European Journal of Operational Research 203 (2010) 583–592

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

Efficient and effective heuristics for the coordinated capacitated lot-size problem

Arunachalam Narayanan [a,*], Powell Robinson [b,1]

[a] Department of Engineering Technology and Industrial Distribution, Texas A&M University, College Station, TX 77843-3367, United States
[b] Department of Information and Operations Management, Mays Business School, Texas A&M University, College Station, TX 77843-4217, United States

ARTICLE INFO

Article history:
Received 31 July 2008
Accepted 26 August 2009
Available online 1 September 2009

Keywords:
Joint replenishment
Coordinated replenishment
Lot sizing
Heuristics

ABSTRACT

Coordinating procurement decisions for a family of products that share a constrained resource, such as an ocean shipping container, is an important managerial problem. However due to the problem's difficult mathematical properties, efficient and effective solution procedures for the problem have eluded researchers. This paper proposes two heuristics, for the capacitated, coordinated dynamic demand lot-size problem with deterministic but time-varying demand. In addition to inventory holding costs, the problem assumes a joint setup cost each time any member of the product family is replenished and an individual item setup cost for each item type replenished. The objective is to meet all customer demand without backorders at minimum total cost. We propose a six-phase heuristic (SPH) and a simulated annealing meta-heuristic (SAM). The SPH begins by assuming that each customer demand is met by a unique replenishment and then it seeks to iteratively maximize the net savings associated with order consolidation. Using SPH to find a starting solution, the SAM orchestrates escaping local solutions and exploring other areas of the solution state space that are randomly generated in an annealing search process. The results of extensive computational experiments document the effectiveness and efficiency of the heuristics. Over a wide range of problem parameter values, the SPH and SAM find solutions with an average optimality gap of 1.53% and 0.47% in an average time of 0.023 CPU seconds and 0.32 CPU seconds, respectively. The heuristics are strong candidates for application as stand alone solvers or as an upper bounding procedure within an optimization based algorithm. The procedures are currently being tested as a solver in the procurement software suite of a nationally recognized procurement software provider.

1. Introduction

The coordinated, capacitated lot-size problem (CCLSP) determines the time-phased replenishment schedule that minimizes the sum of ordering and inventory costs subject to capacity constraints. A family setup cost is incurred each time one or more items in the product family are replenished, and a minor setup cost is charged for each item replenished. Item demand, which must be met without backorders, is dynamic but deterministic over the planning horizon. As noted by Silver (1979), Shapiro et al. (2002) and Robinson and Lawrence (2004), coordinated lot-size problems are often encountered when managing manufacturing, transportation and procurement processes. The need to coordinate the procurement decisions for a family of products that share a constrained transportation resource, such as an ocean shipping container, motivates this particular research.

Due to their importance in industry and mathematical complexity, coordinated lot-size problems are frequently studied in the supply chain management literature. However, while effective heuristic and exact algorithms exist for the uncapacitated problem variant, the more mathematically challenging capacitated problem remains virtually unsolved. Only Erenguc and Mercan (1990), Robinson and Lawrence (2004) and Federgruen et al. (2007) propose algorithms and provide computational results for solving the CCLSP. In each case, the authors experience computational difficulty finding optimal solutions and suggest that the literature will mainly develop in the direction of discovering effective and computationally feasible heuristic procedures. Federgruen et al. (2007) describe an effective progressive interval/expanding horizon heuristic, but solution time grows rapidly with problem size. Hence, further research in this area is well-justified.

This paper proposes a six-phase heuristic (SPH) and simulated annealing meta-heuristic (SAM) for the CCLSP. Conceptually the SPH is related to the heuristics in Dogramaci et al. (1981) for the multi-item capacitated lot-size problem (MCLSP) and Robinson et al. (2007) for the coordinated uncapacitated lot-size problem (CULSP). However, while the MCLSP assumes the replenishment

* Corresponding author. Tel.: +1 979 845 1462; fax: +1 979 845 4980.
E-mail addresses: chalam@tamu.edu (A. Narayanan), p-robinson@mays.tamu.edu (P. Robinson).
[1] Tel.: +1 979 845 1616; fax: +1 979 845 5653.

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.08.015

decision for each item is independent and capacity is available each time period, the CCLSP models a shared family setup cost where capacity is only available in time periods in which the setup cost is paid. This unique problem characteristic provides an additional layer of hierarchical setup decision variables to the MCLSP, which further complicates the problem's solution requiring more intricate procedures for savings calculations and ensuring capacity feasibility. In addition the inclusion of capacity constraints in CCLSP further complicates solution of the CULSP due to the need to allocate scarce capacity among competing items.

Over a wide range of parameter values, the SPH finds solutions with an average optimality gap of 1.53% in an average time of 0.023 CPU seconds. However, solution quality is sensitive to the time between orders (TBO). Embedding the SPH into a SAM procedure yields significantly improved solutions and mitigates the impact of TBO on solution quality. For a set of 1635 test problems, spanning all experimental designs reported in the literature, the SAM finds solutions with an average optimality gap of 0.47% in an average time of 0.32 CPU seconds. This improvement in solution quality comes at a very modest increase in computational resources. These results are encouraging and suggest the potential application of SPH and SAM as stand-alone solvers, lot-sizing procedures embedded within requirements based planning software, and as upper bounding procedures in optimization-based algorithms.

Section 2 provides a brief literature review. Section 3 provides the mixed-integer-programming formulation that is solved to establish performance benchmarks for the heuristic. The heuristic procedures are detailed in Section 4. Section 5 describes the experimental design and provides the results. Finally Section 6 gives the conclusion and implications of the research.

2. Literature survey

CCLSP is a generalization of the single-item uncapacitated lot-size problem (ULSP), the single-item capacitated lot-size problem (CLSP), the multiple-item capacitated lot-size problem (MCLSP), and the coordinated, uncapacitated lot-size problem (CULSP). A variety of taxonomies for classifying the general lot-sizing problem are proposed in Drexl and Kimms (1997), Karimi et al. (2003), Robinson and Lawrence (2004), Jans and Degraeve (2008) and Robinson et al. (2009). Belvaux and Wolsey (2000, 2001) and Pochet and Wolsey (2006) discuss modeling and solving capacitated lot-sizing problems as mixed-integer programs. Jans and Degraeve (2007) review meta-heuristic approaches for solving lot-sizing problems. Instead of duplicating these reviews, we summarize the literature most closely related to the research reported here.

In the CULSP, a family of items shares a setup cost and each unique item ordered incurs an item setup cost. The CULSP is shown to be *NP*-complete by Arkin et al. (1989) and Joneja (1990). Zangwill (1966), Veinott (1969), Kao (1979), Silver (1979) and Haseborg (1982) develop dynamic programming algorithms for which solution time grows exponentially in problem size. Erenguc (1988), Kirca (1995) and Robinson and Gao (1996) propose specialized branch and bound algorithms for the problem. Federgruen and Tzur (1994) describe a branch and bound technique whose lower bound is provided by a partitioning heuristic. Fogarty and Barringer (1987), Atkins and Iyogun (1988), Iyogun (1991), Silver and Kelle (1988), Boctor et al. (2004), and Robinson et al. (2007) describe dynamic programming, construction and meta heuristics for the CULSP.

The CCLSP contains both the shared capacity constraints across items that complicate solution of the MCLSP and the binary family setup decision variables that complicate the mathematical structure of the CULSP. Erenguc and Mercan (1990) consider multiple product families assuming that labor is a sunk cost and capacity is consumed by product family and item setup and production run time. Computational results for problems with up to eight items, 10 time periods, and four families are reported.

Robinson and Lawrence (2004) propose a Lagrangean heuristic for the single-product family CCLSP with backorders. Computational experiments provide heuristic solutions with average optimality gaps of 0.44%, 3.9%, and 4.72% at the 5%, 45% and 85% capacity utilization levels, respectively. Optimal solutions to 12-period problems with an 85% capacity utilization level and more than two items could not be found within 100 minutes CPU time by general-purpose optimization software. These findings highlight the difficulty of finding both good heuristic and optimal solutions for the CCLSP.

Federgruen et al. (2007) develop a strict partitioning (SP) and a progressive interval/expanding horizon heuristic (EHH) for the CCLSP. Their findings show that item and family time-between-orders and capacity utilization impact heuristic solution quality. The SP heuristics are computationally efficient, but have an average 14.7% optimality gap. The EHH heuristics provides solutions with an average optimality gap of 1.2%, but computational requirements increase rapidly with problem size. For example, a 10-item and 10-period problem requires 30 CPU seconds, while a 25-item and 10-period problem requires approximately 5.5 hours to solve.

3. Problem formulation

One challenge in verifying heuristic performance is to obtain optimal solutions to large-size problems for performance benchmarking. Toward this end, Robinson et al. (2009) evaluated the computational requirements associated with solving the CCLSP formulations in Gao and Robinson (2003), Robinson and Lawrence (2004) and Federgruen et al. (2007). The most effective formulation, and the one used for benchmarking purposes in this research, is the Gao and Robinson (2003) formulation. This is an extension of Robinson and Gao's (1996) formulation for the uncapacitated coordinated lot-sizing problem, which is based on the well known simple plant location formulation of the lot-sizing problem (Krarup and Bilde, 1977). An alternative to this formulation is the network formulation by Eppen and Martin (1987) for MCLSP.

Minimum cost replenishment policies are to be determined over a T period planning horizon for a single product family of items. Demand, d_{it}, is stated in capacity units for item $i = 1, 2, \ldots, I$ in period $t = 1, 2, \ldots, T$ is deterministic, may vary over time, and must be satisfied without backorders. Order lot-splitting is permissible. A family setup cost $S_{t'}, t' = 1, 2, \ldots, T$ is required each period any member of the product family is ordered. A minor setup cost $s_{it'}$ and a unit cost of $c_{it'}$ is associated with ordering item i in period t'. The per unit inventory holding cost for serving demand for item i in period t from production in period t' is $h_{it't}$. The total unit cost for supplying demand for item i in time t from a production in period t' is $C_{it't} = c_{it'} + h_{it't}$. Since we only consider a single product family, we can implicitly consider product family setup times by defining $P_{t'}$, the total available capacity for period t', as the design capacity minus the family setup time. Decision variable, $Z_{t'}$, equals 1 if the product family is setup in period t', and 0 otherwise. The decision variable, $Y_{it'}$, equals 1 if item i is ordered in period t', and 0 otherwise. $X_{it't}$ is the portion of demand for item i in period t that is served from an order in period t'. The mixed-integer programming formulation of CCLSP is

$$Z_p = Min \sum_{t'=1}^{T} S_{t'} Z_{t'} + \sum_{t'=1}^{T} \sum_{i=1}^{I} s_{it'} Y_{it'} + \sum_{t'=1}^{T} \sum_{i=1}^{I} \sum_{t=t'}^{T} C_{it't} X_{it't} d_{it} \qquad (1)$$

Subject to

$$\sum_{t'=1}^{t} X_{it't} = 1 \quad \forall i = 1,\ldots,I,\ t = 1,\ldots,T, \tag{2}$$

$$\sum_{i=1}^{I}\sum_{t=t'}^{T} d_{it}X_{it't} \leqslant P_{t'}Z_{t'} \quad \forall t' = 1,\ldots,T, \tag{3}$$

$$Y_{it'} \leqslant Z_{t'} \quad \forall i = 1,\ldots,I,\ t' = 1,\ldots,T, \tag{4}$$

$$X_{it't} \leqslant Y_{it'} \quad \forall i = 1,\ldots,I,\ t' = 1,\ldots,T,\ t = t',\ldots,T, \tag{5}$$

$$Z_{t'} = 0 \text{ or } 1 \quad \forall t' = 1,\ldots,T, \tag{6}$$

$$Y_{it'} = 0 \text{ or } 1 \quad \forall i = 1,\ldots,I,\ t' = 1,\ldots,T, \tag{7}$$

$$0 \leqslant X_{it't} \leqslant 1 \quad \forall i = 1,\ldots,I,\ t' = 1,\ldots,T, t = 1,\ldots,T. \tag{8}$$

Constraints (2) insure that each item's demand is satisfied in each period. Capacity constraints are represented in Eq. (3). Constraints (4) prevent an item setup from occurring unless the product family is setup, while constraints (5) prohibit ordering an item unless the item setup charge is incurred. Constraints (6)–(8) force decision variables to take on feasible solution values.

4. Experimental heuristics

4.1. Six phase heuristic (SPH)

The SPH considers the impact of the shared family setup on both cost and capacity. The shared family setup adds a hierarchical layer of binary decision variables to the MCLSP and the capacity constraints add a layer of complexity not found in the CULSP. SPH is the first published improvement heuristic for CCLSP. Improvement heuristics begin with a starting solution which may be infeasible and then iteratively adjusts the replenishment schedule seeking to restore feasibility and lower cost in a greedy manner. Examples of improvement heuristics for MCLSP include Dogramaci et al. (1981) and Karni and Roll (1982). SPH contains three major subroutines that are implemented in six-phases. A brief description of the subroutines and the six-phase approach follows. The details of the heuristics are provided in Appendix.

Subroutine I: Cost minimizing left-shift. The cost minimizing left-shift subroutine iteratively reschedules either individual items or the product family, whichever one yields the greatest net cost savings, into earlier production time periods. Left-shifting accepts higher inventory costs in return for lower setup costs. Only "open" time periods (i.e., a product family setup is scheduled) are considered to receive the rescheduled production. The entire product family cannot be left-shifted into an earlier time slot if "closing" the period (i.e., removing the product family setup from the production schedule) will result in insufficient aggregate production capacity over the production horizon to provide a feasible schedule. This subroutine guarantees aggregate capacity feasibility, but does not guarantee individual period capacity feasibility.

Subroutine II: Feasibility seeking left-shift. This subroutine reschedules production into earlier time periods as necessary to guarantee individual time period capacity feasibility, while minimizing the cost increase.

Subroutine III: Cost minimizing right-shift. This procedure attempts to reduce inventory and total schedule costs by right-shifting production as late as possible while still maintaining capacity feasibility and meeting demand due dates.

4.1.1. Six-phase heuristic

The six-phase heuristic begins by verifying that the problem is "aggregate" capacity feasible over the planning horizon without backordering. An initial production schedule is established with lot-sizes $a_{it} = d_{it}$ for all i and t, where d_{it} is the demand of item i in time period t stated in capacity units. Next, the heuristic is implemented in six-phases as follows:

Phase 1. Run Subroutine I to generate a lower cost production schedule if possible.
Phase 2. Run Subroutine II to insure that the production schedule is capacity feasible in each time period.
Phase 3. Since the schedule changes during Phase 2 may generate new opportunities for cost reduction, Phase 2 is followed by another application of Subroutine I.
Phase 4. Running Subroutine I in Phase 3 may destroy individual time period capacity feasibility. Hence, Subroutine II is invoked to guarantee feasibility.
Phase 5. Run Subroutine III in an attempt to decrease inventory costs by moving production as late as possible in the planning horizon while maintaining capacity feasibility.
Phase 6. Run Subroutine I to search for additional potential cost reductions. However in this final phase, only lot-size consolidations that do not violate individual time period capacity constraints are permitted. At the conclusion of Phase 6, the heuristic terminates with a capacity feasible solution.

The procedures identify feasible solutions at the end of Phase 2, 4, 5 and 6.

The SPH is an extension of the two phase heuristic (TPH) for the CULSP in Robinson et al. (2007), which considers both schedule and cost adjustments related to capacity restrictions. The SPH differs from the TPH in the following way. First, the TPH consists of two subroutines: cost minimizing left-shift (similar to subroutine I above) and cost minimizing right-shift (similar to subroutine III above). A feasibility seeking left-shift subroutine is not needed in the two phase approach, since capacity constraints are not considered in CULSP. Second, subroutine I for the SPH includes an aggregate capacity feasibility check prior to rescheduling production earlier in time. Third, subroutine III in the SPH must check individual period capacity before shifting production into later time periods. The final difference is the cost savings calculation in subroutine I where there is no need for either item or family savings adjustment (as described in the Appendix: step 2 of subroutine I) in the TPH since the left shift operation is not capacity constrained. The SPH is a generalization of the TPH.

4.2. Simulated annealing meta-heuristic (SAM)

A simulated annealing meta-heuristic mimics physical annealing processes to escape from local optima solutions by applying transition probabilities and Monte Carlo simulation to jump from a current solution to a neighboring candidate solution. The transition probability, Pr, is

$$Pr = Min\left[1, e^{(\widehat{C}-C')/\theta_n}\right], \tag{9}$$

where $\widehat{C}$ is the objective function value of the current solution, C' is the objective function value of the candidate solution, and θ_n is the current temperature. If $C' < \widehat{C}$, then $Pr = 1$ and the current solution is replaced with the candidate solution. If $C' \geqslant \widehat{C}$, the current solution is replaced by the candidate solution with probability Pr.

The meta-heuristic is initiated with a relatively high temperature, θ_0, which yields a high probability of accepting candidate solutions and escaping from local optima in the early stages of algorithm implementation. During the search, the temperature is systematically reduced according to a cooling schedule making it less likely to jump to an inferior candidate solution. We apply a homogeneous cooling schedule that maintains a constant

temperature for a specified number of iterations before being decreased. This enhances the probability of jumping to multiple new solution neighborhoods during each phase of the cooling cycle. The cooling schedule used in this experiment is,

$$\theta_n = \theta_{n-1} * 0.8, \tag{10}$$

where θ_{n-1} is the temperature from previous iteration. At each temperature level, the search for an improved solution is performed five times prior to cooling. The search stops when the $\theta = 1$ or when an improved solution is not found after λ successive iterations. In preliminary experiments we evaluated alternative values for λ and θ_o, with $\lambda = 3T$ and $\theta_o = 1000$ providing the best results.

The SAM uses the SPH to find a feasible starting solution and improve the randomly generated solutions in the search process. The steps of the SAM follow.

Step 1: Initialization. Set $n = 0$ and $\theta_o = 1000$. Solve the SPH for an initial problem solution. Set $\widehat{C}$ equal to the objective function value of the SPH solution. Set $C_B = \widehat{C}$ and the iteration counter, *count* = 1.

Step 2: Neighbor generation. Randomly choose a value of $t \in \{1, 2, \ldots, T\}$ and attempt to change solutions by perturbing the current solution as follows. If it is not possible to perturb the selected time period, then choose another t.

Case 1: If the current solution replenishes any items in period t, reschedule all the items in t into the immediately preceding open replenishment period that has sufficient aggregate capacity. This removes the family setup in period t.

Case 2: If the current solution does not have the product family scheduled in period t, schedule a family setup in t. Next, from the immediate preceding open replenishment period, reschedule any items that demanded in time period t or later into period t.

Step 3: Neighbor improvement. Attempt to improve the perturbed solution generated in Case 1 or 2 by applying the SPH, while maintaining the status of the perturbed family setup in period t. The resulting solution provides a new candidate solution with an objective function value C'.

Step 4: Neighbor search. Compute the transition probability *Pr* using Eq. (9). If *Pr* is greater than or equal to a randomly generated number between [0,1], replace the current solution $\widehat{C}$ with the candidate solution C'; otherwise reject the candidate solution. If $\widehat{C} < C_B$ update the best known solution and set $C_B = \widehat{C}$ and reset *count* = 1. Otherwise, set *count* = *count* + 1. Repeat Steps 2–4 five times.

Step 5: Update cooling temperature: Set $n = n + 1$. Update the temperature using (10).

Step 6: Termination. If *count* $\geqslant 3T$ or $\theta_n \leqslant 1$, stop and report the best found solution. Otherwise, go to step 2.

There are two primary differences between the simulated annealing procedures as developed by Robinson et al. (2007) and those proposed by this research. First, the neighborhood generation procedure must ensure aggregate capacity feasibility for the capacitated problem. Second, our procedures use the SPH to generate the initial feasible solution and improve upon randomly generated solutions in the search process.

5. Computational experiments and results

We conducted three computational experiments based on the designs in Erenguc (1988), Robinson and Gao (1996), Robinson and Lawrence (2004), and Federgruen et al. (2007). Table 1 summarizes the experimental designs. Optimal problem solutions for performance benchmarking are found by solving the formulation given in Section 3 using Xpress-MP Version 2003F (Xpress Optimizer Version 14.24), a state of the art optimization software package. The SPH and SAM are coded in C++. The experiments were conducted on a personal computer running a Pentium® 4 processor at 1.9 GHz.

5.1. Experiment 1

Experiment 1 is based on Erenguc (1988) and Robinson and Gao's (1996) experimental designs for the CULSP with the necessary extensions to consider capacity. The experimental factors include the number of items $I \in \{10, 20, 40\}$, planning horizon length $T \in \{12, 18, 24\}$, family setup cost $S_{t'} \in \{\$120, \$480, \$960\}$, and capacity utilization (the ratio of total demand divided by the total available capacity over the planning horizon) $CU \in \{0.2, 0.4, 0.6, 0.8\}$.

Demand is assumed to be normally distributed and varies by item and time period. Odd numbered items have a mean demand of 50 units and a standard deviation of 20 units: even numbered items have a mean demand of 100 units and a standard deviation of 20 units. Demand is randomly generated to occur in 50% of the time periods. Unit production costs are equal to zero and inventory holding cost per unit per time period is \$1.

Erenguc (1988) found that uncapacitated problems with higher family setup ratios, i.e., $S_{t'}/\sum_{i=1}^{I} s_{it'}$, are more difficult to solve. To study this factor, we construct test problems with the mean setup cost ratios per time period ranging from 0.05 to 1.6. The mean family setup cost is drawn from a normal distribution with a mean $S_{t'} \in \{\$120, \$480, \$960\}$ and a standard deviation of \$36. $S_{t'}$ is constant in all time periods in a test problem. Item setup costs, $s_{it'}$, are drawn from a normal distribution with a mean = \$60 and a standard deviation = \$18, where $s_{it'}$ varies by item, but is constant in all time periods for each individual item.

For a specified value of *CU*, the problem's demand stream is generated and then the available capacity per time period $P_{t'}$ is calculated, where $P_{t'} = \left(\sum_{i=1}^{I}\sum_{t=1}^{T} d_{it}\right)/(T * CU)$. Since backorders are not permitted, each test problem must be aggregate capacity feasible in each time period, i.e., $\sum_{t'=1}^{j} P_{t'} - \sum_{i=1}^{I}\sum_{t'=1}^{j} d_{it'} \geqslant 0$ for all $j = 1, 2, \ldots T$. A few of the randomly generated test problems for $CU \geqslant 0.8$ were aggregate capacity infeasible in an early time period(s). This was resolved by increasing $P_{t'}$ in the associated time period(s) to attain feasibility. Otherwise, $P_{t'}$ is constant across time

Table 1
Experimental designs.

Experimental design	No. of items	Setup cost (or) time between orders (TBOs)	Capacity utilization	Length of planning horizon	No. of problem instances
Based on Erenguc (1988) and Robinson and Gao (1996)	{10, 20, 40}	$s_{it'}$ – mean \$60 and std. dev. \$18 $S_{t'} \in \{\$120, \$480, \$960\}$	{0.2, 0.4, 0.6, 0.8}	{12, 18, 24}	1080
Robinson and Lawrence (2004)	{2, 4, 6, 10, 20, 30, 40}	$s_{it'} \in \{\$100, \$300\} S_{t'}$ ranges from \$190 to \$1920	{0.05, 0.45, 0.85}	12	420
Federgruen et al. (2007)	10	Low TBO – [1,3] Med. TBO – [2,6] High TBO – [5,10]	{0.5, 0.75, 0.9}	15	135

Table 2
Summary results for the Experiment 1.

Experimental factor	SPH average intermittent Optimality gap[a]			Average optimality gap[a]		Time for the heuristic (CPU second)		Xpress-MP time (CPU seconds)
	Phase 2 (%)	Phase 4 (%)	Phase 5 (%)	SPH (%)	SAM (%)	SPH	SAM	
$I = 10$	0.98	0.95	0.90	0.89	0.49	0.01	0.15	51.68
$I = 20$	0.48	0.45	0.44	0.44	0.23	0.02	0.32	90.90
$I = 40$	0.14	0.13	0.13	0.13	0.07	0.06	0.80	12.55
$T = 12$	0.51	0.49	0.47	0.46	0.17	0.01	0.10	3.77
$T = 18$	0.54	0.52	0.51	0.50	0.29	0.02	0.35	19.95
$T = 24$	0.55	0.52	0.49	0.49	0.33	0.06	0.81	131.41
$S_t = 120$	0.21	0.19	0.17	0.16	0.11	0.03	0.42	2.66
$S_t = 480$	0.46	0.44	0.42	0.41	0.22	0.03	0.42	18.72
$S_t = 960$	0.93	0.91	0.88	0.87	0.46	0.02	0.42	133.75
$CU = 0.2$	0.33	0.33	0.32	0.31	0.04	0.02	0.37	1.22
$CU = 0.4$	0.45	0.45	0.45	0.44	0.16	0.02	0.42	4.39
$CU = 0.6$	0.57	0.55	0.51	0.51	0.34	0.04	0.43	38.37
$CU = 0.8$	0.78	0.71	0.67	0.67	0.51	0.03	0.46	162.85
Overall average	0.53	0.51	0.49	0.48	0.26	0.03	0.42	51.7

[a] Opt. gap = 100 ∗ (heuristic objective value − opt. objective value)/opt. objective value.

periods. For each combination of experimental factors, ten problem instances are randomly generated resulting in a total of 1080 test problems.

5.1.1. Experiment 1 results

Table 2 summarizes the experimental results, where the performance metrics are the percent optimality gap (i.e., 100 ∗ (heuristic solution value − optimal solution value)/optimal solution value), computational requirements, and problem size.

The SPH and SAM efficiently find solutions with average optimality gaps of 0.48%, and 0.26%, gap standard deviations of 0.81% and 0.53%, and maximum gaps of 7.58% and 6.87%, respectively. SPH and SAM find optimal solutions for 318 and 460 of the 1080 test problems, respectively. The computation time for SPH averages 0.03 CPU seconds with a maximum of 0.328 CPU seconds. SAM averages 0.42 CPU seconds with a maximum of 2.515 CPU seconds. For comparison purposes, Xpress-MP's solution time averages 51.7 CPU seconds with a maximum of 9565 CPU seconds (2.65 hours).

The SPH provides feasible solution at the end of phase 2, 4 and 5, before providing the best found solution in phase 6. The solution progressively improves from phase 2 to phase 6. There is 10% improvement in the optimality gap from phase 2 to phase 6, while the time requirements for achieving this improvement is negligible as the entire SPH heuristic takes an average of 0.03 CPU seconds. As expected most of the improvement occurs in the first two phases, the marginal improvements in the subsequent phases are constrained by the local optimal solution obtained in the initial phases. To escape the entrapment at local optima we developed the SAM.

The optimality gaps of both heuristics are positively correlated with the family setup costs (i.e., family fixed cost ratios), and capacity utilization. SAM results are also positively correlated with the number of time periods in the planning horizon. SPH and SAM performance is negatively correlated with the number of items. All optimality gaps for the factor summaries are below 0.90% indicating robust performance.

As expected, the computational requirements for both procedures increase with the number of items and length of the planning horizon but are fairly constant across family setup costs and capacity utilization. On average SAM improves SPH solution quality by 46% at a cost of 0.39 CPU seconds.

Fig. 1 indicates a two-way interaction between capacity utilization and the number of items where lower quality solutions are associated with fewer items and higher capacity utilization levels. This result is explained by viewing a larger number of items as

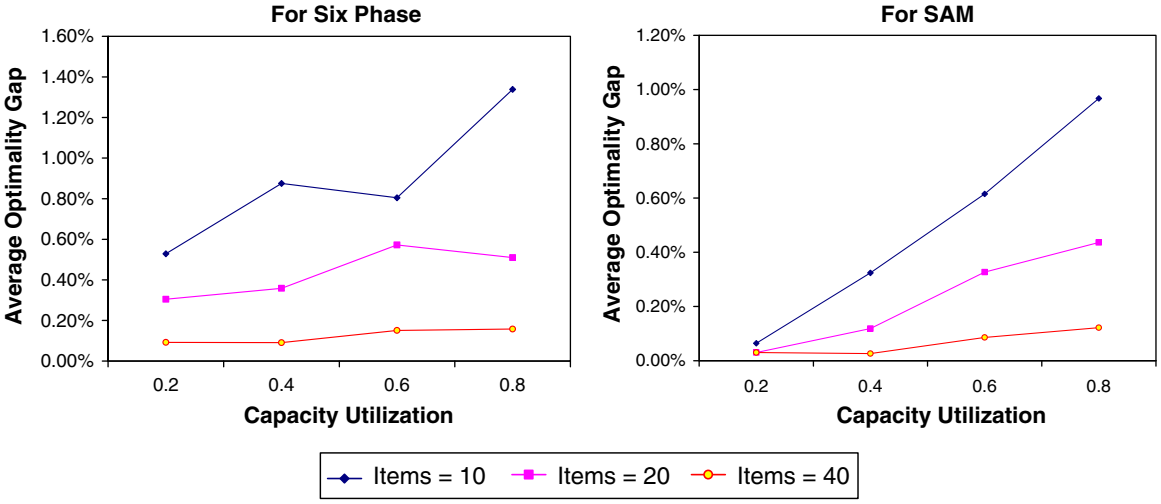

Fig. 1. Two-way interaction between number of items and capacity utilization.

increased granularity, which permits the items to more effectively fit within the capacity constrained resource.

5.2. Experiment 2

Experiment 2 compares the performance of the SPH and SAM procedures with the Robinson and Lawrence (2004) Lagrangean heuristic (RLH). Since their computer code and experimental data is not available for direct comparison, we replicate their experimental design to facilitate the comparison. The experimental factors include number of items $I \in \{2, 4, 6, 10, 20, 30, 40\}$, item setup cost $s_{it'} \in \{\$100, \$300\}$, and capacity utilization $CU \in \{0.05, 0.45, 0.85\}$. We consider all possible combinations of experimental factors and generate ten random problems for each combination resulting in 420 test problems.

The planning horizon length is 12 time periods for all test problems. Item demand for each time period is randomly generated from a uniform distribution on the interval [50,150]. To represent demand lumpiness, item demand is set to zero when the generated demand value is less than 60 units. Hence each item experiences demand in approximately 90% of the time periods.

As in Robinson and Lawrence (2004) and Erenguc (1988), major setup costs range from \$190 to \$1920, with larger setup costs assigned to problems with more items. This reduces the effect of the family setup cost ratio on the experimental results. Replenishment capacities are determined as described in Experiment 1.

5.2.1. Experiment 2 results

Table 3 summarizes the results by capacity utilization level. For $CU = 0.85$, RLH could not verify optimal solutions for problems with more than 10 items. The results reported in Table 3 for the RLH reflect this limitation. Using a more efficient problem formulation and more powerful solver than Robinson and Lawrence (2004), we obtain optimal benchmark solutions for all but four test problems. The average (maximum) MIP gap of these four problems is 0.19% (0.45%), justifying their use as performance benchmarks.

SAM finds substantially better solutions than SPH and RLH at all capacity utilization levels. Average optimality gaps for SAM, SPH and RLH are 0.51%, 1.53% and 3.02%, respectively. The SPH performs better than RLH in moderate and tight capacity situations, while RLH finds better solutions for the relatively uncapacitated problems. Optimality gaps increase with higher capacity utilization and minor setup costs for all solution procedures. Similarly, Xpress-MP solution times increases exponentially as the capacity utilization increases, requiring more than two hours to obtain and verify optimality in some instances when $CU = 0.85$. In contrast, SPH and SAM solution times are nominal at all capacity utilization levels. The SPH requires milliseconds to solve the test problems, while SAM requires on average 0.13 CPU seconds. In contrast, the RLH could not obtain solutions in less than 100 CPU minutes for tightly constrained problems with more than 10 items. The SPH and SAM provide higher quality solutions, require less computational time, and solve larger sized problems than the RLH.

5.3. Experiment 3

Maes and Van Wassenhove (1988) and Federgruen et al. (2007) find that solution difficulty for MCLSPs increases with longer times between orders (*TBO*). Since the test problems in Experiments 1 and 2 have relatively low *TBO* values, Experiment 3 replicates the high *TBO* experimental design in Federgruen et al. (2007) and compares the performance of the SPH and SAM with Federgruen et al. (2007)'s reported findings for their best performing progressive interval/expanding horizon heuristic (EHH).

The experimental factors include capacity utilization $CU \in \{0.5, 0.75, 0.9\}$, three levels of item and family *TBO*, where $TBO_{Item} = \sqrt{2s_{it'}/hd}, TBO_{Family} = \sqrt{2S_{t'}/hD}, d$ is the average item demand per time period, and D is the average family demand per time period. Low, medium and high *TBO*-values are randomly generated from uniform distribution intervals [1,3], [2,6] and [5,10], respectively and then the associated setup costs are calculated. The available capacity $P_{t'}$ is computed from the *CU* as discussed in previous sections.

Individual item demand per period is randomly generated from a normal distribution with mean of 100 and a standard deviation of 10. The inventory holding cost per unit per period is \$1.00. All problems consider 10 items and 15 periods. The full factorial design has 27 combinations of factors. Since finding optimal solutions requires more than 1 hour CPU time, we generate five test problems for each combination of factors as in Federgruen et al. (2007).

5.3.1. Experiment 3 results

Xpress-MP required on average 4251.45 CPU seconds to find optimal problem solutions for this experiment. For the problems that did not solve to optimality within a 2-hour time limit, the average MIP gap is 0.82%. Only 18 of these problems had an MIP gap greater than 1.0%. We used the best found solution for benchmarking.

The SPH and SAM procedures find solutions with average optimality gaps of 9.92%, and 2.08%, standard deviations of optimality gaps of 11.7% and 1.97%, and CPU running times of 0.006 seconds and 0.10 seconds, respectively. SAM had a maximum optimality gap of 8.65% and 65% of the problems had gaps less than its average 2.08% optimality gap. The findings are summarized in Table 4 by TBO_{Item} and TBO_{Family}. For both procedures, solution quality is lower at higher TBO_{Item} values but higher at lower TBO_{Family} values. These results are consistent with the observations in Maes and Van Wassenhove (1988) that improvement heuristics, such as SPH, may not be very effective in solving high TBO_{Item} problems. However, the SAM procedure significantly improves the poor quality of the SPH solutions. For example, on the most difficult problems with high TBO_{Item} and low TBO_{Family}, the average SPH optimality gap of 29.99% is reduced to 5.71% by SAM. Across all test problems SAM improves the SPH optimality gaps by an average of 80%.

Table 3
Summary results Experiment 2.

Capacity utilization	Average optimality gap[a]			Time for the heuristic (seconds)			Xpress-MP time (seconds)
	RLH (%)	SPH (%)	SAM (%)	RLH[b]	SPH	SAM	
0.05	0.44	0.81	0.05	6.90	0.001	0.12	1.68
0.45	3.91	1.48	0.46	10.47	0.003	0.16	12.40
0.85	4.72[c]	2.30	1.02	11.91[c]	0.014	0.11	1155.63[d]

[a] $100 * (\text{heuristic objective value} - \text{optimal objective value})/\text{optimal objective value}$.
[b] RLH was run in IBM 3090-400E systems and was coded in FORTRAN 90.
[c] For $CU = 0.85$, only problems with up to 10 items were solved by Robinson and Lawrence (2004).
[d] Four test problems could not be solved within a 2 hour time limit.

Table 4
Summary results for Experiment 3.

TBO_{Family}	Low			Medium			High		
	Six-phase	SAM	Xpress-MP[a]	Six-phase	SAM	Xpress-MP[a]	Six-phase	SAM	Xpress-MP[a]
Low TBO_{Item}[b]	1.72%	1.18%	–	1.09%	0.58%	–	0.38%	0.19%	–
CPU time (seconds)	0.00	0.14	1938	0.01	0.12	2569	0.01	0.11	3346[c]
Medium TBO_{Item}[b]	10.09%	2.98%	–	5.35%	1.59%	–	2.23%	0.63%	–
CPU time (seconds)	0.01	0.10	3376	0.00	0.11	3843[c]	0.01	0.10	4285[c]
High TBO_{Item}[b]	29.99%	5.71%	–	20.81%	3.86%	–	11.04%	1.97%	–
CPU time (seconds)	0.00	0.07	5983[c]	0.01	0.08	6012[c]	0.01	0.07	6914[c]

[a] Optimal or Best integer solution obtained from Xpress-MP formulation in Section 3.
[b] Each cell in the row represents the average optimality gap 100 ∗ ((heuristic objective value − optimal objective value)/optimality objective value) of five test problems.
[c] Indicates that one or more problem instances could not be optimally solved within a 2 hour time limit.

Table 5
Experiment 3 detailed results for SAM.

Capacity utilization (*CU*)			0.5			0.75			0.90		
TBO_{Family}			Low	Medium	High	Low	Medium	High	Low	Medium	High
Low TBO_{Item}	Opt. gap[b]	SAM (%)	0.67	0.48	0.16	1.65	0.68	0.27	1.21	0.58	0.15
	CPU time (seconds)	SAM	0.24	0.20	0.20	0.13	0.11	0.10	0.04	0.04	0.03
		Xpress-MP	23	128	613	400	2014	2727	5391[a]	5564[a]	6698[a]
Medium TBO_{Item}	Opt. gap[b]	SAM (%)	2.44	1.44	0.55	2.86	1.49	0.57	3.65	1.85	0.76
	CPU time (seconds)	SAM	0.18	0.21	0.19	0.09	0.10	0.09	0.03	0.02	0.02
		Xpress-MP	246	762	559	2516	3394[a]	4920[a]	7366[a]	7372[a]	7375[a]
High TBO_{Item}	Opt. gap[b]	SAM (%)	3.84	2.05	1.07	6.96	4.48	2.01	6.34	5.06	2.83
	CPU time (seconds)	SAM	0.13	0.13	0.13	0.05	0.07	0.07	0.03	0.03	0.02
		Xpress-MP	3074[a]	3163[a]	5921[a]	7402[a]	7489[a]	7412[a]	7474[a]	7383[a]	7410[a]

[a] Indicates that one or more problem instances could not be solved to optimality within the pre-set time limit of 2 hours.
[b] Average optimality gap for the heuristic = (heuristic objective value − optimal objective value)/optimality objective value.

Table 5 displays the results for SAM and Xpress-MP by TBO_{Item}, TBO_{Family}, and *CU*. The results confirm that increasing capacity utilization increases the difficulty for SAM to find good heuristic solutions. However, TBO_{Item} is the primary driver of solution quality. While Xpress-MP experiences considerable difficulty finding optimal solutions for medium and high TBO_{Item} problems and high capacity utilization problems, the results for SAM show the opposite impact.

While the SAM finds high quality solutions (2.08% average optimality gap), it is not as effective as the EHH in Federgruen et al. (2007) which has an average 1.2% optimality gap. However, SAM's comparative advantage lies in its computational efficiency and ability to solve large scale sized problems. Where SAM averages less than 0.10 CPU seconds to solve the test problems EHH averages 16.6 CPU seconds. While this difference for the 10-item and 15-period test problems may not seem material, CPU requirements increase rapidly with problem size for the EHH. This is illustrated in Table 6, as reported in Federgruen et al. (2007), for medium difficulty test problems with *CU* = 0.75, medium TBO_{Item} and medium TBO_{Family}. For example, a 10-item and 10-period problem requires 29 CPU seconds, while a 25-item and 10-period problem requires approximately 20,335 CPU seconds (5.64 hours) to solve. Considering that industry applications in procurement or transportation planning often require coordinating more than 25 items over a 24 time period horizon, solution time is a major limitation of the EHH procedure. On the other hand, the SAM efficiently handles large problems with no degradation in heuristic performance. SAM requires just 3 CPU seconds to solve the 40-item and 24-time period problems in Experiment 1. Furthermore, high TBO_{Item} and high *CU* problems which confound EHH solutions times are relatively easier to solve for the SAM.

As a final comparison, Federgruen et al. (2007) also propose a strict partitioning heuristic that averages less than 1 CPU second to solve Experiment 3's test problems. However, while offering no computational advantage, the strict partitioning heuristic's optimality gap is 14.7% versus 2.08% for the SAM and 9.92% for the SPH.

Table 6
Running times in seconds for EHH from Federgruen et al. (2007).

Periods	10	25	50
5 items	7	42	124
10 items	29	184	524
15 items	416	2694	4310
20 items	1600	9372	16159
25 items	20335	66634	58264

6. Conclusions and implications

Capacitated coordinated lot-size problems are commonly encountered during the management of production and distribution systems. However, the problem's mathematical complexity, which contains both complicating capacity constraints and joint setup costs, has thwarted past research efforts in their attempt to design effective heuristic and optimization-based approaches for the problem. This research proposes a six phase improvement heuristic and a simulated annealing meta-heuristic for the CCLSP. Over a wide range of test problems, the SPH and SAM procedures find heuristic solutions with average optimality gaps of 1.53% and 0.47%, respectively. Computational requirements average 0.023 CPU seconds for the SPH and 0.32 CPU seconds for the SAM. The SAM provides considerable advantages over the existing heuristics in the literature. It is more efficient and provides higher quality solutions than the Lagrangean heuristic in Robinson and Lawrence (2004). When compared to the EHH in Federgruen

et al. (2007), SAM finds slightly higher cost solutions (2.08% versus 1.2%), but quickly solves industry sized problems in seconds that cannot be solved in an hour by the EHH.

SAM's combination of providing high quality solutions over a wide combination of parameter settings, extreme computational efficiency, and the capability of solving large-scale problems makes it an attractive heuristic for industry application. We envision the SPH and SAM being used as stand alone solvers, embedded within requirements planning software, and as upper bounding procedures for complex optimization based algorithms. These heuristics are currently being tested as a solver in the procurement planning software for one of the nation's leading electronic and direct mail marketers. Preliminary results are promising where problems with 239 items and 26 time periods are solved in less than 14 seconds. These results strongly support SAM's potential application in logistics, operations and supply chain planning software.

Appendix A. Six phase heuristics

A.1. Subroutine I: cost minimizing left-shift

The subroutine begins with a problem that is aggregate capacity feasible, but may violate capacity constraints in individual time periods. A cost reducing procedure iteratively left-shifts the cost minimizing lot-size(s) of either an individual item or a family of items from period t into period $t' < t$ until no further savings are possible. The procedure only considers left-shifting production into open periods (i.e., those with a scheduled family setup). The savings, $C_i(t',t)$, for left-shifting item i's production from period t into t' is:

$$C_i(t',t) = (c_{it} - c_{it'})a_{it} + (Y_{it'} - 1)s_{it'} + s_{it} - I_i(t',t) \quad \text{for } a_{it} > 0 \text{ and 0 otherwise},$$

where $Y_{it'} = 1$ if production is scheduled for item i in period t', s_{it} is the item's setup cost in period t, the inventory carrying cost from period t' to period t is $I_i(t',t) = (t - t')h_i a_{it}$, h_i is the unit inventory carrying cost per period for item i, a_{it} is the current production quantity for item i in period t, $c_{it'}$ is the unit cost associated with ordering item i in period t'. The unit production cost is assumed to zero or constant across all periods in the experimental design. The savings associated with re-scheduling the entire product family from period t into period t' is

$$C(t',t) = (Z_{t'} - 1)S_{t'} + S_t + \sum_{i=1}^{I} C_i(t',t),$$

where $Z_{t'} = 1$ if a family setup is scheduled in period t' and 0 otherwise, and S_t is the product family setup cost in period t.

The maximum quantity that can be left-shifted into period t', $E_{t'}$, is the unallocated capacity in open periods from 1 to t' less the capacity shortage in periods $t'+1$ to $t-1$. Defining P_j as the total available capacity for period j, the design capacity minus the family setup time, the unallocated capacity in open period j is

$$\tilde{e}_j = \left(P_j - \sum_{i=1}^{I} a_{ij}\right) Z_j,$$

where, $\tilde{e}_j > 0$ indicates capacity is available and $\tilde{e}_j < 0$ indicates that the current production schedule exceeds the available capacity of period j. The capacity shortage in period $t'+1$ to $t-1$ that must be supplied from period t' or earlier is

$$G(t'+1, t-1) = Min\{0, \tilde{e}_{t'+1}, \tilde{e}_{t'+1} + \tilde{e}_{t'+2}, \tilde{e}_{t'+1} + \tilde{e}_{t'+2} + \tilde{e}_{t'+3}, \ldots, \tilde{e}_{t'+1} + \cdots + \tilde{e}_{t-1}\},$$

where $G(t'+1, t-1) < 0$ signals a shortage. The maximum quantity that can be left-shifted into period t' is

$$E_{t'} = Max\left\{0, \sum_{j=1}^{t'} \tilde{e}_j - |G(t'+1, t-1)|\right\}.$$

The steps of the subroutine follow.

Step 1. Compute unallocated capacities. Compute the unallocated capacity, e_t, in each period t for the current production schedule, where

$$e_t = P_t - \sum_{i=1}^{I} a_{it} \quad \text{for } t = 1, 2, \ldots, T.$$

Step 2. Compute item and family savings. Calculate $C_i(t',t)$ and $C(t',t)$ for all $i, t' < t$, and t for which $E_{t'} > 0$. Each item lot-size with $C_i(t',t) > 0$ and $a_{it} \leqslant E_{t'}$ and each family of items with $C(t',t) > 0$ and $\sum_{i=1}^{I} a_{it} \leqslant E_{t'}$ are candidates for left-shifting from period t into period t'. In addition, a product family cannot be left-shifted from period t if removing the capacity violates aggregate capacity feasibility when considering the opened production periods. Specifically, the following must hold:

$$\sum_{j=1}^{t-1} \tilde{e}_j - \sum_{i=1}^{I} a_{it} + \sum_{j=t+1}^{T} \tilde{e}_j \geqslant 0$$

2a. Item saving adjustment. Shifting a production lot-size into period t' may exceed the period's available capacity. In this case, some of the production load in t' must move into an earlier time period(s) causing additional inventory holding costs. The quantity moved earlier, $f_{t'}$, is

$$f_{t'} = |Min\{0, e_{t'} - a_{it}\}|,$$

where a_{it} is the quantity rescheduled from period t to period t'. The adjusted item cost savings is $AC_i(t',t) = C_i(t',t) - Cf_{t'}$, where $Cf_{t'}$ is the incremental inventory carrying costs associated with rescheduling quantity $f_{t'}$ into a period earlier than t'. The penalty $Cf_{t'}$ is calculated by first identifying $f_{t'}$, the capacity violation in period t'. Next, the procedure looks earlier in time for the first available capacity and computes the minimal incremental inventory holding costs if $f_{t'}$ is moved into this time period. If the available capacity is insufficient to accommodate $f_{t'}$, the penalty for moving the overflow units earlier in time is calculated. The procedure continues until the incremental inventory holding costs for moving all $f_{t'}$ units into available capacity slots is determined. The adjusted cost savings, $AC_i(t',t)$, is the maximum potential savings obtained by left-shifting item i into period t'.

2b. Family saving adjustment. Rescheduling a family of items into an earlier time period may require two types of capacity related inventory cost adjustments.

Type 1 cost adjustment, CN_t. When rescheduling a family of items from period t to t', we eliminate the family setup and thereby the capacity in period t. Part of this capacity, N_t, may have been used to satisfy demand in periods greater than time t, which must now be supplied from a period(s) earlier than t, thereby increasing inventory holding costs. Specifically, N_t is expressed as:

$$N_t = Min\{Max\{0, \tilde{e}_t\}, |G(t+1, T)|\},$$

where, $Max\{0, \tilde{e}_t\}$ is the available capacity in period t that can be used to satisfy demand in periods $t+1$ to T, and $|G(t+1, T)|$ is the capacity shortage in periods $t+1$ to T that is currently supplied from production in period t or earlier. Mathematically,

$$G(t+1, T) = Min\{0, \tilde{e}_{t+1}, \tilde{e}_{t+1} + \tilde{e}_{t+2}, \tilde{e}_{t+1} + \tilde{e}_{t+2} + \tilde{e}_{t+3}, \ldots, \tilde{e}_{t+1} + \cdots + \tilde{e}_T\}.$$

Two cases are possible.

Case 1: $N_t = |G(t+1,T)|$. In this case, $|G(t+1,T)|$ was entirely supplied by period t., The minimum possible increase in inventory costs CN_t is computed in a similar manner as for $Cf_{t'}$.

Case 2: $N_t < |G(t+1,T)|$. In this situation, only a part of $|G(t+1,T)|$ is supplied by period t with the remainder $|G(t+1,T)| - N_t$ units coming from an earlier period(s). In this case, we first temporarily adjust the available capacity as necessary in the open periods from 1 to $t-1$ to account for the quantity $|G(t+1,T)| - N_t$ and then compute the cost adjustment, CN_t.

Type 2 cost adjustment, $CF_{t'}$. Left-shifting the product family into period t' may exceed the period's capacity. In this situation, a portion of the production must be moved into a period earlier than t' thereby, incurring additional inventory holding costs. The quantity moved forward is

$$F_{t'} = \left| Min\left\{0, e_{t'} - \sum_{i=1}^{I} a_{it}\right\}\right|.$$

The minimum possible increase in inventory holding costs for moving $F_{t'}$ units forward is $CF_{t'}$, which is computed similar to $Cf_{t'}$.The adjusted family cost savings, $AC(t',t) = C(t',t) - CN_t - CF_{t'}$, is the maximum potential savings from left-shifting the product family from period t to t'.

Step 3. Left shift phase. If all $AC_i(t',t) \leqslant 0$ and $AC(t',t) \leqslant 0$, STOP, otherwise select the Maximum$\{AC_i(t',t), AC(t',t)\}$ for all $i, t' < t$, and t for left-shifting. If the maximum savings calls for left-shifting item i, update the order schedules by setting $a_{it'} = a_{it'} + a_{it}$ and $a_{it} = 0$. Otherwise, the maximum saving is associated with left-shifting a product family, so set $a_{it'} = a_{it'} + a_{it}$ and $a_{it} = 0$ for all i. Go to *Step 1*.

A.2. Subroutine II: Feasibility seeking left-shift

Subroutine II left-shifts production as necessary to guarantee capacity feasibility in each time period while minimizing the increase in schedule cost.

Step 1. Initialize. Set $t = T$.

Step 2. Check individual time periods for feasibility. If the production in period t is within the capacity limit, i.e., $e_t \geqslant 0$, go to *Step 6.* Otherwise, continue.

Step 3. Insure sufficient capacity is available in earlier opened time periods. If sufficient capacity is available to cover the shortage in time t, $|e_t|$, then continue. Otherwise, schedule a family setup in the time period(s) immediately prior to t until sufficient capacity is available to cover the shortage.

Step 4. Compute the marginal cost of rescheduling. For each item i scheduled in period t, calculate the cost of producing the item in the immediate preceding open time period t'. The amount rescheduled, r_{it}, is either a_{it} or $|e_t|$ according to the following two cases.

Case 1: If $|e_t| > a_{it}$, then $r_{it} = a_{it}$. The marginal cost of rescheduling item i from period t to t' is $MC_i(t',t) = (t-t')h_i r_{it} + (1 - Y_{it'})s_{it'} - s_{it}$.

Case 2: If $|e_t| \leqslant a_{it}$, at least $|e_t|$ units must be transferred for feasibility. However, assuming sufficient capacity is available in earlier time periods, the entire lot size, a_{it}, could be rescheduled into period t' if it results in lower incremental cost. The value of r_{it} yielding the minimum cost in the following equation is rescheduled

$$MC_i(t',t) = Min\left\{\begin{array}{l}[(t-t')h_i r_{it} + (1-Y_{it'})s_{it'}, \text{ where } r_{it} = |e_t|] \\ [(t-t')h_i r_{it} + (1-Y_{it'})s_{it'} - s_{it}, \text{ where } r_{it} = a_{it}]\end{array}\right\}$$

Step 5. Left-shift phase. Select the Minimum $\{MC_i(t',t)\}$ for all i and $t' < t$ and reschedule its production by setting $a_{it'} = a_{it'} + r_{it}$ and $a_{it} = a_{it} - r_{it}$. Next, update the available capacity in periods t' and t by setting $e_{t'} = e_{t'} - r_{it}$ and $e_t = e_t + r_{it}$. If $e_t < 0$, go to *Step 4* otherwise go to *Step 6*.

Step 6. Roll back. If $t = 1$ stop, otherwise set $t = t - 1$ and go to *Step 2*.

A.3. Subroutine III: cost minimizing right-shift

Subroutine III attempts to reduce costs by shifting production as late as possible subject to capacity availability. The procedure begins at time $t = T$ and iteratively works backward. For each period t with $e_t > 0$, the cost savings associated with shifting earlier production into period t from t' is calculated. The shift resulting in the maximum savings is implemented. The procedure continues shifting production into period t until $e_t = 0$ or further savings are not possible. The algorithm then moves to period $t-1$ and continues. A single item or a product family is candidate for right-shifting. The maximum quantity of item i that can be shifted from period t' to t, $v_{it'}$, is the minimum of the unallocated capacity in period t, e_t, and the quantity available to transfer out of period t' as detailed in the following equation:

$$v_{it'} = Min\left\{e_t, \left(a_{it'} - \left[\begin{array}{c}\text{Net requirement for} \\ \text{item } i \text{ in period } t'\end{array}\right] - \left[\begin{array}{c}\text{Net requirement for item } i \\ \text{from period } t'+1 \text{ to } t-1\end{array}\right]\right)\right\}.$$

The net requirement for item i in time period t' is $Max\left[\left(d_{it'} - \sum_{q=1}^{t'-1}(a_{iq} - d_{iq})\right), 0\right]$.

For $v_{it'} > 0$, the resulting cost savings is $H_i(t',t) = h_i v_{it'}(t-t') - s_{it}(1 - Y_{it}) + s_{it'}X_{it'}$, where $X_{it'} = 1$ if the complete lot-size is shifted and 0, otherwise. For $v_{it'} = 0, H_i(t',t) = 0$. A family of items is feasible for right shifting when $\sum_{i=1}^{I} v_{it'} \leqslant e_t$ with a cost savings of $H(t',t) = S_{t'}X_{t'} - S_t(1 - Z_t) + \sum_{i=1}^{I} H_i(t',t)$, where $X_{t'} = 1$ if the whole product family is shifted from t' to t. The steps of the subroutine follow.

Step 1. Initialize. Set $t = T$.

Step 2. Check for capacity availability. If $e_t \leqslant 0$ go to *Step 5.*

Step 3. Identify potential cost saving. For $i = 1, 2, \ldots, I$ and $t' = 1, 2, \ldots, t-1$ compute $v_{it'}$ and $H_i(t',t)$. If $\sum_{i=1}^{I} v_{it'} \leqslant e_t$ then compute $H(t',t)$. If all $H_i(t',t) \leqslant 0$ and $H(t',t) \leqslant 0$ go to *Step 5.*

Step 4. Right-shift phase. Select the Maximum $\{H_i(t',t), H(t',t)\}$ for all i and t' for right-shifting into period t. If the maximum savings is for right-shifting item i, update $a_{it'} = a_{it'} - v_{it'}, a_{it} = a_{it} + v_{it}, e_{t'} = e_{t'} + v_{it'}$ and $e_t = e_t - v_{it'}$. Otherwise, the maximum savings is associated with rescheduling a product family, so set $a_{it'} = a_{it'} - v_{it'}$ and $a_{it} = a_{it} + v_{it}$ for all $i, e_{t'} = e_{t'} + \sum_{i=1}^{I} v_{it'}$, and $e_t = e_t - \sum_{i=1}^{I} v_{it'}$. If $e_t \leqslant 0$, go to *Step 5* otherwise, go to *Step 3*.

Step 5. Roll back. If $t = 1$ stop, otherwise set $t = t - 1$ and go to *Step 2*.

References

Arkin, E., Joneja, D., Roundy, R., 1989. Computational complexity of uncapacitated multi-echelon production planning problems. Operations Research Letters 8, 61–66.

Atkins, D.R., Iyogun, P.O., 1988. A heuristic with lower bound performance guarantee for the multi-product dynamic lot-size problem. IIE Transactions 20 (4), 369–373.

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 (7), 993–1007.

Boctor, F.F., Laporte, G., Renaud, J., 2004. Models and algorithms for the dynamic-demand joint replenishment problem. International Journal of Production Research 42 (13), 2667–2678.

Dogramaci, A., Panayiotopoulos, J.C., Adam, N.R., 1981. The dynamic lot-sizing problem for multiple items under limited capacity. AIIE Transactions 13 (4), 294–303.

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

Eppen, G.D., Martin, R.K., 1987. Solving multi-item capacitated lot-sizing problems using variable redefinition. Operations Research 35, 832–848.

Erenguc, S.S., 1988. Multi-product dynamic lot-sizing model with coordinated replenishments. Naval Research Logistics 35, 1–22.

Erenguc, S.S., Mercan, H.M., 1990. A multifamily dynamic lot-sizing model with coordinated replenishments. Naval Research Logistics 37, 539–558.

Federgruen, A., Tzur, M., 1994. The joint replenishment problem with the time-varying costs and demands: Efficient, asymptotic and ε-optimal solutions. Operations Research 42, 1067–1086.

Federgruen, A., Meissner, J., Tzur, M., 2007. Progressive interval heuristics for multi-item lot-sizing problems. Operations Research 55 (3), 490–502.

Fogarty, D.W., Barringer, R.L., 1987. Joint order release decisions under dependent demand. Production and Inventory Management Journal 28, 55–61.

Gao, L., Robinson, E.P., 2003. A Lagrangian/dual-ascent based heuristic for the capacitated coordinated replenishment with dynamic demand and backlogging. In: Proceedings of Decision Sciences Institute Annual Meeting November 2003, Washington DC.

Haseborg, F., 1982. On the optimality of joint ordering policies in a multi-product dynamic lot size model with individual and joint set-up costs. European Journal of Operational Research 9, 47–55.

Iyogun, P., 1991. Heuristic methods for the multi-product dynamic lot size problem. Journal of the Operational Research Society 42 (10), 889–894.

Jans, R., Degraeve, Z., 2007. Meta-heuristics for lot sizing problems: Review and comparison of solution approaches. European Journal of Operational Research 177, 1855–1875.

Jans, R., Degraeve, Z., 2008. Modeling industrial lot sizing problems. International Journal of Production Research 46 (6), 1619–1643.

Joneja, D., 1990. The joint replenishment problem: New heuristics and worst case performance bounds. Operations Research 38 (4), 711–723.

Kao, E.P.C., 1979. A multi-product dynamic lot-size model with individual and joint setup costs. Operations Research 27 (2), 279–289.

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.

Karni, R., Roll, Y., 1982. A heuristic algorithm for the multi-item lot-sizing problem with capacity constraints. IIE Transactions 14, 249–256.

Kirca, O., 1995. A primal-dual algorithm for the dynamic lot-sizing problem with joint setup costs. Naval Research Logistics 42, 791–806.

Krarup, J., Bilde, O., 1977. Plant location, set covering and economic lot size: An O(mn) – Algorithm for structured problems. Numerische Methoden bei Optimierungsaufgaben, Band 3: Optimierung bei Graph-Theoretischen und Ganz- zahligen Problemen. Switzerland, pp. 155–180.

Maes, J., Van Wassenhove, L., 1988. Multi-item single-level capacitated dynamic lot-sizing heuristics: A general review. Journal of the Operational Research Society 39 (11), 991–1004.

Pochet, Y., Wolsey, L.A., 2006. Production Planning by Mixed Integer Programming. Springer, New York.

Robinson, E.P., Gao, L., 1996. A dual ascent procedure for multi-product dynamic demand coordinated replenishment with backlogging. Management Science 42 (11), 1–9.

Robinson, E.P., Lawrence, F.B., 2004. Coordinated capacitated lot-sizing problem with dynamic demand: A Lagrangian heuristic. Decision Sciences Journal 35 (1), 25–54.

Robinson, E.P., Narayanan, A., Gao, L., 2007. Effective heuristics for the dynamic demand joint replenishment problem. Journal of Operational Research Society 58 (6), 808–815.

Robinson, E.P., Narayanan, A., Sahin, F., 2009. Coordinated deterministic dynamic demand lot sizing problem: A review of models and algorithms. Omega: The International Journal of Management Science 37 (1), 3–15.

Shapiro, R., Rosenfield, D., Stecke, K.E., 2002. Baker Precision Instruments, Inc., HBS-9-687-052, Boston, Harvard Business School Publishing.

Silver, E.A., 1979. Coordinated replenishment of items under time-varying demand: Dynamic programming formulation. Naval Research Logistics Quarterly 26 (1), 141–151.

Silver, E.A., Kelle, P., 1988. More on 'Joint order release decisions under dependent demand'. Production and Inventory Management Journal 29 (2), 71–72.

Veinott Jr., A.F., 1969. Minimum concave-cost solution of Leontief substitution models of multi-facility inventory systems. Operations Research 17, 262–291.

Zangwill, W.I., 1966. A deterministic multiproduct, multifacility production and inventory system. Operations Research 14, 508–846.