Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 175 (2006) 1070–1083
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

A Lagrangian-based heuristic for the capacitated lot-sizing problem in parallel machines

Franklina Maria Bragion Toledo [a], Vinícius Amaral Armentano [b,*]

[a] *Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Caixa Postal 668, São Carlos–SP, CEP 13560-970, Brazil*
[b] *Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Caixa Postal 6101, Campinas–SP, CEP 13083-970, Brazil*

Received 13 April 1999; accepted 17 June 2005
Available online 31 August 2005

Abstract

This paper addresses the capacitated lot-sizing problem involving the production of multiple items on unrelated parallel machines. A production plan should be determined in order to meet the forecast demand for the items, without exceeding the capacity of the machines and minimize the sum of production, setup and inventory costs. A heuristic based on the Lagrangian relaxation of the capacity constraints and subgradient optimization is proposed. Initially, the heuristic is tested on instances of the single machine problem and results are compared with heuristics from the literature. For parallel machines and small problems the heuristic performance is tested against optimal solutions, and for larger problems it is compared with the lower bound provided by the Lagrangian relaxation.

Keywords: Heuristics; Lot-sizing; Parallel machines; Lagrangian relaxation

1. Introduction

The lot-sizing problem over periods of a finite horizon consists of determining how much to produce of each item in order to satisfy the demand in each period without violating capacity constraints. The resulting production plan should minimize the sum of setup, production and inventory costs. This problem is known in the literature as the capacitated lot-sizing problem (CLSP), which can model single stage and multistage production structures. There has been a vast amount of research in this area and surveys can be found in

* Corresponding author. Tel.: +55 19 788 7914; fax: +55 19 239 1395/3289 1395.
E-mail addresses: fran@icmc.usp.br (F.M.B. Toledo), vinicius@densis.fee.unicamp.br (V.A. Armentano).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2005.06.029

Bahl et al. (1987) and Kuik et al. (1994). Drexl and Kimms (1997) present and discuss several lot-sizing and scheduling models other than CLSP. Recently, Karimi et al. (2003) presented a review of models and algorithms for the single level capacitated lot-sizing problems.

Belvaux and Wolsey (2001) discuss models for lot-sizing problems and present valid inequalities to obtain tight mixed-integer programs, which can be solved by commercial systems such as XPRESS and CPLEX. Wolsey (2002) suggests a classification for lot-sizing problems and indicates for each problem class families of valid inequalities, existence of tight reformulations and complexity of the optimization algorithms.

This paper deals with the single stage CLSP in the setting of unrelated parallel machines. Each item can be produced on any machine and a setup time is incurred before starting production. It is well known that the presence of setup times poses hard difficulties (Trigeiro, 1987) in devising efficient optimal algorithms and heuristics. It has been shown by Maes et al. (1991) that the problem of finding a feasible solution in this case is NP-complete.

Focusing on previous works that deal with the single stage, single machine CLSP with setup times it can be noted that few solution methods have been proposed. For the single stage problem, optimal algorithms based on the branch-and-bound method have been developed by Diaby et al. (1992a) and Armentano et al. (1999). The optimal cross decomposition method by Van Roy (1983) was used by Souza and Armentano (1994) to solve the problem with bounds on the production variables. Heuristic methods have been proposed to overcome the limitations of optimal methods. Trigeiro et al. (1989) suggest a heuristic based on the Lagrangian relaxation of the capacity constraints and subgradient optimization. At each step of the subgradient method a procedure, which shifts production between periods, is used to obtain a feasible solution. A similar approach was proposed by Lozano et al. (1991) and Hindi et al. (2003). Diaby et al. (1992b) also suggest a heuristic based on the Lagrangian relaxation of the capacity constraints. At the end of the subgradient optimization the setup decisions are retained and transportation problems are solved in order to obtain feasible production quantities. Recently, Jans and Degraeve (2004) developed improved lower bounds. Extensions of this problem have been considered in the literature. Özdamar et al. (2002) include overtime, Hung and Hu (1998, 2003) consider multiple capacitated resources and backorder. Gopalakrishnan et al. (2001) propose a tabu search algorithm for the CLSP with setup carryover, and also apply the algorithm for solving the CLSP without carryover.

The research reported in the literature on lot sizing in parallel machines is quite limited. Sung (1986) suggests a dynamic programming algorithm to solve a single item, parallel facility production planning model without capacity constraints and no backlogging allowed. Lasdon and Terjung (1971) propose a heuristic for the discrete lot-sizing and scheduling problem (DLSP) for identical parallel machines without setup time. In the DLSP only one item can be produced per machine and per period and, if so, production uses the full capacity. Carreno (1990) proposes a heuristic for the economic lot scheduling problem (ELSP) for identical parallel machines with setup times. The ELSP is a continuous time and infinite horizon model with constant demand rate for each item. To the best of our knowledge the CLSP for unrelated parallel machines, denoted CLSPP, has not been addressed in the literature.

This paper proposes a heuristic based on the Lagrangian relaxation of the capacity constraints and subgradient optimization for solving the CLSPP. At every step of the subgradient method feasibility procedure and an improvement procedure are applied to the solution of the Lagrangian problem. Both procedures are based on shifts of production amounts between periods and machines in an attempt to construct a capacity feasible solution and improve it. Initially, the heuristic is tested on instances generated by Trigeiro et al. (1989) for the single machine problem and its performance is compared with that of Trigeiro et al. (1989) and Gopalakrishnan et al. (2001). For parallel machines and small problems the heuristic solutions are compared with optimal solutions and for larger problems involving up to 50 items, 18 periods and 6 machines the heuristic performance is tested against the lower bound provided by the Lagrangian relaxation.

2. Problem formulation

In order to state the problem mathematically, let

s_{ij} setup cost of item i on machine j;
c_{ij} unit production cost of item i on machine j;
h_i unit inventory cost of item i;
d_{it} demand of item i in period t;
b_{ij} time to produce one unit of item i on machine j;
f_{ij} setup time of item i on machine j;
C_j capacity of machine j (in units of time);
x_{ijt} amount of item i produced on machine j in period t (a decision variable);
y_{ijt} a binary variable which assumes value 1 if item i is produced on machine j in period t and 0, otherwise (a decision variable);
I_{it} inventory of item i at the end of period t (a decision variable);
M an upper bound on x_{ijt}.

The CLSPP can be formulated as the following mixed integer programming model.

$$\text{Minimize} \quad \sum_{t=1}^{T}\sum_{j=1}^{m}\sum_{i=1}^{n}(s_{ij}y_{ijt}+c_{ij}x_{ijt})+\sum_{t=1}^{T}\sum_{i=1}^{n}h_iI_{it}$$

$$\text{subject to} \quad \sum_{j=1}^{m}x_{ijt}+I_{i,t-1}-I_{it}=d_{it} \quad i=1,\ldots,n; \quad t=1,\ldots,T, \tag{1}$$

$$\sum_{i=1}^{n}(b_{ij}x_{ijt}+f_{ij}y_{ijt})\leqslant C_j \quad j=1,\ldots,m; \quad t=1,\ldots,T, \tag{2}$$

$$x_{ijt}\leqslant My_{ijt} \quad i=1,\ldots,n; \quad j=1,\ldots,m; \quad t=1,\ldots,T, \tag{3}$$

$$y_{ijt}\in\{0,1\}, x_{ijt}\geqslant 0, I_{it}\geqslant 0 \quad i=1,\ldots,n; \quad j=1,\ldots,m; \quad t=1,\ldots,T. \tag{4}$$

The objective function expresses the sum of setup, production and inventory costs. Constraints (1) represent the inventory balance equations. Constraints (2) indicate that the amount of capacity used for production is limited. Constraints (3) ensure the incidence of a setup cost and a setup time when x_{ijt} is positive. Backlogging is not allowed as indicated by the nonnegative restriction on I_{it} in (4).

3. Lagrangian relaxation

Consider the Lagrangian relaxation with respect to the capacity constraints (2) and let $\mu_{jt}\geqslant 0$, $j=1,\ldots,m$; $t=1,\ldots,T$ denote the associated Lagrangian multipliers. The resulting Lagrangian problem is given by

$$g(\mu_{jt})=\underset{x_{ijt},I_{it},y_{ijt}}{\text{Minimize}} \sum_{t=1}^{T}\sum_{j=1}^{m}\sum_{i=1}^{n}(s_{ij}y_{ijt}+c_{ij}x_{ijt})+\sum_{t=1}^{T}\sum_{i=1}^{n}h_iI_{it} + \sum_{t=1}^{T}\sum_{j=1}^{m}\mu_{jt}\left[\sum_{i=1}^{n}(b_{ij}x_{ijt}+f_{ij}y_{ijt})-C_j\right]$$

subject to (1), (3), and (4).

It is easy to see that $g(\mu_{jt})$ decomposes into the sum of the constant $-\sum_{t=1}^{T}\sum_{j=1}^{m}\mu_{jt}C_j$ and n independent subproblems, one for each item, with the form

$$g_i(\mu_{jt}) = \underset{x_{ijt},y_{ijt},I_{it}}{\text{Minimize}} \sum_{t=1}^{T}\sum_{j=1}^{m}[(s_{ij}+\mu_{jt}f_{ij})y_{ijt}+(c_{ij}+\mu_{jt}b_{ij})x_{ijt}]+\sum_{t=1}^{T}h_iI_{it}$$

$$\text{subject to } \sum_{j=1}^{m}x_{ijt}+I_{i,t-1}-I_{it}=d_{it}, \quad t=1,\ldots,T;$$

$$x_{ijt}\leqslant My_{ijt} \quad j=1,\ldots,m;\ \ t=1,\ldots,T;$$

$$y_{ijt}\in\{0,1\}, \quad x_{ijt}\geqslant 0, I_{it}\geqslant 0 \quad j=1,\ldots,m;\ \ t=1,\ldots,T.$$

Note that $g_i(\mu_{jt})$ is a minimization problem with a concave objective function and linear constraints. It follows that the minimum occurs in an extreme point, which in the case of a polyhedral set has at most the number of constraints (T) nonzero variables. If $d_t > 0$, $t = 1,\ldots,T$, then either $I_{t-1} > 0$ or $x_{ijt} > 0$ for some $j \in \{1,\ldots,m\}$. Thus, every extreme point must satisfy the following property:

$$I_{t-1}x_{ijt}=0, \quad j=1,\ldots,m;\ \ t=1,\ldots,T.$$

The characterization of an extreme point above is a generalization of the single machine case which was stated by Wagner and Whitin (1958). It is also valid in the case where $d_{it} = 0$ for some t and the reasoning follows the same line presented by Johnson and Montegomery (1974). Based on this characterization, we extended the single machine dynamic programming algorithm proposed by Evans (1985) and used it to solve the Lagrangian problem.

The subgradient method was used to maximize the dual function $g(\mu_{jt})$ over $\mu_{jt} \geqslant 0$, $j = 1,\ldots,m, t = 1,\ldots,T$. The algorithm proposed by Held et al. (1974) and the modified algorithms suggested by Camerini et al. (1975) and Crowder (1976) were implemented and a better dual function value was achieved by the first algorithm for a given number of iterations. The step size in the direction of the subgradient is proportional to a parameter λ and the most effective rule for generating a decreasing sequence for λ is characterized by the parameters (λ_0, r_0, d, r_m), which can be described as follows (Held et al., 1974; Crowder, 1976). The sequence is started by letting $\lambda = \lambda_0$ for $r = r_0$ iterations. Both λ and r are then divided by d; the sequence now has this new value of λ for r iterations, at the end of which both λ and r are again divided by d. This process is repeated until $r \leqslant r_m$, thereafter, λ is divided by d every r_m iterations. The upper bound used in the subgradient algorithm is initially set to a large number and then it is updated according to the value of the incumbent solution. Computational tests have shown that suitable values for the parameters are (1.75, 25, 2, 10) and that the number of iterations should be set to 150.

4. Heuristic description

The heuristic suggested for solving CLSPP contains three phases. Given a capacity infeasible solution obtained by minimizing the Lagrangian problem the feasibility phase shifts production between periods and machines in an attempt to find a feasible solution for the problem with additional capacity, i.e., $(1+\alpha)C_j$, where α is chosen in the interval $[a, b]$ with uniform probability. The idea is to distribute the capacity excess among the periods. If the resulting solution is infeasible, then the feasibility phase is again applied, now with the original capacity C_j. Finally, the improvement phase starts from a feasible solution and searches for lower cost feasible solutions. The feasibility phase is applied to the solution of the Lagrangian problem at every step of the subgradient method. The proposed heuristic is summarized by the following algorithm:

Step 1. If the solution obtained by minimizing the Lagrangian problem is feasible go to *Step 6*.
Step 2. Increase the capacity of machine by the factor $(1 + \alpha)$, with α selected in the interval $[a, b]$ with uniform distribution;
Step 3. Apply the feasibility phase;
Step 4. If the solution is infeasible return to the original capacity and apply the feasibility phase.
Step 5. If a feasible solution is found, apply the improvement phase and go to *Step 6*; else, execute an iteration of subgradient optimization and go to *Step 1*;
Step 6. Update the incumbent solution.

4.1. Feasibility phase

Assume that a solution violates constraints (2), i.e. the capacity of at least one machine j is exceeded in at least one period t. A feasible solution is searched by moving production on machine j from period t to another period and another machine or to the same period t and another machine.

Let

$$\Delta(j,t) = \sum_{i=1}^{n} (b_{ij}x_{ijt} + f_{ij}y_{ijt}) - C_j,$$

and define $a^+ = \max\{a, 0\}$.

The capacity exceeded on machine j in period t is represented by

$$\text{Excess}(j,t) = \Delta(j,t)^+.$$

In order to eliminate Excess(j,t) one considers moving a quantity q of an item i produced in (j,t) to (jl,tl) with $jl \neq j$ and/or $tl \neq t$. The quantity q is chosen as

$$q = \min\{Z_{ijt}, Q_{ijt}\},$$

where Z_{ijt} = maximum amount of item i produced on machine j that can be transferred from period t to period tl. This amount depends whether $tl < t$ or $tl > t$, and

$$Q_{ijt} = \frac{\text{Excess}(j,t)}{b_{ij}}.$$

Note that if $Q_{ijt} \leqslant Z_{ijt}$, then Q_{ijt} represents the exact amount of item i that reduces Excess (j,t) to zero.

The selection of the quadruple (q,i,jl,tl) which denotes the quantity q of the item i to be moved to a machine jl in period tl is such that it minimizes the following criterion, which represents a tradeoff between cost variation and capacity violation:

$$\frac{\text{Cost\_variation} + \beta \cdot \text{Penalty}}{\text{Excess\_reduction}}. \tag{5}$$

The cost variation caused by the transference of the quantity q of the item i moved from (j,t) to (jl,tl) is given by

$$\text{Cost\_variation} = \frac{q \cdot h_i(t - tl) + (1 - y_{i,jl,tl})s_{i,jl} - \delta_1 s_{ij}}{(\text{solution\_value})/(Tmn)},$$

where solution value denotes the cost of the current solution, before performing the transference, and

$$\delta_1 = \begin{cases} 1, & \text{if } q = x_{ijt}, \\ 0, & \text{if } q < x_{ijt}. \end{cases}$$

The penalty in (5) is a nonnegative term that can be interpreted as a cost for overuse of capacity in (j,t) and (jl,tl), and is expressed as

$$\text{Penalty} = \left\{ \frac{\text{Excess\_after}(j,t)}{C_j} + \frac{\text{Excess\_after}(jl,tl) - \text{Excess\_before}(jl,tl)}{C_{jl}} \right\},$$

where

$$\text{Excess\_after}(j,t) = [\Delta(j,t) - (q\, b_{ij} + f_{ij}\delta_1)]^+,$$
$$\text{Excess\_after}(jl,tl) = [\Delta(jl,tl) + (qb_{i,jl} + (1 - y_{i,jl,tl})f_{i,jl})]^+,$$
$$\text{Excess\_before}(jl,tl) = \Delta(jl,tl)^+.$$

The factor β in (5) denotes the penalty weight, which is increased when a feasible solution cannot be found with its current value. Finally, the denominator Excess_Reduction in (5) is the difference between Excess_after (j,t) and Excess_before (j,t), i.e.

$$\text{Excess\_reduction} = \frac{qb_{ij} + f_{ij}\delta_1}{C_j}.$$

The feasibility phase consists of a number of p sequences of backward and forward steps. In the first sequence the penalty weight is set to β. If a feasible solution is not found then the weight is increased to 2β in the second sequence to give more importance to the capacity constraint violation. In the pth sequence, the weight becomes $p\beta$, and in case a feasible solution is not found, then either the problem has no feasible solution or the feasibility phase fails to produce a feasible solution.

4.2. Backward shifts

Production shifts from periods $t = T, T-1, \ldots, 2$ to earlier periods tl are examined in that order. For each pair machine-period (j,t) which exceeds the capacity C_j, one moves quantities q of items produced in (j,t) to other pairs (jl,tl) until the capacity constraint in (j,t) is satisfied. For a given item i the periods tl are such that $\tau \leqslant tl \leqslant t$, where $\tau = \max\{1,$ the last period before t in which there is production of item $i\}$, and if $tl = t$ then $jl \neq j$.

From constraints (1) it follows that the move of the quantity q of x_{ijt} to a period $tl < t$ causes the inventory levels $I_{ik}, k = tl, \ldots, t-1$ to be increased by q. Thus,

$$Z_{ijt} = x_{ijt}.$$

If after all moves, the capacity constraints in period one are met then a feasible solution is obtained. Otherwise, forward shifts are applied as described next.

During computational tests a kind of cycling was detected by the indefinite transfer of the same item between two machines in the same period. In order to avoid this, a cycle counter was included to prevent this situation. This counter is represented by a matrix Cycle (i,j) and its elements take on the value zero, when shifts are started in period t. At each shift of an item i to a machine j the element Cycle (i,j) is incremented by one and to avoid cycles we allow a shift of an item i to a machine j only if Cycle $(i,j) < 2$.

The backward procedure is described by the following algorithm:

Step 1. Let $t = T$;
Step 2. Initialize the cycle counter;
Step 3. If no machine has capacity violation, go to *Step 7*;
Step 4. Compute the best shift of item i produced in period t and machine j;
Step 5. Execute the shift;

Step 6. If the capacity of this machine is violated, go to *Step 4*, otherwise go to *Step 3*;
Step 7. Let $t = t - 1$;
Step 8. If $t \geqslant 2$ go to *Step 2*, otherwise stop.

At iteration t, nm operations are performed in *Step 2* to initialize the matrix of the cycle counter with elements equal to zero. In the worst case, we have n items produced on a single machine that can be moved to all m machines and all T periods, i.e., $O(nmT)$ operations are performed in *Step 4*. The best shift of an item i produced in period t and on machine j is computed by (5). At most T operations are performed in *Step 5* to update the inventory amount in each period. *Steps 4–6* correspond to an internal loop that moves at most n items while there are capacity violated machines, and hence its complexity is $O(n)$. *Steps 3–7* constitute a loop that is repeated at most $O(nm)$ times due to the cycle counter, i.e., an item can leave the same machine twice, thus in the worst case n items can leave twice m machines. Thus, the resulting complexity for the backward procedure is $O(n^3m^2T^3)$.

4.3. Forward shifts

Production shifts from periods $t = 1, 2, \ldots, T-1$ to later periods tl are examined in that order. For each pair machine-period (j, t) which exceeds the capacity C_j one moves quantities q of items produced in (j, t) to other pairs (jl, tl). For a given item i the periods tl are such that $t \leqslant tl \leqslant \tau$, where $\tau = \min\{T$, the first period after t in which there is production of item $i\}$, and if $tl = t$ then $jl \neq j$.

From constraints (1) it follows that the move of the quantity q of x_{ijt} to a period $tl > t$ causes the inventory levels $I_{ik}, k = t, t+1, \ldots, tl$ to be reduced by q. Since I_{ik} is nonnegative and nonincreasing, it follows that

$$Z_{ijt} = \min\{x_{ijt}, I_{i,tl-1}\}.$$

If after all moves the machine capacity constraints are met then a feasible solution is obtained. Otherwise, backward shifts are applied again. Cycling as described above can also occur here, and the complexity of the forward procedure is identical to the backward procedure.

The feasibility phase is described by the following algorithm:

Step 1. Let count = 1;
Step 2. If the solution obtained is feasible or count $> p$, stop;
Step 3. Apply the backward shifts;
Step 4. If the solution obtained is feasible, stop;
Step 5. Apply the forward shifts;
Step 6. Count = count + 1;
Step 7. Go to *Step 2*.

4.4. Improvement phase

This phase starts from a feasible solution and searches for lower cost solutions by performing backward and forward production shifts while maintaining capacity feasibility. In the backward stage initially, a pair machine-period (jl, tl) with capacity slack is identified and this is done by examining $tl = T, \ldots, 1$ and $jl = 1, \ldots, m$, in that order. One then considers moving production from the pairs (j, t) with $t = T, \ldots, tl$ and $j = 1, \ldots, m$ to the pair (jl, tl) with capacity slack. Let

$$\text{Slack}(jl, tl) = [-\Delta(jl, tl)]^+$$

denote the capacity slack of machine jl in period tl.

The quantity of a given item i produced in (j,t) to be transferred to (jl,tl) with $t \geqslant tl$ is given by

$$q = \min\left\{\frac{\text{Slack}(jl,tl) - (1 - y_{i,jl,tl})f_{i,jl}}{b_{i,jl}}, x_{ijt}\right\}. \tag{6}$$

The first term in (6) represents the maximum amount of item i that can be transferred to (jl,tl) without violating the capacity C_j, and the second term results from the production x_{ijt}. The selected quadruple (q,i,j,t) is the first one that yields cost reduction. If Slack(jl,tl) is still positive after the transference, the process is repeated in an attempt to further reduce the cost. The forward procedure is similar to the backward procedure, except by the quantity to be transferred

$$q = \min\left\{\frac{\text{Slack}(jl,tl) - (1 - y_{i,jl,tl})f_{i,jl}}{b_{i,jl}}, \min\{x_{ijt}, \min_{t\leqslant k\leqslant tl}\{I_{ik}\}\}\right\}. \tag{7}$$

5. Computational results

The heuristic was programmed in C and tests were run on 2.08 GHz AMD Athlon XP2600 + microcomputer, 1GB RAM and Linux operational system. Computational tests involve three experiments. In the first one, the heuristic is tested for the single machine CLSP and its performance is compared with the heuristics proposed by Trigeiro et al. (1989) and Gopalakrishnan et al. (2001) on instances provided by the first authors. In the second experiment, the heuristic is tested on randomly generated small instances of the CLSPP, which were solved to optimality by using the solver CPLEX 7.5.0. In the third experiment, the heuristic is tested on randomly generated larger instances of the CLSPP involving up to 50 items, 6 machines and 18 periods and the solutions quality are evaluated by the lower bound generated by the Lagrangian relaxation. In all experiments, for each instance the heuristic is applied ten times in order to check its robustness relative to the random capacity factor α. Suitable values for the parameters required by the heuristic were found by computational tests and are as follows: $\alpha \in [0.05, 0.20]$, $\beta = 0.15$ and $p = 15$.

5.1. Results for experiment 1

Table 1 shows the results for the instances of single machine CLSP, which are divided in three groups. Group EW contains 58 instances with 6 items and 15 periods, 6 instances with 4 items and 15 periods and 6 instances with 8 items and 15 periods. The group FG has 70 instances with 6 items and 15 periods and 71 instances with 6,12 and 24 items, and 15 and 30 periods. There are five problems for each combination of number of items and periods, except for the combination of 6 items and 15 periods for which there are 46 problems. Finally, Group X contains 540 instances with 20 periods and 10 items. Gopalakrishnan et al. (2001) applied the proposed tabu search algorithm, denoted (HG), to the groups FG and X of instances. Computational times for the proposed heuristic (H) and the heuristic developed by Trigeiro et al. (1989) (HT) are very small, not exceeding 2 seconds. The gaps in Table 1 are computed with respect to lower

Table 1
Computational results for the single machine CLSP

Group	Gap HT	Gap HG	Gap H (%)			Best (%)			Ties (%)		
			Mean	Worst case	Best case	Mean	Worst case	Best case	Mean	Worst case	Best case
EW	4.1	–	4.1	4.3	4.1	57	53	60	7	6	7
FG	4.2	3.2	4.4	4.6	4.3	47	44	50	7	6	9
X	2.3	4.0	2.5	2.6	2.5	35	32	36	16	15	16

bounds also provided by Trigeiro et al. (1989), whereas best and ties refer to the number of instances for which H outperformed HT, and the number of instances for which the heuristics reach the same least cost solution. First, note that the proposed heuristic is robust, since the mean, worst and best gaps among ten trials are very similar. The gaps for H and HT show that the heuristics are very competitive. For groups EW and FG, H is slightly superior to HT in terms of number of best solutions. On the other hand, HT performs better for group X. Finally, the gaps in Table 1 show that HG performs better than H and HT for group FG, but it is outperformed by H and HT for group X.

In the following, we present results for six instances of group FG, with 6, 12 and 24 items and 15 and 30 periods, one for each combination, which are considered difficult instances by Belvaux and Wolsey (2000). The optimal solutions for such instances were obtained by the solver CPLEX 7.5.0, spending on average 41 minutes in a station Sun Ultra 60. Table 2 shows the gaps for the solutions values found by HT and H relative to the optimal solution values. Computational times spent by the heuristic are below one second. Table 2 shows that heuristic H performs better than HT, in terms of mean gap for three instances.

5.2. Results for experiment 2

The heuristic was tested on a set of 80 small instances with number of items $n = 8$, number of machines $m = 2$, and number of periods $T = 4$, for which optimal solutions were obtained by using the solver CPLEX 7.5.0. The heuristic performance was evaluated in terms of the quality of the solution, the ability to find feasible solutions, and it is analyzed according to three factors: setup cost, setup time and capacity. Setup cost and setup time can be high or low and capacity can be normal or tight. For each combination of these three factors 10 instances were generated with cost parameters and machine capacities constant over time. The parameters were generated in an interval $[a, b]$ with uniform distribution and denoted $U[a, b]$:

- unit production cost (c_{ij}): $U[1.5, 2.5]$,
- setup cost (s_{ij}): $U[5.0, 95.0]$; low setup costs are generated in this interval and high setup costs are obtained by multiplying low setup costs by 10,
- unit inventory cost (h_i): $U[0.2, 0.4]$,
- unit processing time (b_{ij}): $U[1.0, 5.0]$,
- setup time (f_{ij}): $U[10.0, 50.0]$; low setup times are generated in this interval and high setup times are obtained by multiplying low setup time by 1.5,
- demand (d_{it}): $U[0, 180]$.

Capacity was generated as follows. The demand of each item in each period is split among the machines. The lot-for-lot policy is then applied to each machine and the average is taken over the number of machines and periods. This results in the following expression for the capacity in each period,

Table 2
Computational results for six instances of group G

	GAP HT	GAP H		
		Mean	Worst case	Best case
G30	3.5	5.3	5.8	4.6
G53	0.9	3.1	3.8	2.4
G57	0.4	0.2	0.2	0.1
G62	2.3	2.1	2.3	1.6
G69	1.3	0.8	0.8	0.5
G72	0.1	0.1	0.1	0.1

$$\text{Cap} = \frac{\sum_{t=1}^{T}\sum_{j=1}^{m}\sum_{i=1}^{n}\left(\frac{d_{it}}{m}b_{ij} + f_{ij}\right)}{mT}.$$

In the above expression, the setup of every item takes place in every machine and period, which generates too much capacity for each period, specially for four and six machines. Preliminary computational tests using Cap disclosed that the generated solutions utilized a low capacity level, on average 70% and 65% for low and high setup costs. In order to generate instances with mean capacity utilization around 80% for low setup cost and 2, 4 and 6 machines, we used the least squares method to fit a linear expression of the capacity as a function of the number of machines:

$$\text{Cap}' = (1.18 - 0.07m)\text{Cap}.$$

Tight capacity is obtained by multiplying Cap′ by 0.9.

In order to evaluate the quality of the heuristic solutions the following relative deviations are considered:

$$\text{GAP1} = \frac{z_{\text{H}} - z_{\text{L}}}{z_{\text{L}}} \cdot 100, \quad \text{GAP2} = \frac{z_0 - z_{\text{L}}}{z_{\text{L}}} \cdot 100, \quad \text{GAP3} = \frac{z_{\text{H}} - z_{\text{O}}}{z_{\text{O}}} \cdot 100,$$

where

z_{H} heuristic solution value,
z_{O} optimal solution value,
z_{L} lower bound provided by the Lagrangian relaxation.

Table 3 shows the heuristic behavior for the set of small instances. Note that GAP2 is larger for high setup costs, which seems to indicate the existence of a duality gap. This is probably due to the fact that for high setup costs the solution of the Lagrangian problem tends to have few machine setups, thereby resulting in a poor lower bound. GAP3 shows that the heuristic is better than the quality measure given by GAP1. Observe that the heuristic presents a better performance for low setup costs. For instances with tight capacity and high setup times, capacity constraints become tighter and as a result the number of feasible solutions obtained by the heuristic is smaller when compared to instances with tight capacity and low setup time. However, the gap of the feasible solutions is not altered. Running times for the heuristic are below one second for all instances, while CPLEX times varied between 1 and 8 seconds. The heuristic found the same number of feasible solutions as CPLEX, and FEA denotes the percentage of instances for which a feasible solution was found. We can also observe that the proposed heuristic is robust, since the mean, worst and best gaps are similar.

Table 3
Relative deviations for small instances with two parallel machines

Capacity/setup[a]	FEA	GAP1[b]			GAP2[b]			GAP3[b]		
		M	W	B	M	W	B	M	W	B
NLL	100	1.5	1.6	1.3	1.2	1.3	1.0	0.3	0.3	0.3
TLL	93	3.2	3.7	2.8	2.7	3.1	2.4	0.4	0.6	0.3
NHL	100	15.8	20.7	14.6	12.5	16.8	11.6	2.9	3.1	2.6
THL	99	32.6	39.0	30.2	29.0	35.3	27.4	2.6	3.1	2.1
NLH	100	2.1	2.5	1.9	1.8	2.2	1.6	0.3	0.3	0.2
TLH	73	5.4	6.1	4.8	4.2	4.7	3.6	1.2	1.4	0.9
NHH	100	20.6	24.8	17.3	17.2	21.3	14.1	2.8	3.3	2.6
THH	70	41.7	43.5	34.4	34.9	36.5	27.7	4.4	4.8	3.9

[a] Capacity/setup cost/setup time (N) normal, (T) tight, (L) low, (H) high.
[b] (M) Mean, (W) worst case, (B) best case.

5.3. Results for experiment 3

In this experiment, the heuristic was tested on a set of 2880 instances of sizes $n = 6, 12, 25, 50$; $m = 2, 4, 6$; $T = 6, 12, 18$, which were generated as above. Table 4 shows the heuristic performance for all sizes, according to the three factors: setup cost, setup time and capacity. The cells of the table associated to columns GAP1 and Time represent the average, in seconds, over 360 instances, replicated 10 times. The high percentage of feasible solutions found shows that the heuristic is very effective. Note that larger gaps are associated with high setup costs and times and this may be due to the duality gap, as observed in Table 3. As in Table 3 a smaller number of feasible solutions is found by the heuristic in the case of tight capacity and high setup time. GAP3 shows that setup times do not affect the solution quality, but high setup costs cause a small deterioration in the solution. The maximum time required to solve an instance is less than 11 minutes, which is reasonable since the largest instance has 5.400 binary variables. For all instances, the heuristic presented small gap variations for the 10 replications.

The average capacity utilization of the heuristic solution in the case of normal capacity is 79% and 72% for low and high setup costs, respectively. For tight capacity the solutions utilized on average 85% and 77% of the capacity for low and high setup costs, respectively. The setup time consumes at most 12% of the normal or tight capacity. The improvement procedure yields gains ranging from 1% to 23%, being more significant for high setup costs, irrespective of the setup times.

Table 4
Heuristic performance for larger instances

Capacity/setup[a]	FEA	GAP1	Mean time	Max. time
NLL	99	2.0	3.9	70.9
TLL	98	3.4	8.9	183.5
NHL	99	11.7	5.3	49.7
THL	98	18.5	7.8	108.1
NLH	98	2.5	4.9	48.8
TLH	95	4.2	14.4	631.7
NHH	98	13.8	5.9	62.3
THH	96	23.5	8.4	95.5

[a] Capacity/setup cost/setup time: (N) Normal; (T) Tight; (L) Low; (H) High.

Table 5
Normal capacity, low setup time and low setup cost

m	n	GAP1			Mean time	Improvement		Utilized capacity		FEA (%)
		M	W	B		Gain	Mean time	Mean	Mean setup	
2	6	2.2	2.5	1.8	0.3	1.1	0.1	79.9	5.8	99
	12	1.0	1.1	0.8	0.6	0.5	0.4	80.6	5.3	100
	25	0.3	0.4	0.2	1.5	0.2	0.8	83.7	5.2	100
	50	0.1	0.1	0.1	2.6	0.0	1.1	82.5	5.2	100
4	6	5.8	6.0	5.7	0.9	3.3	0.4	72.3	6.4	100
	12	1.8	2.1	1.5	1.7	1.6	0.9	75.6	5.4	100
	25	0.3	0.4	0.3	3.0	0.2	1.7	77.3	5.3	100
	50	0.1	0.1	0.1	6.2	0.0	3.0	80.7	5.4	100
6	6	10.4	11.2	9.6	1.6	4.8	0.6	64.2	6.9	90
	12	2.4	2.8	2.0	3.1	1.9	1.8	71.9	6.2	100
	25	0.9	1.0	0.7	6.9	0.8	3.9	79.8	5.8	100
	50	0.3	0.3	0.2	14.9	0.1	6.7	83.5	5.8	100

Table 6
Normal capacity, low setup time and high setup cost

m	n	GAP1			Mean time	Improvement		Utilized capacity		FEA (%)
		M	W	B		Gain	Mean time	Mean	Mean setup	
2	6	18.2	20.6	15.9	0.5	16.9	0.3	73.7	3.7	95
	12	7.1	8.3	5.9	0.9	6.3	0.6	71.7	3.0	100
	25	3.4	4.5	2.1	2.3	2.0	1.2	75.6	2.9	100
	50	1.0	1.1	1.0	6.1	0.5	1.9	76.6	2.9	100
4	6	31.2	33.0	29.5	0.9	18.1	0.5	62.4	4.5	100
	12	8.9	10.4	7.9	2.1	8.3	1.3	67.5	3.5	100
	25	3.1	3.6	2.6	4.3	1.7	2.6	73.3	3.2	100
	50	1.1	1.1	1.0	10.3	0.5	4.6	74.2	2.9	100
6	6	52.8	54.9	51.3	1.7	21.0	0.7	57.3	5.4	90
	12	16.6	17.6	15.0	3.7	10.9	2.4	63.8	4.1	100
	25	4.9	5.5	4.5	8.0	3.0	4.9	71.9	3.5	100
	50	1.7	2.0	1.6	17.4	0.8	9.0	75.6	3.2	100

Regarding the parameters that define the size of the generated instances, the heuristic behaves as follows. The increase of the number of periods does not affect the solution gap, and computational times increases accordingly. Tables 5 and 6 illustrate the effect of the number of items and number of machines on the heuristic performance, for instances with 12 periods, low setup time and low/high setup costs. For the remaining cases of different number of periods, high setup time and low/high setup costs the behavior of the heuristic is similar to that presented in Tables 5 and 6. These tables show the mean, worst and best gaps over 10 replications, the mean computational time, the mean gain and mean computational time due to the improvement phase, the mean capacity utilization and the mean capacity used for the setup time.

Tables 5 and 6 show that the solution gap decreases with the number of items, sometimes in a significant way as in the case of high setup cost and six machines, where the mean gap is 52.8% for 6 items and 1.7% for 50 items. The mean gap gain that results from the improvement phase is more significant for 6 and 12 items and the required computational time is low. In addition, the heuristic is able to find a feasible solution for almost all instances.

Comparing the results of Table 5 with those of Table 6, we note that the mean gap is higher for high setup cost, which is probably due to a duality gap. The mean gain is also higher in this case, due to the merge of setups. Finally, high setup cost implies fewer setups and consequently, a smaller setup time.

6. Conclusion

This paper dealt with the capacitated lot-sizing problem involving the production of multiple items on unrelated parallel machines. A simple heuristic based on the Lagrangian relaxation of the capacity constraints and subgradient optimization was proposed. Despite its simplicity it showed to be competitive with heuristics from the literature for the single machine capacitated lot-sizing problem. Results for parallel machines showed that the heuristic is fast and is able to find a high number of feasible solutions with high quality.

Acknowledgements

This research was partially funded by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior (CAPES) and

Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP). The authors are grateful to Dr. Trigeiro for supplying the code of his algorithm and the problem instances. We are also indebted to an anonymous referee for valuable suggestions.

References

Armentano, V.A., França, P.M., Toledo, F.M.B., 1999. A network flow model for the capacitated lot-sizing problem. OMEGA 27, 275–284.

Bahl, H.C., Ritzman, L.P., Gupta, J.N.D., 1987. Determining lot sizes and resources requirements: A review. Operations Research 35, 329–345.

Belvaux, G., Wolsey, L.A., 2000. BC-PROD: A specialized branch-and-cut system for lot-sizing problems. Management Science 46, 724–738.

Belvaux, G., Wolsey, L.A., 2001. Modelling Practical Lot-Sizing Problems as Mixed-Integer Programs. Management Science 47, 993–1007.

Camerini, P.M., Frata, L., Maffioli, F., 1975. On improving relaxation methods by modified gradient techniques. Mathematical Programming Study 3, 26–34.

Carreno, J.J., 1990. Economic lot scheduling for multiple products on parallel identical processors. Management Science 36, 348–358.

Crowder, P., 1976. Computational improvements for subgradient optimization. Symposia Matematica 19, 357–372.

Diaby, M., Bahl, H.C., Karwan, M.H., Zionts, S., 1992a. Capacitated lot-sizing and scheduling by Lagrangian relaxation. European Journal of Operational Research 59, 444–458.

Diaby, M., Bahl, H.C., Karwan, M.H., Zionts, S., 1992b. A Lagrangian relaxation approach for very-large-scale capacitated lot sizing. Management Science 38, 1329–1340.

Drexl, A., Kimms, A., 1997. Lot sizing and scheduling—survey and extensions. European Journal of Operational Research 99, 221–235.

Evans, J.R., 1985. An efficient implementation of the Wagner–Whitin algorithm for dynamic lot-sizing problem. Journal of Operations Management 5, 229–235.

Gopalakrishnan, M., Ding, K., Bourjolly, J.M., Mohan, S., 2001. A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. Management Science 47, 851–863.

Held, M., Wolfe, P., Crowder, P., 1974. Validation of subgradient optimization. Mathematical Programming 6, 62–88.

Hindi, K.S., Fleszar, K., Charalambous, C., 2003. An effective heuristic for the CLSP with set-up times. Journal of the Operational Research Society 54, 490–498.

Hung, Y.F., Hu, Y.C., 1998. Solving mixed integer programming production planning problems with setups by shadow price information. Computers and Operations Research 25, 1027–1042.

Hung, Y.F., Hu, Y.C., 2003. Using tabu search with ranking candidate list to solve production planning problems with setups. Computers and Industrial Engineering 42, 615–634.

Jans, R., Degraeve, Z., 2004. Improved lower bounds for the capacitated lot-sizing problem with setup times. Operations Research Letters 32, 185–195.

Johnson, L.A., Montegomery, D.C., 1974. Operations Research in Production Planning, Scheduling and Inventory Control. John Wiley & Sons, New York.

Karimi, B., Fatemi Ghomi, S.M.T., Wilson, J.M., 2003. The capacitated lot-sizing problem: A review of models and algorithms. OMEGA 31, 365–378.

Kuik, R., Salomon, M., Wassenhove, 1994. Batching decisions: Structure and models. European Journal of Operational Research 75, 243–263.

Lasdon, L.S., Terjung, R.C., 1971. An efficient algorithm for multi-item scheduling. Operations Research 19, 946–969.

Lozano, S., Larraneta, J., Onieva, L., 1991. Primal-dual approach to the single level capacitated lot-sizing problem. European Journal of Operational Research 51, 354–366.

Maes, J., McClain, J.O., Van Wassenhove, L.N., 1991. Multilevel capacitated lotsizing and complexity. European Journal of Operational Research 53, 131–148.

Özdamar, L., Birbil, S.I., Portmann, M.C., 2002. Technical note: New results for the capacitated lot-sizing problem with overtime decisions and setup time. Production Planning and Control 13, 2–10.

Sung, C.S., 1986. A single-product parallel-facilities production-planning model. International Journal of Systems Science 17, 983–989.

Souza, K.X.S., Armentano, V.A., 1994. Multi-item lot-sizing by a cross decomposition based algorithm. Annals of Operations Research 50, 557–574.

Trigeiro, W.W., 1987. The effect of setup time on production lot sizes. Production and Inventory Management 28, 50–52.

Trigeiro, W.W., Thomas, L.J., McClain, J.O., 1989. Capacitated lot sizing with setup times. Management Science 35, 353–366.
Van Roy, T.J., 1983. Cross decomposition for mixed integer programming. Mathematical Programming 25, 46–63.
Wagner, H.M., Whitin, T.M., 1958. Dynamic version of the economic lot size model. Management Science 5, 89–96.
Wolsey, L.A., 2002. Solving Multi-Item Lot-Sizing Problems with an MIP Solver Using Classification and Reformulation. Management Science 48, 1587–1602.