생산 순서에 영향을 받는 준비 시간과 준비 비용을 고려한 용량 제약이 있는 로트사이징 문제의 시뮬레이티드 어닐링 해법

A Simulated Annealing Algorithm for the Capacitated Lot-sizing and Scheduling problem under Sequence-Dependent Setup Costs and Setup Times

정지영, 박성수
한국과학기술원 산업공학과

Abstract

In this research, the single machine capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times (CLSPSD) is considered. This problem is an extension of capacitated lot-sizing and scheduling problem (CLSP) with an additional assumption of sequence-dependent setup costs and setup times. The objective of the problem is minimizing the sum of production costs, inventory holding costs and setup costs while satisfying the customer demands.

It is known that CLSPSD is NP-hard. In this paper, a MIP formulation is presented. To handle the problem more efficiently, a conceptual model is suggested, and one of the well-known meta-heuristics, simulated annealing approach is applied. To illustrate the performance of this approach, various instances are tested and the results of this algorithm are compared with those of CLPEX. Computational results show that the approach generates optimal or near optimal solutions.

Keywords: lot-sizing, sequence-dependent setup costs, simulated annealing algorithm

1. Introduction

The goal of this paper is to establish efficient production and inventory plan which satisfies customer demands with minimum overall costs including production costs, inventory holding costs and setup costs. In such competitive situations like these days, the production and inventory plan with minimum associated costs is an important issue, and modeling of the successful production plan can give great benefit to companies. In the view of just-in-time(JIT) philosophy, frequent setups are suitable for supplying the right part in right time and those can reduce inventory holding costs. However, as the number of setups grows, it can generate high setup costs and long setup times. Hence, we have to think over sequencing and determining of production quantity together. Despite of the intimate relations between lot-sizing and sequencing, only few attempts have been made at the capacitated lot-sizing and scheduling problem (CLSP). Moreover, most of the existing researches can not demonstrate the real world situations including sequence-dependent setup costs and setup times. In this paper, we consider the CLSPSD with a single machine produce multi-items over multiple periods.

The lot-sizing and scheduling problem can be

classified into two different ways by the time period, i.e. macro-periods or micro period, (A. Drexl, A.Kimms [1]). First is the discrete lot-sizing and scheduling problem (DLSP), also called the "small bucket" problems. In DLSP, the time periods are short and only one item may be produced per period. Second is the capacitated lot-sizing and scheduling problem (CLSP), known as the "big bucket" problems. Multiple items may be produced during a single period in CLSP. In this research, we extend the lot-sizing and scheduling problem to the lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. In DLSP, Salomon et al.[2] reformulate DLSP with sequence-dependent setup costs and setup times(DLSPSD) as traveling salesman problem with time windows(TSPTW) and developed the exact solution method. On the other hand, in CLSP, K.Hasse[3], and K. Sungmin et al.[4] consider the CLSP with sequence-dependent setup costs. However, they assume that setup times are zero. Another extensive works that allow product dependent setup times but sequence-independent setup costs are researched by Aras, O.A., Swanson, L.A. [6], Trigeiro et al.[7], Diaby et al. [8]. CLSPSD is handled by Diwakar Gupta, Thorkell Magnusson[5] and they provide a formulation and present the heuristic algorithm. However, the average deviation of heuristic solutions from optimal solutions is large. Therefore, we develop a simulated annealing algorithm that can give good performance. The CLSPSD is NP-hard, since CLSP with setup times is special case of the CLSPSD and it is known that CLSP is NP-hard(Garey and Johnson[9]). Therefore, many authors have developed heuristic to solve the CLSP and only a few attempts have been made to solve the CLSP optimally. Barany I., Van Roy TJ, Wolsey L.[10] reformulated the CLSP without setup times adding a strong valid inequality, and they optimally solved this problem using a cutting plane algorithm. Eppen,G.D.,Martin, R.K.[11] also optimally solved the multi-item capacitated lot-sizing problems using variable redefinition. Another various heuristic approaches are proposed by Mohan Gopalakrishnan et. al [13], Kuik, R., Salomon, M.[12] and Ou Tang [14]. In this paper, we reformulate the CLSPSD under the situation in which empty setup is not allowed, and develop the simulated annealing algorithm.

This paper is organized as follows. Section 2 describes the problem and a mathematical formulation of CLSPSD is presented. The simulated annealing algorithm is explained in detail in section 3. In section 4, numerical examples are provided and we discuss computational results. Finally, concluding remarks are drawn.

2. Problem description and formulation

In this paper, the capacitated single-machine multi-item lot-sizing and scheduling problem is considered. Demand, d_{it} , for item $i = 1,...,N$ over a time horizon $t = 1,...,T$ is given. These are dynamic and deterministic. All items must be produced on a single machine which has finite capacity. Machine capacity is expressed in units of time. More than one item is produced in a period and a setup must take place to prepare proper operation if different items are produced in a single machine. Setup times depend on the production sequence, so-called sequence-dependent setup times, and satisfy the triangular inequality, i.e. $st_{ik} + st_{kj} \geq st_{ij}$, for all $i, j, k = 1,...N$, where *N* is the number of different items to be considered. Total costs include production costs, inventory holding costs, and setup costs. We

assume that the production costs are uniform regardless of items. Setup costs also depend on the production-sequence. The object of this problem is to find a feasible production plan with minimum (or close to minimum) total costs. We suppose that setup always starts at the beginning of the period. In other words, we do not allow the empty setup. Furthermore, setup is kept on over idle time, i.e. setup state is not lost after idle time. We assume that machine is initially setup for some item and shortages and back-logging are not allowed. The following notation and decision variables are used to present formulation.

[Notation]

- N the number of different items
- T the number of periods
- M big number
- C_t the capacity of machine available in period t. Machine capacity is limited on time consumption.
- d_{it} the demand for item i at the period t
- h_i holding cost which is incurred to hold one unit of item i at the end of period
- sc_{ij} the setup cost when production change from item i to j
- st_{ij} the setup time when production change from item i to j
- p_i the time which is needed to produce one unit of item i

[Decision variables]

- q_{it} the quantity of item i to be produced in period t
- I_{it} the inventory of item i at the end of period t $(I_{i0} = I_{iT} = 0)$
- z_{it} the machine state in period t: $z_{it} = 1$, if the machine is used to produce item i in period t
- x_{ijt} the setup state in period t: $x_{ijt} = 1$ if the machine is setup from product i to j in period t. 0, otherwise
- y_{ijt} the setup state between period t and period $t+1$: $y_{ijt} = 1$ if the machine is setup from product i to j at the beginning of period $t+1$. 0, otherwise

CLSPSD:

$$\min \quad \sum_{i}^{N}\sum_{t}^{T} h_i I_{it} + \sum_{i}^{N}\sum_{j}^{N}\sum_{t}^{T} sc_{ij}(x_{ijt} + y_{ijt}) \qquad (1)$$

s.t

$$q_{it} + I_{it-1} = d_{it} + I_{it} \qquad \forall i,t \qquad (2)$$

$$\sum_{i}^{N} p_i q_{it} + \sum_{i}^{N}\sum_{j}^{N} st_{ij}(x_{ijt} + y_{ijt-1}) \le C_t \qquad \forall t \qquad (3)$$

$$q_{it} \le M z_{it} \qquad \forall i,t \qquad (4)$$

$$x_{ijt} \le z_{it} \qquad \forall i,j,t(i \ne j) \qquad (5)$$

$$y_{ijt} \le z_{it} \qquad \forall i,j,t \qquad (6)$$

$$x_{jit} \le z_{it} \qquad \forall i,j,t(i \ne j) \qquad (7)$$

$$y_{jit} \le z_{it-1} \qquad \forall i,j,t \qquad (8)$$

$$\sum_{i,j \in S} x_{ijt} \le |S| - 1 \qquad S \subset N, 2 \le |S| \le N-2, \forall t \qquad (9)$$

$$x_{ijt} + x_{jit} \le 1 \qquad \forall i,j,t(i \ne j) \qquad (10)$$

$$z_{it} \le \sum_{j} x_{ijt} + \sum_{j} y_{ijt} \le 1 \qquad \forall i,j,t(i \ne j) \qquad (11)$$

$$z_{it} \le \sum_{j} x_{jit} + \sum_{j} y_{jit-1} \le 1 \qquad \forall i,j,t(i \ne j) \qquad (12)$$

$$\sum_{i}\sum_{j} y_{ijt} = 1 \qquad \forall t(t \ne T) \qquad (13)$$

$$x_{ijt}, y_{ijt}, z_{it} \in \{0,1\} \qquad \forall i,j,t \qquad (14)$$

$$q_{it,} I_{it} \ge 0 \qquad \forall i,t \qquad (15)$$

The objective function (1) minimizes the sum of inventory holding costs and sequence-dependent setup costs. Constraints (2) are the inventory balancing constraints. Constraints (3) are the capacity constraints. We already mentioned that the machine capacity is defined as available operating time. Thus, finite machine capacity is further reduced by sequence-dependent setup times. Constraints (4)

guarantee that production takes place only if the machine is setup for an item. Constraints (5), (6), (10) and (11) specify that exactly one setup among any setups from each of *N* items, *j=1,...,N*, to fixed item *i,* takes place in period t when item i is produced. Similarly, constraints (7), (8), (10) and (12) represent that just one setup among any setups from fixed item i to each of *N* items, *j=1,...,N*, occurs only if item i is produced in period *t.* Constraints (9) are subtour elimination constraints: they prevent the subtours can be generated while find a production sequence. Constraints (13) make sure that at least one item has to be produced in each period. Finally, constraints (14) and (15) impose the binary and continuous conditions on the variables, respectively.

3. The simulated annealing algorithm.

Simulated annealing algorithm was introduced by Kirkpatrick[15], for solving combinatorial optimization problems. Simulated annealing technique use an analogous cooling operation for transforming a poor, unordered solution into an ordered, desirable solution, so as to optimize the objective function. The basic idea is to choose a neighbor randomly. The neighbor then replaces the incumbent with probability 1 if it has a better objective value, and with some probability strictly between 0 and 1 if it has a worse objective value. Therefore, it is possible to escape from the traps of local optima. The following elements are considered in the simulated annealing algorithm:

■ A description of possible problem solutions, so-called *configuration*.

■ An *objective function* to measure how good any given placement configuration is.

■ A set of random changes that will permit us to reach all feasible configurations.

■ *Cooling schedule* to anneal the problem from a random solution to a good, frozen, placement.

3.1 Generation of an initial solution

To generate an initial solution, first, we decide the production lot, and determine the production sequence afterward.

Determination of an initial production lot: We fix the sequence-dependent setup times to the average setup times of all setup times, st_0, and ignore the sequence dependency. We settle the production quantity of item i in period t, q_{it}, to the demand of the item i in period t. After that, compute the total capacity used in period t. If there are periods which violate the machine capacity, we modify the current production lot. Starting from the last period, excess capacity, so called overtime, is moved to the preceding period having available capacity. We first move the item having the lowest unit inventory holding costs and only shift the amount of production lots that are needed to get rid of overtime. We repeat this procedure until overtime has been eliminated overall periods.

Scheduling the production sequence: We consider a scheduling of the production sequence in each period as TSP and determine production sequence by applying a nearest neighbor algorithm (Cook et al. [16]). Following is the detailed sequencing procedure. First, we generate all possible product pairs which are consist of two items representing the item which are produced in first and at last in a period respectively. Second, find the sequence which has minimum setup costs. Starting at first item, choose the next item which has the minimum setup costs while changing over from a previous item. Repeat this process until all items have been selected, finally reach the last item. We store the cheapest cost to the last item in

period t among those setup costs generated by all possible pairs. Between consecutive periods, select the lowest setup costs among possible setup costs. We also store the cheapest cost to the first item in next period. Repeat the previous processes in each period until reaching the last period. The cheapest accumulated setup cost out of the all setup costs at last items which are produced in last period is the total setup costs, and this sequence is the production sequence.

Feasibility check: We use the feasible solution as an initial solution. Thus, we check the periods whether violate the capacity constraints or not. If there are no periods occurring excessive amount of the machine capacity, called overtimes, current solution is an initial solution. If not, we start from the last period, and move excess production lot to the immediately preceding period having available capacity. An item which has the lowest unit inventory holding cost is shifted first. We only move the amount of the production lots that are needed to eliminate overtime. We repeat this procedure until overtime has been eliminated in all periods. This is the initial solution. And compute the total costs.

3.2 Searching for a neighboring solution.

Similar to the procedure of generating an initial solution, first we generate a neighboring production lot by moving production quantities, and then schedule the production sequence.

Generating a neighboring production lot: We use the amount of production quantities of item i which are produced in period t as a configuration for generating a neighboring production lot. Under the current production lots, we rearrange production lots and generate a neighboring production lot using following process. Before rearranging production lots, we randomly choose the period t_1 and item which is produced in period t_1. After selecting moving period t_1 and moving item, m_i, we decide how many production lots move to which period, so-called target period. The m_i can be moved to earlier or later period.

(a) When the total production quantity of m_i from period 1 to period t_1 is equal to the sum of demands of m_i from period 1 to period t_1 : Production lot is only moved to the immediately earlier period, t_2, which has an available capacity.

(b) When the total production quantity of m_i from period 1 to period t_1 is larger than the sum of demands of m_i from period 1 to period t_1 :Find the right after period having an available capacity, t_l, and check whether the production quantity from period t_1+1 to period t_l-1 can satisfy the demand from period t_1+1 to period t_l-1. If production lots satisfy the demands, the target period is t_l. If not, target period is the right earlier period which has available capacity.

Now, we decide the production lots to be moved. Let the product quantity of m_i in period t_1 is $q_{m_i t_1}$, the available capacity in target period t_2 is $slack_{t_2}$, and Δ is the production lots to be moved.

(a) If target period is earlier period:

$$\Delta = \min\{q_{it}, slack_{t_2}\}$$

(b) If target period is later period:

$$\Delta = \min\{q_{it}, slack_{t_2}, \sum_{\tau=1}^{t_1} q_{i\tau} - \sum_{\tau=1}^{t_1} d_{i\tau}\}$$

After obtaining neighboring production lot, we determine the production sequence based on neighboring production quantities and check the feasibility of neighboring solution. This procedure is same as those which are used in the procedure of generating an initial solution.

3.3 Cooling schedule

We allow to a worse solution moves restrictively by introducing the probability of acceptance, *PA*. *PA* is calculated as follows. Let *Q* is an initial solution, *Q'* is a neighboring solution, *C(Q)* is objective value of a current solution, and *C(Q')* is objective value of a neighboring solution. Annealing temperature T is reduced when the neighboring solution increase the objective value. A new temperature is $a \times T$ with reduction factor a ($0 < a < 1$).

$$PA = \exp[\frac{C(Q) - C(Q')}{C(Q) \times T}]$$

We compare *PA* with a random number and if *PA* is greater than random number, we accept a worse neighboring solution.

3.4 A stop criterion

We complete the algorithm when the whole the iteration is over the *MAX_ITER* or the number of iterations in a state of no improvement of solution value is over fixed number, MAX_ITER_a.

4. Computational results

To experiment the proposed algorithm, we ran a C-language in a PC with Pentium 4 (2.53GHz) with 512 MB RAM. We use CLPEX 9.0 to compare our heuristic solution with optimal solution. We randomly generate total 144 instances. The number of items range from 3 to 10 and the number of periods ranges from 3 to 10, and 20. We get the rest of the input data as follows:

■ Demand of item i in period t, d_{it}, is chosen out of the interval [20,60] with uniform distribution.

■ A setup time from item i to j, st_{ij} $(i \neq j)$, is randomly chosen out of the interval [2,10], and setup time within same items is zero, i.e. $st_{ii} = 0$. Setup times of a changeover from item i to j is same as the setup times of a changeover from product j to i, i.e. $st_{ij} = st_{ji}$, for all i, j. The choice of setup times is over so that all triangle inequalities are satisfied.

■ We suppose setup costs are proportional to setup times, i.e. $st_{ij} = f_{sc} \times st_{ij}$, for all i, j. The parameter f_{sc} is 50.

■ Inventory holding costs for each items, h_i, are randomly chosen out of the interval [2,10] with uniform distribution.

■ Machine capacity, C, is determined according to: $C = \{40 \times (number\ of\ items)\} \div u$

u is the capacity utilization and systematically differed from 0.4, and 0.6.

■ We set the quantity a (Boltzmann's constant) to 0.99 by testing algorithms many times and selecting the best one which gives the best result.

■ *MAX_ITER* is 2000 and MAX_ITER_a is 500.

Table 1 shows the results of our study. Let *C** is the optimal value obtained by CPLEX, and *C* is the objective value obtained by running the simulated annealing algorithm. And the GAP(%) is calculated as (C – C*)/ C*× 100. Optimal value represent the optimal solution which is obtained from using CPLEX, solution value is the solution which is obtained from solving the simulated annealing algorithm, and CPU time(sec) is running times which are needed to execute a simulated annealing algorithm. The computational results demonstrates that the average GAP is 1.39% when u=0.4 and 1.95% in case of u=0.6. These results verify that our algorithm performs well. Average running time is 34.88 sec. and 41.57 sec. respectively. If problem sizes grow bigger, getting an optimal solution using CPLEX takes long time and we cannot obtain an optimal solution within certain time. In this case, the

optimal value marked by '*'. On the other hand, our algorithm can obtain solutions in reasonable times and we expect our algorithm will also provide good results in the big size problems even though we cannot compare with the optimal solution. Finally, we compare the performance of the simulated annealing algorithm and the greedy heuristic, called ISI heuristic, which is suggested by Diwakar Gupta, Thorkell Magnusson[5]. Table 2 shows the comparison of the solutions obtained from simulated annealing and the solutions obtained from ISI heuristic. We apply the same instances which are used in simulated annealing algorithm when u is 0.6 to the ISI heuristic. Average GAP of our solution is approximately 2%, and that from ISI heuristic is approximately 8%. This result shows that our algorithm improves the solution value and provide near optimal solution.

[Table 1] Summary of the computational results

	u=0.4		u=0.6	
period	CPU time(sec)	GAP(%)	CPU time(sec)	GAP(%)
3	18.88	1.033	20.62	1.165
4	22.03	1.147	24.52	2.529
5	18.95	1.666	29.18	1.101
6	23.71	0.930	22.73	1.193
7	34.00	1.238	39.34	2.257
8	36.29	2.208	34.71	2.396
9	38.73	0.919	40.14	1.859
10	41.97	1.997	53.41	3.066
20	79.36	*	109.51	*

[Table 2] Comparison of the performance (GAP(%))

period	Simulated annealing	ISI Heuristic
3	1.165	6.744
4	2.529	10.907
5	1.101	6.162
6	1.193	6.928
7	2.257	7.910
8	2.396	8.003
9	1.859	8.304
10	3.066	10.365
total	1.947	8.165

5. Concluding remarks

In this paper, we reformulate the CLSPSD and have introduced the simulated annealing algorithm to solve CLSPSD. It is shown that the simulated annealing algorithm performs well and gives good feasible solution values in reasonable times. The overall average Gap is approximately 2%.

Even though simulated annealing methods provide good performance, its computational times are slower than the running time through CPLEX when problem sizes are small. Concerning further researches, it is still possible to improve computational time by reforming the procedure of generating neighboring solutions. Moreover, it could be of concern to research the chance of improving sequencing procedure. As based on the formulation suggested in this paper, it will be possible to obtain optimal solution or lower bound using the column generation method.

References

[1] A. Drexl, A.Kimms (1997), "Lot sizing and scheduling-Survey and extensions", European Journal of Operational Research 99, 221-235.

[2] Salomon M, Solomon MM, Van Wassenhove LN, Duman Y, Duazere-Peres S (1997), "Solving the discrete lot-sizing and scheduling problem with sequence-dependent setup costs and set-up times using the Travelling Salesman Problem with time windows", European Journal of Operational Research 100(3), 494-513.

[3] K.Haase (1996), "Capacitated lot-sizing with sequence-dependent setup costs", OR Specktrum 18, 51-59

[4] Sungmin Kang, Kavindra Malik, L.Joseph Thomas (1999), "Lotsizing and scheduling on parallel machines with sequence-dependent setup costs", Management science 45(2), 273-289.

[5] Diwakar Gupta, Thorkell Magnusson (2005), "The Capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times", Computers & Operations research 32,727-747.

[6] Aras, O.A., Swanson, L.A. (1982), "A lot sizing and sequencing algorithm for dynamic demands upon a single facility", journal of Operations Management 2(3), 177-185

[7] Trigeiro WW, Thomas LJ, McClain JO (1989), "Capacitated lot sizing with setup times", Management Science 35(3), 353-366.

[8] M. Diaby, H.C. Bahl, M.H.Karwan, S.Zionts (1992), "Capacitated lot-sizing and scheduling by lagrangean relaxation", European Journal of Operational Research 59,444-458.

[9] Garey,M. and D. Johnson (1979), Computers and intractability: A guide to the theory of NP-completeness, Freeman and Co., San Francisco.

[10] Barany I., Van Roy TJ, Wolsey L. (1984), "Strong formulations for multi-item capacitated lot sizing", Management Science 30(10), 1255-1261.

[11] Eppen,G.D.,Martin, R.K.(1987), " Solving multi-item capacitated lot-sizing problem using variable redefinition", Operations Research 35,832-848.

[12] Kuik, R., Salomon, M. (1990) "Multi-level lot-sizing problem: Evaluation of a simulated-annealing heuristic", European Journal of Operational Research 45(1), 25-37

[13] Mohan Gopalakrishnan, Ke Ding, Jean-Marie Bourijolly, Srimathy Mohan (2001), "A Tabu-Search Heuristic for the capapcitated lot-sizing problem with set-up carryover", Management science 47(6), 851-863.

[14] Ou Tang (2004), "Simulated annealing in lot sizing problmes", Int. J. Production Economics 88,173-181.

[15] S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vecchi (1983), "Optimization by simulated annealing", Management Science 220, 671-680

[16] William J. Cook, William H. Cunningham, William R. Pulleyblank, Alexander Schrijver, "Combinatorial Optimization".