07%	0.90%	1.11%	0.89%	3.75%	2.39%	1.59%	bf0.60%
15-10-0.8-100	5.93%	3.42%	6.40%	3.44%	6.36%	3.31%	3.52%	**2.83%**	9.03%	6.53%	3.53%	3.40%
15-15-0.6-50	1.85%	0.77%	1.20%	0.73%	1.53%	1.02%	0.90%	**0.58%**	3.82%	3.26%	1.46%	0.64%
15-15-0.6-100	6.78%	3.69%	7.59%	**3.52%**	7.11%	3.76%	5.45%	3.54%	9.05%	7.78%	5.04%	-
15-15-0.8-50	1.65%	0.90%	1.61%	0.94%	1.85%	1.01%	1.09%	**0.84%**	4.10%	3.34%	1.81%	-
15-15-0.8-100	6.64%	3.80%	6.94%	3.60%	6.27%	3.73%	4.68%	**3.46%**	10.17%	8.16%	5.12%	-
25-5-0.6-50	1.43%	0.28%	1.59%	0.22%	0.55%	0.27%	0.24%	0.17%	3.38%	2.24%	0.43%	**0.09%**
25-5-0.6-100	4.78%	1.23%	6.97%	1.31%	4.50%	1.43%	0.94%	0.85%	5.82%	4.66%	1.03%	**0.66%**
25-5-0.8-50	1.24%	0.52%	1.65%	0.54%	0.67%	0.52%	0.47%	0.43%	3.81%	2.51%	0.68%	**0.27%**
25-5-0.8-100	3.06%	1.16%	3.23%	1.08%	2.46%	1.11%	1.08%	0.95%	6.56%	4.02%	1.32%	**0.85%**
25-10-0.6-50	1.89%	0.65%	0.52%	**0.38%**	2.01%	0.73%	0.64%	0.50%	4.22%	4.28%	1.13%	-
25-10-0.6-100	5.53%	2.23%	4.63%	**1.90%**	8.46%	2.04%	3.00%	2.03%	7.47%	7.10%	3.34%	-
25-10-0.8-50	1.56%	0.62%	1.34%	**0.49%**	1.67%	0.68%	0.66%	0.53%	4.74%	4.15%	1.15%	-
25-10-0.8-100	4.69%	2.06%	4.99%	1.93%	4.82%	2.10%	2.58%	**1.78%**	8.82%	6.72%	2.93%	-
25-15-0.6-50	2.09%	0.60%	1.18%	0.55%	1.50%	0.95%	0.81%	**0.54%**	4.74%	4.88%	1.59%	-
25-15-0.6-100	5.98%	2.46%	6.38%	2.19%	6.46%	2.50%	3.68%	**2.17%**	9.01%	9.62%	4.25%	-
25-15-0.8-50	1.69%	0.64%	1.54%	0.64%	1.53%	0.67%	0.94%	**0.62%**	5.28%	5.11%	1.56%	-
25-15-0.8-100	5.18%	2.38%	5.33%	2.26%	4.70%	2.30%	3.18%	**2.22%**	10.48%	8.96%	3.68%	-
Average	3.46%	1.59%	3.55%	1.50%	3.59%	1.66%	1.89%	1.40%	5.98%	4.75%	2.19%	-

Note: A '-' indicates no solution was found in the one hour time limit

6.2 Multiple Machines

The data for multi-machine problems are generated in a similar manner to the problems for single machines, however two extra parameters are required to generate the assignment matrix data A_{mi}, that is a 0/1 parameter that allows the allocation of products to machines. These parameters are the probability of the extra machines being able to produce the same product, $MProb$, and the maximum percentage difference in the number of jobs between machines, $MBal$. When $MProb$ equals to one, each machine can process every product.

As a base model we used a mid-sized problem with 15 products, 10 periods and 80% capacity utilization. We then tried different perturbations of M, N, T, Cut, θ, $MProb$ and $MBal$. For each combination 10 instances were generated. The average results are presented in Table 3. Note that all problems have a 3600 second time limit. The Fixed and Optimise Heuristics have a $3600/K$ second sub-MIP time limit, while the INSRF heuristic has a 600 second XPH sub-MIP time limit.

Table 2: Average Computational Times

Problem Type	Time in Seconds											
N-T-*Cut-θ*	PHOC	XPHOC	PH2	XPH2	PH3	XPH3	PHRF	XPHRF	TS	VNS	FOHRF	CPLEX
15-5-0.6-50	2.7	2.9	1.3	3.0	3.1	4.1	6.7	5.3	39.0	600.2	5.0	23.1
15-5-0.6-100	4.9	8.7	2.9	6.9	5.8	12.1	19.6	14.9	30.7	493.3	14.8	56.9
15-5-0.8-50	6.6	4.8	3.8	3.9	6.2	7.3	10.0	8.3	13.3	716.5	7.5	20.3
15-5-0.8-100	8.2	22.4	8.0	10.8	36.9	45.4	36.5	28.7	9.7	589.2	27.3	851.8
15-10-0.6-50	8.0	10.7	3.3	4.5	9.9	20.7	20.0	18.5	767.8	2755.3	14.9	2539.2
15-10-0.6-100	28.4	19.2	9.5	21.4	13.4	37.5	54.2	52.1	84.0	2730.7	42.9	3600.0
15-10-0.8-50	15.0	16.0	10.1	14.5	14.4	29.2	28.6	24.8	398.2	2694.7	20.3	2444.7
15-10-0.8-100	27.6	33.0	25.8	37.0	81.3	119.0	122.1	94.2	53.4	2736.7	87.6	3600.0
15-15-0.6-50	16.4	19.5	5.4	16.0	15.2	19.0	35.6	34.3	1344.6	2822.1	25.4	3275.9
15-15-0.6-100	16.8	31.7	17.2	35.5	22.3	48.7	105.7	102.1	598.9	2810.0	86.1	3600.5
15-15-0.8-50	24.2	30.5	17.8	24.9	30.7	48.9	48.9	44.8	428.4	2664.0	35.5	3300.7
15-15-0.8-100	47.1	57.7	44.4	64.8	148.9	208.7	222.9	189.0	239.5	2791.8	170.9	3600.1
25-5-0.6-50	29.9	36.2	7.7	20.9	15.7	24.7	41.9	34.8	566.7	1894.4	30.2	94.0
25-5-0.6-100	55.2	79.2	18.9	43.9	37.1	91.0	118.8	95.7	127.6	1646.8	79.4	1737.9
25-5-0.8-50	36.5	54.8	27.0	38.7	47.7	68.3	102.4	79.3	101.3	1996.7	75.0	224.6
25-5-0.8-100	108.5	176.3	90.6	208.0	130.8	229.3	280.4	283.5	193.2	1755.7	209.3	3369.6
25-10-0.6-50	34.3	103.0	16.5	41.1	45.1	72.2	153.0	137.3	1887.2	3099.7	101.3	3600.1
25-10-0.6-100	74.3	242.4	51.4	113.9	58.2	169.6	275.6	274.1	789.2	2908.1	210.8	3600.9
25-10-0.8-50	148.7	176.5	95.6	126.5	131.6	197.7	225.3	195.0	1377.1	3086.9	152.0	3602.1
25-10-0.8-100	608.7	475.1	233.2	260.9	453.0	668.0	496.6	455.6	895.8	2943.1	358.7	3600.0
25-15-0.6-50	59.7	161.0	33.4	109.8	76.6	124.8	385.4	347.0	2785.8	3053.0	192.1	3600.8
25-15-0.6-100	125.0	337.6	92.1	240.6	110.6	273.8	734.6	822.9	1188.3	2870.3	480.6	3601.1
25-15-0.8-50	179.8	336.8	185.1	218.5	269.4	284.7	407.4	359.6	2381.5	2822.9	261.4	3607.4
25-15-0.8-100	518.8	597.3	385.6	456.6	806.8	821.8	891.2	967.4	1163.3	2964.0	712.7	3600.3
Average	91.1	126.4	57.8	88.4	107.1	151.1	201.0	194.5	727.7	2310.3	141.7	2548.0

CPLEX could not find solutions to these problems within the one hour time limit, however it was used to calculate a lower bound that was based on a plant-location reformulation of our model which is known to produce tighter bounds.

From Table 3 we can see that XPHRF on average is marginally better than the FOHRF. Adding an extra machine clearly makes the problem much more difficult to solve as can be seen by the increasing average deviations between those problems with 2 machines and those with 3 machines. As the sub-MIPs for these techniques have been time limited, the difficulty of the problem will be reflected in a loss of quality rather than an increase in computational time. In several cases we can see this occurring with the problems with 3 machines actually taking less computational times than those with 2 machines. So although we can solve larger problems with this technique, the importance of providing an appropriate amount of time for both the overall search and the sub-MIPs is clearly a vital aspect which dictates the performance of these heuristics. The larger problems, in general, will require much more sub-MIP and overall computational time to obtain similar levels of performance to problems with fewer machines. The parameter $MProb$ (related to the flexibility of the machinery) has an impact on the performance of the algorithms. As $MProb$ increases, the problem gets more difficult as many alternative optimal solutions can be obtained. $MBal$ seems to have no impact on the behaviour of the solution procedures.

7 Conclusions

In this paper we have presented heuristics based on MIP to solve the CLSD and CLSD-PM. The first two heuristics are construction heuristics which solve a MIP optimally scheduling a specified number of periods at a time. The third is a stochastic improvement heuristic which reoptimises the sequence and production quantities

Table 3: Multi-Machine Results - Summary

Problem Type	Ave. % Dev. from Lower Bound		Ave. Time in Seconds	
M-N-T-Cut-θ-$MProb$-$MBal$	XPHRF	FOHRF	XPHRF	FOHRF
2-15-5-0.8-50-80-20	1.49%	2.05%	323.7	50.6
2-15-10-0.8-50-80-20	2.60%	3.60%	527.1	110.3
2-15-10-0.8-100-80-20	7.10%	8.31%	1371.2	345.0
2-20-10-0.8-100-80-20	5.90%	6.17%	3756.1	994.8
2-15-10-0.8-100-60-20	6.02%	7.19%	322.2	139.5
2-15-10-0.8-100-80-10	6.98%	8.46%	1254.5	370.9
3-15-10-0.8-50-80-20	10.78%	7.30%	7623.0	682.8
3-15-5-0.8-50-80-20	4.37%	4.89%	3023.8	173.8
3-15-10-0.8-100-60-20	12.86%	14.73%	2910.9	711.2
3-15-10-0.6-100-60-20	8.00%	10.86%	960.3	186.5
Average	6.61%	7.36%	2207.3	376.5

for given periods, and the fourth is a randomized improvement heuristic that reoptimises the production quantities for all periods, while reoptimising the sequence for only a given set of periods. The computational experiments found that the improvement heuristics proposed here, preceded by the construction heuristics, outperform other meta-heuristic approaches proposed in the literature in terms of quality and often in time, as well as other MIP-based heuristics proposed in the literature. These heuristics also clearly outperformed a commercial MIP solver for the larger instances.

References

Absi, N., and S. Kedad-Sidhoum. 2007. "MIP-based heuristics for multi-item capacitated lot-sizing problem with setup times and shortage costs." *Rairo-Operations Research* 41 (2): 171–192.

Almada-Lobo, B., and R.J.W. James. 2010. "Neighbourhood Search Metaheuristics for Capacitated Lotsizing with Sequence-dependent Setups." *International Journal of Production Research* 48 (3): 861–878.

Almada-Lobo, B., D. Klabjan, M. A. Carravilla, and J. F. Oliveira. 2007. "Single Machine Multi-product Capacitated Lotsizing with Sequence-dependent setups." *International Journal of Production Research* 45 (20): 4873–4894.

Barany, I., T. J. Vanroy, and L. A. Wolsey. 1984. "Strong Formulations for Multi-Item Capacitated Lot Sizing." *Management Science* 30 (10): 1255–1261.

Bitran, G.R., and H.H. Yanasse. 1982. "Computational complexity of the capacitated lot size problem." *Management Science* 28 (10): 1174–1186.

Clark, A. R., B. Almada-Lobo, and C. Almeder. 2010. "Editorial on lotsizing and scheduling: industrial extensions and research opportunities." *Accepted for publication in International Journal of Production Research.*

Haase, K. 1996. "Capacitated lot-sizing with sequence dependent setup costs." *Operations Research Spektrum* 18:51–59.

Merce, C., and G. Fontan. 2003. "MIP-based heuristics for capacitated lotsizing problems." *International Journal of Production Economics* 85 (1): 97–111.

Pochet, Y., and L. A. Wolsey. 2006. *Production Planning by Mixed Integer Programming.* Springer Series in Operations Research and Financial Engineering. New York: Springer.