IIE Transactions (2000) **32**, 1043–1057

The capacitated lot sizing problem with overtime decisions and setup times

LINET ÖZDAMAR and MEHMET ALI BOZYEL

Department of Industrial Engineering, Doğus University, Gayrettepe Emekli Subay Evleri, 23/5 Istanbul, Turkey
E-mail: lozdamar@hotmail.com

Received September 1996 and accepted September 1999

The Capacitated Lot Sizing Problem (CLSP) consists of planning the lot sizes of multiple items over a planning horizon with the objective of minimizing setup and inventory holding costs. In each period that an item is produced a setup cost is incurred. Capacity is limited and homogeneous. Here, the CLSP is extended to include overtime decisions and capacity consuming setups. The objective function consists of minimizing inventory holding and overtime costs. Setups incur costs implicitly via overtime costs, that is, they lead to additional overtime costs when setup times contribute to the use of overtime capacity in a certain period. The resulting problem becomes more complicated than the standard CLSP and requires methods different from the ones proposed for the latter. Consequently, new heuristic approaches are developed to deal with this problem. Among the heuristic approaches are the classical HPP approach and its modifications, an iterative approach omitting binary variables in the model, a Genetic Algorithm approach based on the transportation-like formulation of the single item production planning model with dynamic demand and a Simulated Annealing approach based on shifting family lot sizes among consecutive periods. Computational results demonstrate that the Simulated Annealing approach produces high quality schedules and is computationally most efficient.

1. Introduction

The single stage Capacitated Lot Sizing Problem (CLSP) is a NP-hard problem (Bitran and Yanasse, 1982) in production planning that arouses continuous interest among researchers (Kırca, 1990; Kırca and Kökten, 1994; Lambrecht and Vanderveken, 1979; Maes and van Wassenhove, 1986). In the CLSP, time-varying deterministic item demands are assumed to take place over a finite planning horizon, limited capacity is shared by all items produced in each period and no backorders are permitted. Each period that an item is produced a setup cost is incurred. The objective function minimizes setup and inventory carrying costs. In some formulations setups also consume capacity (Kim and Mabert, 1995; Trigeiro *et al.*, 1989). Heuristic solution methods proposed for this problem usually involve a period-by-period approach (Dixon and Silver, 1981; Doğramacı *et al.*, 1981; Günther, 1987; Lambrecht and Vanderveken, 1979) where lot sizes for items are determined based on a cost saving criterion. Each period, future demand is scheduled to be produced in that period until no further costs can be saved or the capacity of that period is exhausted. There are also alternative heuristic approaches based on mathematical programming (Cattrysse *et al.*, 1990; Thizy and van Wassenhove, 1985; Trigeiro, 1987). Kırca and Kökten (1994) propose an iterative approach called the item-by-item approach where an iteration consists of scheduling a subset of items consuming the available capacity gradually at each step.

The model proposed here extends the CLSP in two respects: (i) it assumes two sources of capacity, regular time and overtime, resulting in the loss of capacity homogeneity; Dixon *et al.* (1983) consider overtime decisions in the capacitated lot sizing problem, but their case is restricted to a single item; (ii) setups consume capacity and they do not incur costs other than labor costs. Additional costs related to setups, such as scrap or cleansing material costs, are not accrued. Therefore, setup costs are not explicitly stated in the objective function. Rather, setups incur costs only if they contribute to overtime. Thus, the objective function consists of inventory carrying and overtime costs while the constraints guarantee that demand is satisfied in each period and regular time and overtime capacities are not exceeded. In fact, we prefer calling this problem the capacitated multi-family lot sizing problem with overtime decisions, because if end items with the similar setup requirements are grouped into families, we have a much more compact mathematical formulation of the model given in Table 1 and the number of binary variables reduce considerably. The proposed model has a wide application area both in process industries and

Table 1. Mathematical formulation of the capacitated family lot sizing problem with overtime decisions and setup times

Model P: Min $\sum_t \sum_j I_{jt} h_j + \sum_t ov O_t$

subject to

$I_{j,t-1} + Y_{jt} - I_{jt} = d_{jt} \quad \forall j, t,$

$\sum_j (p_j Y_{jt} + w_{jt} s_j) \leq C_t + O_t \quad \forall t,$

$O_t \leq CO_t \quad \forall t$

$Y_{jt} \leq M w_{jt} \quad \forall j, t,$

$w_{jt} = 0, 1; \; Y_{jt}, I_{jt}, O_t \geq 0 \quad \forall j, t,$

where

d_{jt} = demand of family j in period t;
p_j = process time per unit of family j (h);
ov = overtime cost per hour;
h_j = holding cost per unit of family j per period;
C_t = regular time capacity in period t;
CO_t = overtime capactiy in period t;
I_{jt} = inventory of family j held at the end of period t,
Y_{jt} = production lot size of family j in period t;
O_t = overtime capactiy used in period t;
s_j = setup time required for family j
w_{jt} = binary variable $\left(= \begin{cases} 1, \text{ if family } j \text{ is produced in period } t; \\ 0, \text{ otherwise} \end{cases}\right)$
M = a large number.

in discrete manufacturing. When discrete manufacturing is considered the variables representing production quantities should be integral. However, in model P we ignore such integrality constraints, because the model becomes much more difficult to solve and furthermore, since we concentrate on heuristic approaches to be used in practice, rounding off real valued production quantities seems to be a common sense approach.

The model involves a trade-off between overtime and inventory carrying costs while setups may or may not result in overtime. This trade-off occurs because in periods where demanded quantities result in idle capacity, it might be beneficial to produce more and hold inventory rather than to produce the demanded amounts using overtime capacity during the peak periods. Naturally, such decisions depend on the ratio of unit holding cost to unit overtime cost. The implicitness of the setup costs complicates the optimization process specifically when capacity is moderately abundant. On the other hand, when capacity constraints are quite tight it is hard to find feasible solutions. Due to capacity consuming setups, it is not easy to conceive of feedback/lookahead feasibility mechanisms such as the ones described in Maes and van Wassenhove (1985) which enable period-by-period heuristics to avoid capacity violations. In fact, the feasibility problem of the CLSP with setup times is NP-Complete as shown by Garey and Johnson (1979).

2. Solution methods

The solution approaches proposed here do not consist of period-by-period "fill up to capacity" priority rules, because of issues of infeasibility which may arise in future periods due to setup times and also because it is hard to deduce rules which would result in near-optimal solutions when the capacity is not homogeneous. While increasing the lot sizes of families, the decision depends on the prediction of capacity bottlenecks to take place in future periods due to rises in demand and whether it is more advantageous to hold inventory in the current period instead of allocating (probably overtime) capacity to an extra setup in the future bottleneck period. In the existence of multiple families to be scheduled over the planning horizon, the decision becomes quite complicated. Thus, to deal with this problem, we propose a number of different approaches including the Hierarchical Production Planning (HPP) approach, an iterative relaxation approach, a Genetic Algorithm (GA) and a Simulated Annealing (SA) approach. In the following sections, we describe these solution approaches and compare their results against the optimal (best) ones on three sets of randomly generated test problems.

2.1. *The hierarchical production planning approach*

In the HPP approach we aggregate the family demands in each period into a "type" demand over the planning horizon and calculate the weighted average of family process times with respect to their demands to result in a single aggregate process time. Furthermore, by reserving a percentage of regular production capacity for setup times we avoid explicit binary setup variables in the capacity constraints and end up in the model P1 in Table 2. The setup allowance percentage, A, is calculated by multiplying the ratio of total setup time to total required production time by the expected number of seasons, N, in the planning horizon which is identified during the forecasting phase of the planning process.

2.1.1. *The filling procedure*

Model P1 is solved first to determine the aggregate type production quantities (and thereby, overtime decisions) over the planning horizon. Then, the lot sizes of triggering families are calculated for every period with their sum always equal to the optimal type production quantity X_t^*. A family triggers in period t if its lower bound lb_{jt} is positive. (lb_{jt} equals the net demand ($= \max\{0, d_{jt} - I_{jt-1}\}$) for family j in period t.) Family disaggregation is carried out by using the Filling Procedure (FP) discussed

Table 2. Aggregate mathematical formulation used in the HPP approach

Model P1: Min $\sum_t (hI_t + ovO_t)$

subject to

$I_{t-1} + X_t - I_t = D_t \quad \forall t,$

$PX_t \leq (1 - A)C_t + O_t \quad \forall t,$

$O_t \leq CO_t \quad \forall t,$

$X_t, I_t, O_t \geq 0 \quad \forall t,$

where

D_t = aggregate type demand in period $t\left(=\sum_j d_{jt}\right)$;

P = aggregate process time per unit type $\left(=\sum_t\sum_j d_{jt}p_j \Big/ \sum_t\sum_j d_{jt}\right)$;

A = setup allowance percentage for capacity $\left(=N\sum_j s_j \Big/ \sum_t\sum_j d_{jt}p_j\right)$ where

N = number of seasons in planning horizon;

h = aggregate holding cost $\left(=\sum_t\sum_j d_{jt}h_j \Big/ \sum_t\sum_j d_{jt}\right)$;

I_t = aggregate inventory held at the end of period t;

X_t = aggregate production quantity in period t.

in previous work (Özdamar *et al.* 1996, 1998). However, in the latter articles, the objective function simply consists of minimizing the number of annual setups ($=\sum s_j\, ub_j/y_j$ where ub_j is the family upper bound equal to the net demand till the end of the planning horizon). In this case, no capacity restrictions exist and family disaggregation is carried out once for a single period, therefore the subscript t is dropped from ub_j and the family lot size y_j. Constraints are used to equate the total production quantity to X^* and to restrict individual family lot sizes within the interval defined by lb_j and ub_j. Thus, FP is executed once for a single period and it is assumed that X^* is known somehow. FP increases family lot sizes one at a time with prespecified increments $\delta_j = (ub_j - lb_j)/SN$, where SN is a user defined step number, until the sum of all family lot sizes is equal to X^*. For the objective of reducing the number of setups, families are selected for increasing their lot sizes at each iteration according to an approximate gradient measure ∇_j, where ∇_j represents the saving on setup costs and is given by $\nabla_j = (s_j ub_j/y_j) - s_j\, ub_j/(y_j + 1)$.

An optimal solution is achieved by FP for every test instance randomly generated in Özdamar *et al.* (1996) when the Step Number, SN, which controls the magnitude of lot size increments, is set to 500. However, the degree of suboptimality that FP would cause in a multi-period planning horizon (model P in Table 1) has not been measured by the authors.

Furthermore, FP guarantees that no backorders take place in only two periods: the current period t, and the last period, T, of the planning horizon. This is because lot sizes are restricted to be between their lower and upper bounds, which consider the current period's net demand and the sum of net demand until period T, respectively. In FP the largest lot size is usually assigned to the family with the largest ∇_j. If this family has a small demand in the next period and a family with a smaller ∇_j has a larger demand, then, FP cannot foresee the situation and holds inventory for the larger ∇_j family and results in backorders for the other family in the next period. Consequently, FP has no guarantee on backorders in periods between the current one and the last.

For the purpose of eliminating backorders throughout the planning horizon, in this study, a Search/Cover procedure is added to FP. The added procedure checks if backorders exist in the current planning period and if they do, the production quantities of families which have positive inventory in that period are transferred in ascending order of $s_j\, ub_{jt}$ to the ones with backorders. If the families with positive inventory in the current period cannot cover all backorders, then the families with positive inventory in the previous period are used for covering the deficit and their lot sizes are transferred to the lot sizes of the deficit families in the previous period. In this manner, the deficit is covered by searching families with positive inventory in former periods until the first period is reached. Since no backorders exist in the aggregate plan, the Search/Cover procedure guarantees to cover all backorders.

In the multi-period context considered here, FP is executed once in each period of the planning horizon (δ_j, lb_j, ub_j all gain a subscript of t and are recalculated each period) and if backorders exist in a period t, the Search/Cover procedure is executed and then FP resumes disaggregation in the next period $t + 1$. The pseudocodes for the HPP main program and FP procedure can be found in Table 3.

2.1.2. *The modified filling procedure*

Since FP does not consider inventory and overtime costs while increasing lot sizes, we modify FP to result in the Modified FP, denoted as MFP, by changing the family selection rule in order to consider explicitly the trade-off between inventory and overtime costs. Families are selected for incrementing their lot sizes in ascending order of z_{jt} where

$$z_{jt} = \sum_{m=t}^{t'-1} I'_{jm}h_j/(ov(p_j\delta_{jt} + s_j)),$$

where I'_{jm} is the inventory that would have been held in period m if the lot size of family j were increased by δ_{jt} in the current period t; t' is the earliest bottleneck period where overtime will be required considering the net

Table 3. Pseudocodes for HPP main program, FP, MFP and Search/Cover

```
HPP Main Program
begin
   Solve Aggregate Model P1;
   Read $X_t^*$ for $t = 1..T$
   for $t = 1$ to $T$ do
   begin
      Calculate $lb_{jt}$, $ub_{jt}$, $\delta_{jt}$; ($lb_{jt}$ = net demand in period $t$; $ub_{jt}$ = net demand from period $t$ to period $T$)
      FP (MFP);
      Calculate Inventory;
      If $X_t' < 0$ then Search/Cover;
      Check Capacity Feasibility;
   end;
   Calculate Cost;
end;
Filling Procedure {**Modified Filling Procedure**}
begin
   Identify $JT_t$ : set of triggered families; {$\boldsymbol{JT_t}$**:all families**};
   $X_t' = X_t^*$;
   $y_{jt} = lb_{jt}$ for all $j \varepsilon JT_t$; Reduce Capacity ; Reduce Distributable Quantity $X_t' = X_t' - \sum_{j\varepsilon JT} y_{jt}$
   If $X_t' < 0$ then $y_{jt} = 0$ for all $j\varepsilon JT_t$; Redistribute $X_t^*$ among $j\varepsilon JT_t$ in descending order of $\nabla_j$;
   If $X_t' > 0$ then
      repeat
         Identify $JC$:set of families in $JT_t$ such that ($y_{jt} + \delta_{jt} \leq ub_{jt}$) and ($\delta_{jt} \leq X_t'$){**and (sufficient capacity exists)**};
         Sort $JC$ in descending order of $\nabla_j$; {**Sort JC in ascending order of** $\boldsymbol{z_{jt}}$};
         Increment $y_{jt}$, where $jt$ is the top index in the list JC, by $\delta_{jt}$;
         Reduce Capacity; Reduce $X_t'$;
      until $JC = \varnothing$;
   If $X_t' > 0$ then
      repeat
         Identify $JC$: set of families in $JT_t$ such that $y_{jt} < ub_{jt}$;
         Sort $JC$ in descending order of $\nabla_j$; {**Sort JC in ascending order of** $\boldsymbol{z_{jt}}$};
         Increment $y_{jt}$, where $jt$ is the top index in the list $JC$, by min $\{ub_{jt} - y_{jt}, X_t' - y_{jt}\}$;
         Reduce $X_t'$;
      until $X_t' = 0$;
end;
Procedure Search/Cover;
begin
   Identify total amount of backorder, $B = -X_t'$;
   Identify $JB$: set of families with $I_{jt} < 0$;
   Identify $JI$: set of families with $I_{jt} > 0$;
   $r = t$;
      repeat
         repeat
            Identify $JJI_r$: set of families in $JI$ with $y_{jr}$ and $I_{jr} > 0$;
            Sort $JJI_r$ in ascending order of $\nabla_j$; {**Sort** $\boldsymbol{JJI_r}$ **in descending order of** $\boldsymbol{z_{jr}}$};
            Transfer min$\{y_{jr}, -I_{kt}, I_{jr}\}$ from the family $j$ at the top of the list to the families $k \varepsilon JB$ with $I_{kt} < 0$ one at a time in
            order of index; {Reduce $B$; Update relevant $y_{jr}$, $I_{jr}$, $y_{kr}$, $I_{kt}$}
         until ($B = 0$) or ($JJI_r = \varnothing$);
      $r = r - 1$;
   until $B = 0$;
end;
```

demands in future periods. The priority rule represents the trade-off between the holding cost caused by a lot size increment δ_{jt} in the current period t, to be incurred up to the bottleneck period t', and the overtime cost which would be incurred in case a production quantity δ_{jt} is assigned to family j in the bottleneck period enforcing a setup time.

A distinctive feature of MFP is that not only are the lot sizes of triggered families increased every period, but all families are subject to a positive lot size assignment. The reason is that under capacity restrictions nontriggered families may also need to continue building up inventories if their demand in the near-future is quite high and above available capacity.

Since setup times are not treated in an exact manner in the aggregate model P1 (to know the exact amount of allowance for setup times in each period we have to determine the optimal solution first!), but by a rough estimate of A, there might be some capacity violations in the disaggregated family solution provided by FP and MFP when the aggregate production quantity, X_t^*, is adhered to every period. Therefore, we include a capacity check in MFP, but not in FP, so that we preserve the original FP algorithm suggested in Özdamar *et al.* (1996). In MFP family lot sizes are incremented if the sum of regular and overtime capacities are not exceeded. However, if X^* cannot be totally distributed among families due to capacity restrictions, then lot sizes are increased without activating the capacity check. Thus, the capacity check included in MFP does not guarantee capacity feasibility as long as the aggregate plan is strictly adhered to. In Table 3, the modifications made on FP are indicated in bold type within parentheses and these result in MFP.

2.1.3. *The iterative HPP approach*

The performance of the HPP approach does not only depend on the selection of families for lot size incrementing, but also on the accuracy of the aggregate model P1. The latter is partially based on how close the setup allowance percentage A is to the optimal one. The value of A might be too restrictive or too large. To overcome this difficulty an Iterative HPP procedure is proposed, denoted by IHPP, where the value of A starts from zero and is gradually increased at each iteration. A solution is obtained by solving the aggregate model P1 with the original regular time capacity and then utilizing MFP for disaggregating the resulting type production quantities among families in every period. Then, the overtime caused by setup times only is calculated for every period, summed up and divided by the number of periods in the planning horizon, T, resulting in the average overtime caused by setups, RCD. In the next iteration every period's regular time capacity is decreased by RCD. Model P1 is re-solved with the latter reduced capacity and MFP finds another solution. These iterations continue until either the cost of a solution is higher than the incumbent best cost, or the aggregate model P1 results in an infeasible solution, or the solution found by MFP violates the sum of regular and overtime capacity. The aim of the iterative solution procedure is to obtain lower capacity utilization peaks and smooth out the capacity utilization profile over the planning horizon so that the amount of overtime caused by setups is reduced. The reduction of regular time capacity not only saves on the magnitude of lot sizes but may also reduce the number of setups taking place during the planning horizon.

Mathematically, RCD is expressed as:

$$RCD = \sum_t \max\{0, CR_t - O_t^* - C_t\}/T,$$

where

CR_t = capacity requirements including setup times resulting from MFP solution in period t,

O_t^* = optimal overtime quantity in period t resulting from the solution of the model P1.

The pseudocode for the main program of the IHPP approach imbedding MFP, is given in Table 4.

2.1.4. *An example*

An illustrative example with four families to be scheduled during a planning horizon of six periods is given in Table 5. The overtime cost is equal to 17/hour and regular and overtime capacities are both 2800 hours per period. The optimal cost for this example is identified as 59 708 by the optimization package Lindo.

The aggregate model P1 is constructed with an aggregate process time of 2.7 hours/unit, and the regular time capacity is decreased to 2471.8 hours by the setup allowance A which is calculated as 11.75% using an estimate of three seasons in the planning horizon. The aggregate holding and overtime costs are 14.1 and 17, respectively. Demand is aggregated into 1488, 1103, 327, 904, 1306 and 924 units, from the first to the sixth periods respectively. The aggregate model P1 is solved to result in the production quantities and inventories found in Table 6.

Table 4. Pseudocode for the main program of iterative HPP approach with MFP

```
IHPP Main Program
begin
  $C_t' = C_t$;
  MinCost = $\infty$;
  Solve Aggregate Model P1 with $C_t$;
  repeat
    Read $X_t^*$, $O_t^*$ for $t = 1..T$
    for $t$ = 1 to $T$ do
    begin
      Calculate $lb_{jt}$, $ub_{jt}$, $\delta_{jt}$;
      MFP;
      Calculate Inventory;
      If $X_t' < 0$ then Search/Cover;
      Check Capacity Feasibility;
    end;
    Calculate $Cost$, $RCD$;
    If ($Cost$ < $MinCost$) and (Capacity Feasible) then
    $MinCost$ = $Cost$;
    $C_t' = C_t' - RCD$;
    Solve Aggregate Model P1 with $C_t'$;
  until (Model P1 is Infeasible) or (MFP Solution Violates
  Capacity ) or ($Cost$ > $MinCost$)
end;
```

Table 5. Data for the example

Family j	s_j	p_j	h_j	d_{j1}	d_{j2}	d_{j3}	d_{j4}	d_{j5}	d_{j6}
1	148	3	11	481	279	185	355	132	322
2	183	2	19	171	183	29	132	383	319
3	195	4	10	200	344	64	73	460	175
4	120	2	17	636	297	49	344	331	108

Iterations of the HPP approach with FP are demonstrated in Table 7 for period 3 where $SN = 3$. Up to period 3 each family's lot size is exactly equal to its demand, since no inventory is held for the first and second period in the aggregate solution. In period 3, the second family is increased by its stepsize in the second iteration since it has the largest ∇_j and the remainder of X_3^* is assigned to the fourth family in the third iteration, since all stepsizes are greater than $X_t' = 108.5$ and ∇_2 becomes 1.75 after y_{23} is set to 308. HPP + FP finds a capacity feasible solution with a cost of 69 587.

The activation of the Search/Cover procedure is demonstrated in Table 8 where the parameter *SN* is changed to 100 and the example is solved again to discover that in the fifth period the third family has negative inventory and the others have a positive inventory. The Search/Cover procedure starts transferring lot sizes in order of ascending ($s_j\ ub_{jt}$). Namely, the inventories of the fourth and second families are driven to zero in period 5 and their excessive lots are reduced. Since the total lot size transferred to the third family (32 + 16.48) is not enough to cover the deficit of (53.44), and since the first family has a lot size of zero in the fifth period, the procedure reduces the first family's lot size in the fourth period by the necessary amount and transfers it to the third family in the fourth period although it had not triggered then.

Table 6. Solution to the aggregate model P1

Period	*1*	*2*	*3*	*4*	*5*	*6*
X_t^*	1488	1103	714.55	915.480	915.48	915.48
I_t^*	0	0	387.55	399.037	8.510	0.00

The solution found by HPP + MFP procedure is demonstrated in Table 9 where $SN = 100$. The Search/Cover procedure has not been activated during the solution process. The bottleneck period found by MFP is the fifth period and inventory holding costs are calculated accordingly. Notice that although the third family is not triggered in period 4, it is produced. The solution has a cost of 71 517.

The aggregate solutions and adjusted capacities found by the IHPP approach at each iteration are given in Table 10. Notice that with each iteration a smoother production level is obtained throughout the planning horizon. The best cost found by IHPP is 73 407.

2.2. *The iterative approach for the original problem*

A second approach is to deal with the original linear mixed integer model P by ignoring the setup consumption expressions in the capacity constraints and excluding the last set of constraints related to the binary variables, re-

Table 7. Iterations made by HPP + FP in the third period

Family	∇_j	lb_{j3}	ub_{j3}	δ_{jt3}	y_{j3} *(1st iteration)*	y_{j3} *(2nd iteration)*	y_{j3} *(3rd iteration)*
1	4.23	185	994	270	185	185	185.00
2	181.00	29	863	279	29	308	308.00
3	36.00	64	772	236	64	64	64.00
4	39.00	49	832	261	49	49	157.55

Table 8. The activation of the Search/Cover procedure in the fifth period with $SN = 100$

	Period 1	*Period 2*	*Period 3*	*Period 4*	*Period 5*	*Inventory at period 5*
y_{jt} *before search/cover*						
Family 1	481	279	185.00	500.48	0.00	13.48
Family 2	171	183	191.00	0.00	369.48	16.48
Family 3	200	344	185.55	0.00	358	−53.44
Family 4	636	297	153.00	415.00	188.00	32.00
y_{jt} *after search/cover*						
Family 1	481	279	185.00	495.51	0.00	8.51
Family 2	171	183	191.00	0.00	353.00	0.00
Family 3	200	344	185.55	4.96	406.44	0.00
Family 4	636	297	153.00	415.00	156.00	0.00

Table 9. Solution found by HPP + MFP

	Period 1	*Period 2*	*Period 3*	*Period 4*	*Period 5*	*Period 6*
Family 1	481	279	311.00	339.00	22.00	322.00
Family 2	171	183	110.00	123.00	311.00	319.00
Family 3	200	344	188.55	110.48	304.48	168.48
Family 4	636	297	105.00	343.00	278.00	106.00

sulting in the linear model P2. Model P2 excludes setup times completely, and therefore, setup times are handled through the iterative approaches described in IHPP. The pseudocode for the Iterative Approach for the Original Problem (IAOP) is given in Table 11. An approach similar to IAOP is successfully utilized by Özdamar and Yazgaç (1997) in generating a Master Production Plan at the family level for a leading kitchen cupboard manufacturer in Turkey.

Considering the example in Table 5, procedure IAOP results in the following regular capacity, cost pairs: (2801, 77 494), (2309, $\underline{67\ 233}$) and (1954, 71 469) and stops in three iterations. The underlined value is the lowest cost.

2.3. *The genetic algorithm approach*

Genetic Algorithms (GAs) are robust search techniques which have been applied to a variety of combinatorial problems in Operations Research (Michalewicz, 1994), especially in the area of scheduling. The principle agents of a GA are the chromosome encoding, which affects the performance of the GA in terms of solution quality and computational efficiency, and the reproduction, crossover and mutation operators, which enable the GA to intensify towards good solutions and diversify to skip local optima. A GA applies reproduction, crossover and mutation repeatedly for a prespecified number of generations after randomly generating an initial population. The best feasible solution is always preserved.

A new GA approach is developed here for the lot sizing problem with overtime decisions as an alternative to the HPP approach and IAOP with the hope that it results in solutions much closer to the global optimum. To our knowledge, there have been no previous GA approaches proposed for the lot sizing problem.

The solution to the capacitated family lot sizing problem can be represented in a tabular form similar to the transportation type of presentation found in the capacitated single-item production planning problem with dynamic demand (Bowman, 1956). In the latter context, the transportation matrix is of size $T \times T$, and for instance, a positive production quantity in cell (2,4) implies that the capacity at period 2 is used to satisfy part or all of the demand in period 4. Thus, each column represents a demand period and each row represents a supply period. In our problem, however, there exist multiple families and two sources of capacity. So, each cell in the matrix consists of $2F$ (F = number of families) elements (production quantities), F of them representing production in regular time and the remaining F representing production in overtime. Each column has F different demands for F families. We note that if we did not have capacity consuming setups and multiple families, then a simple procedure which satisfies demand by using the cheapest source of capacity starting from the first period would result in an optimal solution. Although our model is much more complicated, the above tabular representation is a valid manner of representing a feasible solution.

The manner in which the tabular solution is generated is as follows: An $F \times T$ matrix, **CAP**, is constructed. Each cell (j,t) in **CAP** consists of a vector of size T which represents the priority of capacity usage among periods prior and including period t to satisfy the demand of family j in period t. We denote this vector, $\mathbf{cap}_{jt}$. For instance, if $\mathbf{cap}_{24}$ is equal to [3,1,4,2], then the demand of the second family in the fourth period is satisfied first in period 3 (inventory is held for one period) as much as the available capacity permits. Next, if all demand is not satisfied, the capacity of period 1 is used, etc. Thus, **CAP** is an $F \times T \times T$ cube. We also define a second matrix **DEM** of size $F \times T$, where each column t, denoted as $\mathbf{dem}_t$ is a vector of size F. $\mathbf{dem}_t$ represents the priority among different families for satisfying their demands in period t. If, for instance, $\mathbf{dem}_5$ = [3,2,1,4], then first the demand (of the fifth period) of the third family is satisfied using the corresponding capacity priority vector $\mathbf{cap}_{35}$.

Hence, once a solution is encoded by the matrices **DEM** and **CAP**, it is very easy to generate the corresponding feasible solution heuristically. The pseudocode for generating a schedule for a given encoding is given in Table 12. Starting from the first period, each family's demand is satisfied in order of the vector $\mathbf{dem}_t$. Once selected, each family's demand is satisfied by the available

Table 10. Aggregate solutions and disaggregated solution costs resulting from the IHPP approach

X_1^*	X_2^*	X_3^*	X_4^*	X_5^*	X_6^*	*Cost resulting from IHPP (disagg. sln.)*	C_t^l
1488	1103	462.18	1037.40	1037.40	924.00	78 773	2801
1488	1103	856.18	856.18	856.18	892.44	74 149	2311
1488	1103	788.70	788.70	959.50	924.00	73 407	2129
1488	1103	767.30	767.30	1001.80	924.00	73 760	2072

Table 11. Pseudocode for procedure IAOP

```
Procedure IAOP;
begin
  $C'_t = C_t$;
  $MinCost = \infty$;
  Solve Model P2 with $C_t$;
  repeat
    Read $Y^*_{jt}$, $O^*_t$ for $t = 1..T$;
    Check Capacity Feasibility of the solution including
    setup times;
    Calculate $Cost$, $RCD$;
    If ($Cost < MinCost$) and (Capacity Feasible) then
    $MinCost = Cost$;
    $C'_t = C'_t - RCD$;
    Solve Model P2 with $C'_t$;
  until (Model P2 Infeasible) or (Solution Violates
  Capacity) or ($Cost > MinCost$)
end;
```

regular and overtime capacities in the periods prior and including the current period in order of the vector $\mathbf{cap}_{jt}$. The capacity of each period is exhausted before using the capacity of the next period in vector $\mathbf{cap}_{jt}$. In Table 13 we demonstrate the mechanics of Generate Plan on an arbitrary partial chromosome for the illustrative example, i.e., the vectors $\mathbf{cap}_{j2}$ and $\mathbf{dem}_2$ are given so that the lot sizes for the second period are determined. In the first period all family lot sizes are inevitably equal to their demands and therefore, all regular time capacity of the first period is depleted and 1099 hours of overtime capacity remains. The partial chromosome has the following genes: $\mathbf{dem}_2$ is [2,3,4,1] and $\mathbf{cap}_{22}$, $\mathbf{cap}_{32}$, $\mathbf{cap}_{42}$, $\mathbf{cap}_{12}$ are [2,1], [1,2], [2,1], [1,2], respectively. Table 13 shows how the procedure Generate Plan is activated based on the specified vectors. The second family has the highest priority and $\mathbf{cap}_{22}$ prefers to use the second period's capacity. So, all demand of the second family is satisfied using the regular time capacity of the second period. Next, since the third family prefers to consume the first period's capacity, the overtime capacity of the latter period is consumed first without incurring a setup time, because that family is already assigned a lot size in that period. The remaining demand is satisfied by consuming the regular time capacity of the second period, now causing a setup time. Note that the first family consumes capacity of the second period, since its best preferred period (period 1) has its capacity totally exhausted by higher priority families in $\mathbf{dem}_2$. The minimum cost found by the GA is 68 257 for this problem.

Table 12. The procedure Generate Plan used to develop a production schedule for a given chromosome

```
Procedure Generate Plan;
begin
  Infeasible = false;
  for $t = 1$ to $T$ do
  begin
    $demset = \{1..F\}$;
    repeat
      select highest priority family $j^*$ from $\mathbf{dem}_t$ such that
      $j^* \varepsilon \{demset\}$;
      $demset = demset - \{j^*\}$;
      $capset = \{1..T\}$;
      repeat
        select highest priority period $t^*$ from $\mathbf{cap}_{j^*t}$ such that
        $t^* \varepsilon \{capset\}$;
        $capset = capset - \{t^*\}$;
        satisfy $d_{j^*t}$ as much as possible from $C_{t^*}$, then
        from $CO_{t^*}$;
        Reduce $d_{j^*t}$, $C_{t^*}$, and $CO_{t^*}$;
      until ($capset = \emptyset$) or ($d_{j^*t} = 0$);
      if $d_{j^*t} > 0$ then Infeasible = true;
    until ($demset = \emptyset$) or (Infeasible);
  end;
end;
```

Since a chromosome is represented by priority vectors as described above, the crossover and mutation operators are easy to apply because there are no feasibility problems in case two parent priority matrices are crossed over. As Storer *et al.* (1992) indicate, it is quite practical to represent a solution by priorities which can lead to a heuristic solution. We apply a two-point crossover called Linear Order Crossover described in Falkenauer and Bouffouix (1991) for sequencing problems because no elements are to be repeated in a priority vector and the partial priority sequence has to be preserved in order to intensify the search towards the optimal solution. Namely, given two parent $\mathbf{dem}_t$ vectors [3,2,4,1,5] and [1,4,5,2,3] and two crossover loci, 2 and 4, the offspring would be [3,4,5,2,1] and [5,2,4,1,3]. All vectors in **CAP** and **DEM** are crossed over in the same manner.

Table 13. The application of Generate Plan based on a partial encoding

Lot size assigned to family	*Lot size in period 1*	*Lot size in period 2*	*Capacity consumed from period*	*Capacity type*	*Capacity remaining*
Family 2	171.00	183.00	2	regular	2261
Family 3	200 + 274.75	69.25	1	overtime	0
			2	regular	1789
Family 4	636.00	297.00	2	regular	1075
Family 1	481.00	279.00	2	regular	90

The crossover probability of a chromosome depends on both the deviation of the chromosomes' cost value from the maximum cost in the current population and on the populations' cost range. A similar adaptive crossover probability is described by Srinivas and Patnaik (1994) who report that it works well in the optimization of multi-modal functions. A chromosomes' crossover probability keeps changing according to the convergence of the populations' cost range and increases when the populations' cost range tends to get stuck at a local optimum. Crossover is executed to result in the complete replacement of the old population.

Mathematically, the probability of crossover pc_k can be defined as:

$$pc_k = \begin{cases} (max - cost_k)/(max - min) & \text{if } \min \neq \max, \\ 1.0 & \text{otherwise.} \end{cases}$$

where *max* and *min* are respectively the maximum and minimum cost values in the current population, and $cost_k$ is the cost value of the *k*th chromosome found by the procedure Generate Plan.

The mutation probability is the same as the crossover probability and mutation is applied to a single pair of genes in $\mathbf{dem}_t$ and $\mathbf{cap}_{jt}$ and a single vector is changed in the matrices **DEM** and **CAP** respectively. Namely, a pair of elements are selected randomly in either of the vectors and swapped. After crossover and mutation the offspring/the mutated chromosome are/is evaluated by the procedure Generate Plan.

2.4. *A simulated annealing approach*

The encoding of the GA is indirect (Portmann, 1996) in the sense that a solution to a chromosome is obtained by the constructive algorithm Generate Plan. Although indirect encodings have been previously applied successfully in different contexts (Özdamar, 1999; Uckun *et al*., 1993), here, the GA requires a heavy data structure which limits important GA parameters such as the population size and may take considerable computation time. Consequently, we propose a Simulated Annealing (SA) approach (Kirkpatrick *et al*., 1983) which attacks the problem directly without ignoring binary setup variables. Since we would like to start the search from a good solution to the original problem, the initial solution is obtained from the procedure IAOP. Then, the solution is perturbed for a prespecified number of times. The perturbation scheme is simple.

Table 14 consists of the pseudocode for procedure Perturb which searches for better solutions while going from one neighboring solution to the next. A neighboring solution is one where exactly one family has its lot size changed by an amount "*DeltaLot*" in two consecutive periods. *DeltaLot* is bounded from above by capacity limits, current inventory levels and lot sizes in the selected pair of periods.

Table 14. Pseudocode for Perturb

```
Procedure Perturb;
{$C'_t$, $O'_t$ : remaining regular time/overtime capacity;
$ubLot$: upper bound on transferrable lot size;
$DeltaLot$: transferred lot size}
begin
  Select $t1$, $j^*$, increase/decrease decision;
  if $t1 = T$, then $t2 = T - 1$
  else $t2 = t1 + 1$;
  if (decrease) and ($I_{j^*,t1} > 0$) then
    begin
      if $t1 < T$ then $ubLot = \min\{(C'_{t2} + O'_{t2})/p_{j^*}, y_{j^*t1}, I_{j^*t1}\}$
        else $ubLot = \min\{(C'_{t2} + O'_{t2})/p_{j^*}, y_{j^*t1}\}$;
        ($s_{j^*}$ subtracted from $C'_{t2} + O'_{t2}$ if $y_{j^*t2} = 0$)
      $DeltaLot$ = random (0, $ubLot$);
      Calculate $CostDif$;
      If $CostDif > 0$ then Calculate $PA$, $tSA$
        else $tSA = 1$;
      If (($CostDif \geq 0$) and (Accept)) or ($CostDif < 0$) then
        begin
          Transfer $DeltaLot$ from period $t1$ to period $t2$;
          {$y_{j^*t1} = y_{j^*t1} - DeltaLot$; $y_{j^*t2} = y_{j^*t2} + DeltaLot$;
          Adjust $C'_{t1}$, $C'_{t2}$, $O'_{t1}$, $O'_{t2}$, $I_{j^*t1}$, $I_{j^*t2}$};
        $Cost = Cost + CostDif$;
        If $Cost < MinCost$ then $MinCost = Cost$;
        end;
    end;
  else if (increase) and ($C'_{t1} + O'_{t1} > 0$) then
    begin
      if $t1 = T$ then $ubLot = \min\{(C'_{t1} + O'_{t1})/p_{j^*}, y_{j^*t2}, I_{j^*t2}\}$
        else $ubLot = \min\{(C'_{t1} + O'_{t1})/p_{j^*}, y_{j^*t2}\}$;
        ($s_{j^*}$ subtracted from $C'_{t1} + O'_{t1}$ if $y_{j^*t1} = 0$)
      $DeltaLot$ = random (0, $ubLot$)
      Calculate $CostDif$;
      If $CostDif > 0$ then Calculate $PA$, $tSA$
        else $tSA = 1$;
      If (($CostDif \geq 0$) and (Accept)) or ($CostDif < 0$) then
        begin
          Transfer $DeltaLot$ from period $t2$ to period $t1$;
          {$y_{j^*t1} = y_{j^*t1} + DeltaLot$; $y_{j^*t2} = y_{j^*t2} - DeltaLot$;
          Adjust $C'_{t1}$, $C'_{t2}$, $O'_{t1}$, $O'_{t2}$, $I_{j^*t1}$, $I_{j^*t2}$};
        $Cost = Cost + CostDif$;
        If $Cost < MinCost$ then $MinCost = Cost$;
      end;
    end;
end;
```

The procedure Perturb selects a period ($t1$) and a family (j^*) randomly to change the lot sizes in two consecutive periods, $t1$ and $t1 + 1$. It decides randomly whether the lot size of the family is going to be increased or decreased in period $t1$. Both operations are carried out by considering the lot sizes and inventory levels of the family under consideration in the selected consecutive periods. Capacity limits are always monitored while shifting a family lot size. Thus, an iteration of Perturb consists simply of transferring some random and feasible portion of a lot size from one period to the next. It is usually possible to save on inventory and/or overtime costs during an iteration when for instance, a capacity bottleneck is released by moving a lot to another period and/or reducing inventory costs by transferring a lot to the next (may be nonbottleneck) period. In case a deteriorated cost results from an iteration, a probability of acceptance, *PA*, is calculated. *PA* depends on the amount of deterioration and on the current temperature, *tSA*, which in turn depends on the number of times a deteriorated cost has been obtained consecutively. If a randomly generated number between zero and one turns out to be less than *PA*, then the deteriorating iteration is carried out. *PA* is calculated as follows:

$$PA(CostDif, tSA) = \exp(-CostDif/(MinCost \times tSA))$$

where *CostDif* is the cost difference implied by the iteration and *MinCost* is the minimum cost obtained up to the current iteration. The temperature reduction scheme is continuous (Dowsland, 1993). Namely, after each non-improving move, the temperature, *tSA*, is reduced as follows

$$tSA \leftarrow tSA/(1 + \beta tSA),$$

where β is a nonnegative constant less than one. Initially *tSA* is equal to one. Whenever an improving move is made, *tSA* is reset to one. For achieving good results with the SA approach, β can be determined by exercising a parametric analysis or it may be adjusted dynamically during the search (Bozyel and Özdamar, 1996). In the case where β is dynamic, if the last m consecutive moves have resulted in improving cost values, then β is decreased by a *Stepsize* of the user's choice (a convenient stepsize is 0.01) less than one. On the other hand, if the last m consecutive moves have resulted in non-improving cost values, then β is increased by *Stepsize*. The dynamic temperature reduction scheme serves to update *PA* according to the specific part of the feasible region to which the search is heading. If the search is in a favorable direction, then β is decreased to encourage non-improving moves so that local optima can be skipped. Non-improving moves are discouraged when the search is visiting an unfavorable region of the feasible region, so that the search can escape from a region that is unlikely to store the global optimum solution.

Let us describe an iteration of Perturb on the small illustrative example. Table 15 demonstrates how Perturb carries out its alterations. $t1 = 2$ and $t2 = 3, j^* = 2$, the decision is to increase the lot size of the second family in period 2, and the incumbent solution has a cost of 70 311. Perturb transfers the whole lot size of 29 from the third period to the second. Both periods have overtime. In period 3 ($183 + 29 \times 2 = 241$) overtime hours are saved. However 58 hours of overtime are added to period 2. An inventory of 29 units is added at a cost of $19 \times 29 = 551$. So, net savings are $(241 - 58) \times 17 - 551 = 2560$. The SA approach eventually finds a solution with a cost of 60 480 for this problem.

3. Computational results

The solution procedures described above are tested on 198 randomly generated test instances. The first 90 problems consist of four families to be scheduled on a planning horizon of six periods. This set of small problems are generated to compare results against optimal solutions. The problems are generated according to a full factorial design based on three major problem characteristics: the ratio of average unit holding cost (over all families) to average unit overtime cost (*H/O*), the standard deviation of demand (*STD*) and the tightness of regular and overtime capacity (*CS*). The average overtime cost/unit is estimated as *ov* × *average process time*, where the average process time is the mean of family process times. The factor *H/O* is held at three levels (0.7, 1.1, 1.5) and holding cost per unit per period is randomly assigned a value between one and 20; family process times and setup times are uniformly distributed between [1–5] and [50–200], respectively; the demand is generated from a normal distribution with a mean of 100 units per period with a standard deviation *STD* held at two levels (10, 50); furthermore, demand has a seasonal trend over the planning horizon and not all families' demand patterns need to match; capacity is calculated such that each problem has a feasible solution, i.e., sufficient total capacity (regular time plus overtime) is made available to satisfy the cumulative demand at every period over the planning horizon; an allowance percentage of 5% of demanded capacity is also added to let a gap for setup times, and then the calculated average per period regular and overtime capacities (which are kept constant over time for simplicity) are multiplied with a scalar k to result in the three levels of the factor *CS*: ($k = 1.1, 1.3, 1.5$). That is, capacity is tightest at the *CS* level of 1.1. Total capacity is divided equally into regular time and overtime capacities.

Thus, $3 \times 2 \times 3 = 18$ factor combinations result. For each combination we generate five instances to result in

Table 15. An iteration of Perturb on a solution for the example

	$y_{j^*,2}$ $(I_{j^*,2})$	$y_{j^*,3}$ $(I_{j^*,3})$	C'_2	C'_3	*DeltaLot*	*CostDif*
Before Perturb	183 (0)	29 (0)	3819	3248		
After Perturb	212 (29)	0 (0)	3877	3007	29	−2560

the first set of 90 problems. The second set of 54 problems are generated according to the same design but they consist of 10 families and 10 periods instead. For each factor combination, three test instances are generated with the same set of parameters used in generating the first 90 problems. The third set of 54 problems are also generated similarly, but have 15 families and 10 periods. The first 90 problems are solved to optimality by the optimization package GAMS, version 2.25 (option XA), but the second and third sets of problems cannot be solved optimally, so we compare all results (including GAMS results obtained within a given iteration limit) with the best solution found.

In Tables 16–18 the computational results for the small size 90 problems, the second and third sets of 54 test problems are provided, respectively. The results are represented by the average percentage deviation (standard deviation) of the solutions from the optimum (best solution), the number of optimal (best) solutions found and the average CPU time per problem on a Pentium 75 MHz PC. An exception is made for GAMS which has been run on a Pentium 200 MHz server PC to obtain solutions within reasonable computation times. We allowed 100 000 iterations to be performed in the GAMS solution and the tolerance level (the integrality gap in percentage between the best lower bound and the best integer solution found at the end of GAMS iterations) is set to 0.1%. Although GAMS was run on a much faster computer, the computation times are excessive. The average integrality gap achieved at the end of the GAMS iterations is calculated to be 47.60% for the 10 family problems and 49.70% for the 15 family problems. The relative performance of GAMS deteriorates with increasing problem size.

In Tables 16–18 the number of problems in which a heuristic concludes in an infeasible solution is also indicated. In small problems the nearest the HPP approach can get to the optimum on the average is 14.59% (with IHPP). As the number of families/periods increase, the performance of the HPP approaches deteriorates. In all HPP variants *SN* is set to 100. The HPP approach with FP, which solves the aggregate and family disaggregation models proposed by Hax and Candea (1984), is the poorest performer among all HPP approaches. As expected HPP + MFP performs relatively better than HPP + FP and IHPP is the best empirically since it tries to avoid the disadvantage of the approximation made in the aggregate model.

Surprisingly the simple algorithm IAOP works as well as the GA which was run with a population size of 80 and a maximum generation limit of 1500 for small problems. Unfortunately, although the Generate Plan procedure is quite fast, GA takes a lot of computation time since the number of crossover operations are high. Due to the heavy data structure and excessive memory requirements, GA could not be tested in larger test instances.

Table 16. Results for the 90 small size test problems

Solution procedure	*HPP + FP*	*HPP + MFP*	*IHPP*	*IAOP*	*GA*	*SA (IAOP start. solution)*	*SA-I (random start. solution)*	*SA-II (random start. solution)*	*GAMS*
Percent deviation from the optimum (%) (standard deviation)	22.74 (17.67)	18.18 (15.78)	14.59 (13.43)	5.46 (5.43)	5.62 (10.08)	1.08 (4.02)	4.37 (6.34)	0.24 (0.48)	0
Number of optimal solutions	3	3	7	15	25	39	34	86	90
Number of problems in which no feasible solution is achieved	10	9	7	–	–	–	–	–	–
Total number of iterations	1	1	2–3	3–4 LP solution	1500 × 80	15×10^4	15×10^4	15×10^4	10^5
Average CPU seconds/problem	0	0	2	2	420	1	1	2	1.5*

* CPU time on a Pentium 200 MHz PC.

Table 17. Results for 54 Problems with 10 families/10 periods

Solution procedure	*HPP + FP*	*HPP + MFP*	*IHPP*	*IAOP*	*SA (IAOP start. solution)*	*SA-I (random start. solution)*	*SA-II (random start. solution)*	*GAMS*
Percent deviation from the best solution (%) (standard deviation)	49.5 (34.9)	38.6 (28.2)	33.1 (26.9)	6.8 (7.5)	2.0 (2.4)	5.3 (4.3)	1.8 (2.5)	1.5 (2.2)
Number of best solutions	0	0	0	4	15	2	20	24
Number of problems in which no feasible solution is achieved	5	6	4	3	0	0	0	0
Total number of iterations	1	1	3–4	5–6 LP solution	10×10^5	10×10^5	10×10^5	10^5
Average CPU seconds/problem	0	0	3	4	5	5	6	1926*

* CPU time on a Pentium 200 MHz PC.

Table 18. Results for 54 Problems with 15 families/10 periods

Solution procedure	*HPP + FP*	*HPP + MFP*	*IHPP*	*IAOP*	*SA (IAOP start. solution)*	*SA-I (random start. solution)*	*SA-II (random start. solution)*	*GAMS*
Percent deviation from the best solution (%) (standard deviation)	54.4 (39.9)	37.9 (25.1)	31.4 (2.29)	4.5 (2.6)	1.6 (1.7)	5.4 (6.9)	2.0 (2.0)	2.4 (9.3)
Number of best solutions	0	0	0	0	19	4	15	22
Number of problems in which no feasible solution is achieved	3	4	3	3	–	–	–	–
Total number of iterations	1	1	4–5	5–6 LP solution	10×10^5	10×10^5	10×10^5	10^5
Average CPU seconds/problem	4	3	4	4	5	5	6	3743*

* CPU time on a Pentium 200 MHz PC.

Significantly better results are obtained with SA which starts with the IAOP solution. In case the IAOP solution is infeasible, SA randomly generates its own starting solution by randomly assigning lot sizes to families in each period within capacity limits. If a capacity bottleneck occurs, then lot sizes are added to the ones allocated in previous periods and inventory is thus accumulated. In order to show the effect of starting with a good solution, we also provide results obtained with SA using randomly generated starting solutions under the heading SA-I. In both SA and SA-I, the parameter β has been selected as 0.01 after a preliminary parametric analysis was carried out for the seemingly difficult problems. We also test the effects of using a dynamic parameter β which changes during the search using a stepsize of 0.01. The related results are provided under SA-II. All three variants of SA are permitted to carry out 10 000 iterations and they are re-run 15 times with a new random seed in small problems, and in larger problems, the iteration limit and the number of re-runs are set to 100 000 and 10, respectively. Best solutions are reported. Note that SA-II's performance is considerably better than that of SA-I in smaller problems, but in larger problems the effect of the dynamic β is lost. In larger problems, results become rather insensitive to other values of β when SA is used with a fixed β value. This effect might be due to the increased number of iterations allowed and SA converges to the final solution independent of β. The computation time for SA is considerably lower than the GA's because an iteration in SA consists of only a few basic mathematical operations. The SA approach performs best among all tested approaches with respect to the solution quality and computational efficiency and its memory requirements are minimal. The relatively superior performance of SA is due to its straightforward treatment of the CLSP in the sense that the variables in the original model P are dealt with during the moves which are executed within the limits of the given constraints. Also, a neighboring solution is very close to the current one, because the lot sizes of a family are perturbed in consecutive periods rather than distant ones. Therefore, SA is less liable to skip the optimum solution if the current solution is already near it. On the other hand, HPP based heuristics result in inferior solutions, because they inevitably lead to sub-optimality due to the aggregation of families. The sub-optimality of solutions increases as the number of families increase. Hence, the IAOP method which treats the problem at the family level while relaxing binary setup variables is significantly superior to HPP-based heuristics. When the two meta-heuristics GA and SA are compared we observe that in the GA, both the number of mathematical operations and the memory requirements are excessively high. Consequently, the performance of GA is inferior to SA and anyway the GA is hard to use with larger size problems.

As to the effect of the factorial design on heuristic performance, ANOVA results show that only two main factors (CS and H/O) are statistically significant. However, the effect of CS is much more significant than H/O and all heuristics except SA(IAOP) and IAOP deteriorate in performance when CS is at its high level. This result is due to the natural fact that the set of feasible solutions becomes larger when capacity is abundant. When capacity is tight it is more difficult to find an integer optimal solution and this fact is consistent with the results of the mathematical programming based heuristic provided by Trigeiro *et al.* (1989) for the CLSP with setup times. For example, for larger problems of 15 families GAMS could solve a problem with $CS = 1.5$ within 5 minutes on the Pentium 200 MHz computer and resulted in an integrality gap of 0.1%, but it converged to an integrality gap of 63% for a similar problem with $CS = 1.1$ within 41 minutes on the same computer. The reason why SA is effective in solving tight capacity problems is that it starts from a feasible solution and remains in the feasible region which is comparatively narrow.

The next statistically significant factor is H/O, but its effect is much less detectable than CS. As H/O increases,

Table 19. Results for different values of CS and H/O factors (54 problems with 15 families/10 periods)

		HPP + FP	*HPP + MFP*	*IHPP*	*IAOP*	*SA(IAOP)*	*SA-I*	*SA-II*	*GAMS*
$CS = 1.1$	avg	0.24	0.14	0.13	0.05	0.00	0.02	0.01	0.02
	std	0.22	0.08	0.07	0.02	0.01	0.02	0.01	0.02
$CS = 1.3$	avg	0.46	0.34	0.29	0.05	0.01	0.03	0.02	0.01
	std	0.21	0.18	0.18	0.03	0.02	0.04	0.01	0.01
$CS = 1.5$	avg	0.88	0.60	0.49	0.04	0.03	0.11	0.03	0.04
	std	0.43	0.23	0.24	0.03	0.02	0.09	0.02	0.16
$H/O = 0.7$	avg	0.48	0.35	0.27	0.06	0.02	0.06	0.02	0.01
	std	0.30	0.19	0.16	0.05	0.02	0.04	0.02	0.02
$H/O = 1.1$	avg	0.54	0.42	0.35	0.05	0.02	0.05	0.02	0.01
	std	0.37	0.30	0.29	0.04	0.02	0.05	0.02	0.02
$H/O = 1.5$	avg	0.47	0.40	0.38	0.09	0.02	0.05	0.02	0.02
	std	0.40	0.36	0.34	0.11	0.03	0.04	0.03	0.03

the selection among different families becomes more important for accumulating inventories. The fluctuation of demand, *STD*, does not play an important role in heuristic performance, because all heuristics take action according to future demand forecasts anyway. The latter result is again consistent with the conclusions drawn by Trigeiro *et al.* (1989). In Table 19 the performance of heuristics are provided in terms of average percentage deviation (standard deviation) from the best solution given different values of *CS* and *H*/*O* in the largest size problems with 15 families and 10 periods.

4. Conclusion

The capacitated lot sizing problem with overtime decisions and setup times is relevant in many practical situations where family lot sizes need to be determined in a bottleneck department over a planning horizon. In many cases, setups involve no additional costs apart from extra manhours which incur costs if capacity is tight and overtime takes place.

Since the problem considered is NP-hard, there is a need to develop heuristic approaches to solve it. In computational results we demonstrate that the classical Hierarchical Production Planning approach provides considerably suboptimal solutions to the problem. Consequently, we develop an iterative approach which omits binary variables from the capacity constraints in the problem, a Genetic Algorithm (GA) whose encoding is based on the well-known linear transportation-like formulation of the capacitated single-item production planning problem, and a Simulated Annealing (SA) approach based on shifting lot sizes partially or wholly among consecutive time periods. Among all heuristics, the iterative approach and SA are the procedures providing best solutions. The GA's good performance is restricted to problems of small size due to the heavy data structure of the encoding.

References

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

Bowman, E.H. (1956) Production scheduling by the transportation method of linear programming. *Operations Research*, **3**, 100–103.

Bozyel, M.A. and Özdamar, L. (1996) A heuristic approach to the capacitated lot sizing and scheduling problem (with parallel facilities). Working Paper, Department of Systems Engineering, Yeditepe University, Istanbul.

Catrysse, D., Maes, J. and van Wassenhove, L.N. (1990) Set partitioning and column generation heuristics for capacitated lotsizing. *European Journal of Operational Research*, **46**, 38–47.

Dixon, P.S., Elder, M.D., Rand, G.K. and Silver, E.A. (1983) A heuristic algorithm for determining lot sizes of an item subject to regular and overtime production capacities. *Journal of Operations Management*, **3**, 121–130.

Dixon, P.S. and Silver, E.A. (1981) A heuristic solution procedure for the multi-item single level limited capacity lot sizing problem. *Journal of Operations Management*, **2**, 23–39.

Doğramacı, A., Panayiotopoulos, J.C. and Adam, N.R. (1981) The dynamic lot sizing problem for multiple items under limited capacity. *IIE Transactions*, **13**, 294–303.

Dowsland, K.A. (1993) Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, **68**, 389–399.

Falkenauer, E. and Bouffoix, S. (1991) A genetic algorithm for job-shop, in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA.

Garey, M. and Johnson, D. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Co., San Francisco, CA.

Günther, H.O. (1987) Planning lot sizes and capacity requirements in a single stage production system. *European Journal of Operational Research*, **31**, 223–231.

Hax, A.C. and Candea, D. (1984) *Production and Inventory Management*, Prentice-Hall Englewood Cliffs, NJ.

Kim, D. and Mabert, V.A. (1995) Integrative versus separate cycle scheduling heuristics for capacitated discrete lot sizing and sequencing problems. *International Journal of Production Research*, **33**, 2007–2021.

Kırca, Ö. (1990) An efficient algorithm for the capacitated single item dynamic lot size problem. *European Journal of Operational Research*, **45**, 15–24.

Kırca, Ö. and Kökten, M. (1994) A new heuristic approach for the multi-item dynamic lot sizing problem. *European Journal of Operational Research*, **75**, 332–341.

Kirkpatrick, A., Gelatt, Jr., C.D. and Vechi, M.P. (1983) Optimization by simulated annealing. *Science*, **220**, 671–680.

Lambrecht, M.R. and Vanderveken, H. (1979) Heuristic procedure for the single operation multi-item loading problem. *IIE Transactions*, **11**, 319–326.

Maes, J. and van Wassenhove, L.N. (1985) Multi-item single level capacitated dynamic lotsizing heuristics: a computational comparison, part II: rolling horizon. Working Paper 85-08, Katholieke Universiteit Leuven.

Maes, J. and van Wassenhove, L.N. (1986) A simple heuristic for the multi-item single level capacitated lot sizing problem. *Operations Research Letters*, **4**, 265–273.

Michalewicz, Z. (1994) *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, Berlin.

Özdamar, L. (1999) A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions on Systems, Man and Cybernetics*, **29**, 44–69.

Özdamar, L., Atlı, A.Ö. and Bozyel, M.A. (1996) Heuristic family disaggregation techniques in hierarchical production planning systems. *International Journal of Production Research*, **34**, 2613–2628.

Özdamar, L., Bozyel, M.A. and Birbil, İ. (1998) A hierarchical decision support system for production planning (with case study). *European Journal of Operations Research*, **104**, 403–422.

Özdamar, L. and Yazgaç, T. (1997) Capacity driven due date settings in make-to-order production systems. *International Journal of Production Economics*, **49**, 29–44.

Portmann, M.C. (1996) Genetic algorithms and scheduling: a state of the art and some propositions, in *Proceedings of the Workshop on Production Planning and Control*, FUCAM, Mons, Belgium, pp. 1–24.

Srinivas, M. and Patnaik, L.M. (1994) Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, **24**, 656–667.

Storer, R.H., Wu, S.D. and Vaccari, R. (1992) New search spaces for sequencing problems with application to job-shop scheduling. *Management Science* **38**, 1495–1509.

Thizy, J.M. and van Wassenhove, L.N. (1985) Lagrangean relaxation the multi-item capacitated lot sizing problem: a heuristic implementation. *IIE Transactions*, **17**, 308–313.

Trigeiro, W.W. (1987) A dual cost heuristic for the capacitated lot sizing problem. *IIE Transactions*, **19**, 67–72.

Trigeiro, W.W., Thomas, L.J. and McLain, J.O. (1989) Capacitated lot sizing with setup times. *Management Science*, **35**, 353–366.

Uckun, S., Bagchi, S., Kawamura, K. and Miyabe, Y. (1993) Managing genetic search in job-shop scheduling. *IEEE Expert*, **8**, 15–24.

Biographies

Dr. Linet Özdamar is currently with the Industrial Engineering Department at Doğus University, Istanbul, Turkey. Dr. Özdamar acquired her B.Sc., M.Sc. and Ph.D. degrees from Boğaziçi University, Istanbul. Her past and present research interests lie in the fields of project scheduling and management, lot sizing models, hierarchical production planning, and solution methodologies based on metaheuristics. More recent fields of interest are the development of global optimization and function approximation methods using fuzzy logic. Dr. Özdamar has published related articles in journals such as *International Journal in Production Research*, *European Journal of Operations Research*, *IEEE Transactions on Systems, Man and Cybernetics*, *IEEE Transactions on Fuzzy Systems* and *Journal of Global Optimization*.

Mehmet Ali Bozyel gained B.Sc. and M.Sc. Degrees from the Department of Industrial Engineering, Marmara University, Istanbul. He currently runs a medium-size company manufacturing industrial air conditioning machinery and is responsible for the production planning tasks. Mr. Bozyel has published related work in *International Journal in Production Research*, *IIE Transactions* and *European Journal of Operations Research*.