Industrial Engineering
& Management Systems
Vol 11, No 3, September 2012, pp.288-298
ISSN 1598-7248 │EISSN 2234-6473 │

http://dx.doi.org/10.7232/iems.2012.11.3.288
© 2012 KIIE

Meta-Heuristic Algorithms for a Multi-Product Dynamic Lot-Sizing Problem with a Freight Container Cost

Byung Soo Kim
Graduate School of Management of Technology, Pukyong National University, Busan, Korea

Woon-Seek Lee*
Department of Systems Management and Engineering, Pukyong National University, Busan, Korea

(Received: May 20, 2012 / Revised: August 2, 2012 / Accepted: August 3, 2012)

ABSTRACT

Lot sizing and shipment scheduling are two interrelated decisions made by a manufacturing plant and a third-party logistics distribution center. This paper analyzes a dynamic inbound ordering problem and shipment problem with a freight container cost, in which the order size of multiple products and single container type are simultaneously considered. In the problem, each ordered product placed in a period is immediately shipped by some freight containers in the period, and the total freight cost is proportional to the number of containers employed. It is assumed that the load size of each product is equal and backlogging is not allowed. The objective of this study is to simultaneously determine the lot-sizes and the shipment schedule that minimize the total costs, which consist of production cost, inventory holding cost, and freight cost. Because the problem is NP-hard, we propose three meta-heuristic algorithms: a simulated annealing algorithm, a genetic algorithm, and a new population-based evolutionary meta-heuristic called self-evolution algorithm. The performance of the meta-heuristic algorithms is compared with a local search heuristic proposed by the previous paper in terms of the average deviation from the optimal solution in small size problems and the average deviation from the best one among the replications of the meta-heuristic algorithms in large size problems.

Keywords: Dynamic Lot-Sizing, Shipment Scheduling, Multi-Products, Meta-Heuristics

* Corresponding Author, E-mail: iewslee@pknu.ac.kr

1. INTRODUCTION

Most manufacturing companies do not have a specialty of logistics functions. In order to obtain the benefit of shipping and warehousing costs, they strategically ally with a specialized third-party logistics (TPL) company for offering a wide range of services including: order fulfillment, inbound freight, warehousing, freight consolidation, outbound dispatching. They outsource the logistics operations to TPL company and concentrate on their core manufacturing operation.

Among the services provided by a TPL company, this paper focuses on the most important process is replenishing the multiple products by a number of freight containers at right time from manufacturers to the TPL warehouse. This process leads to managerial decision problems in determining replenishment lot-sizes and shipment schedule for each demand. For these problems the freight cost as well as the ordering and holding cost are considered. Since the replenishment orders are assumed to be shipped in containers to the warehouse, the freight cost is assumed to be proportional to the number of containers used. To solve this problem, this paper proposes an integrated optimization model of inbound ordering and outbound dispatching for single product in a TPL company.

The demand in this paper is assumed to be known, but dynamic over a discrete and finite time horizon. The production-inventory problem with the demand is called the dynamic lot-sizing model (DLSM), which has stemmed from the work of Wagner and Whitin (1958). Though there are a number of research results dealing with the DLSM, the majority of DLSMs have not considered any production-inventory problem incorporating shipment problem as well. These days, the issue of shipment scheduling for ordered products (or delivering orders) by a proper shipping freight container mode at right time becomes significantly important in production and distribution management because of the increasing fuel price.

Several studies have focused on various general costs and the capacitated resources as the extended works of the classical DLSM. Lippman (1969) studied two deterministic multi-period production planning models; monotone cost model and concave model. Florian *et al.* (1980) and Bitran and Yanasse (1982) proved the general capacitated single product lot sizing problem is NP-hard. Hwang and Sohn (1985) dealt with a DLSM in which the transportation mode and the order size for a deteriorating product are simultaneously considered. However, they considered no capacity restriction on the transportation mode. Lee (1989) considered a DLSM allowing multiple set-up costs including a fixed charge cost and a freight cost, where a fixed single container type with limited carrying capacity is considered and the freight cost is proportional to the number of containers used. Fumero and Vercellis (1999) proposed an integrated optimization model for production and distribution planning considering such operational decisions as capacity management, inventory allocation, and vehicle routing. The solution of the integrated optimization model was obtained using the Lagrangian relaxation technique. Lee *et al.* (2003) extended the works of Lee (1989) by considering multiple heterogeneous vehicle types to immediately transport the finished product in the same period it is produced. It is also assumed that each vehicle has a type-dependent carrying capacity and the unit freight cost for each vehicle type is dependent on the carrying capacity. Lee *et al.* (2003) considered a dynamic model for inventory lot-sizing and outbound shipment scheduling in the third-party warehousing domain. They presented a polynomial time algorithm for computing the optimal solution. Jaruphongsa *et al.* (2005) analyzed a dynamic lot-sizing model in which replenishment orders may be delivered by multiple shipment modes with different lead times and cost functions.

Anily and Tzur (2005) considered a dynamic model of shipping multiple items by capacitated vehicles. They presented an algorithm based on a dynamic programming approach. Van Norden and van de Velde (2005) dealt with a multiple product problem of determining transportation lot-sizes in which the transportation cost function has piece-wise linear as to a transportation capacity reservation contract. They proposed a Lagrangian relaxation algorithm to compute lower and upper bounds. Lee *et al.* (2005) proposed a heuristic algorithm for a dynamic lot-sizing and shipping problem, in which the order size of multiple products and a single container type are simultaneously considered. Kim and Lee (2012) are proposed a tight lower-bound using the shortest reformulation model compared to the solution of the upper-bound of the heuristic algorithm proposed by Lee *et al.* (2005). The limitation of their studies is that the performance is not successive in large size problems because the proposed solution approach is a local search heuristic.

In this paper, we propose three meta-heuristic algorithms to define the multi-product lot-size problem and shipment scheduling that minimize the total costs, which consist of ordering cost, inventory holding cost, and freight cost by generalizing the basic model of Kim and Lee (2012). The proposed heuristic algorithms are the first study in simultaneously determining multi-product lot-sizing problem and the shipment scheduling.

This paper is organized as follows. In the next section, the mathematical model of the problem is described. In Section 3, the properties of the optimal solution are characterized. Three meta-heuristic algorithms based on the optimal solution properties are described in Section 4. The computational results from a set of simulation experiments are presented in Section 5.

2. PROBLEM DESCRIPTION

The following notations are defined to formulate the problem:

Parameters

TH	= the set of discrete time horizon, ($t = 1, 2, \cdots, T$)
PD	= the set of products, ($i = 1, 2, \cdots, M$)
d_{ti}	= amount demanded for product i in period t,
w_i	= volume of product i,
W	= carrying capacity of a container,
S_t	= ordering cost of product i in period t,
h_{ti}	= unit inventory holding cost of product i from period t to period t+1,
F	= unit freight cost of container,

Variables

x_{ti}	= amount of product i ordered and shipped by container in period t,
z_{ti}	= 1, if an order is incurred for product i in period t, and 0, otherwise,
y_t	= number of containers used in period t (non-negative integer),
I_{ti}	= inventory level of product i in period t

The objective of the problem is to determine (x_{ti}, y_t) for $\forall i \in TH$ and $t \in TH$ so that all demands over the given horizon are satisfied at the minimum total cost. Therefore, the *T*-period problem (*P*) can be formulated in a mathematical programming as follows:

$$(P)\ Minimize\ \sum_{t \in TH}\left\{\sum_{i \in PD} S_i \cdot z_{ti} + \sum_{i \in PD} h_{ti} \cdot I_{ti} + F \cdot y_t\right\}$$

Subject to

$$I_{ti} = I_{t-1i} + x_{ti} - d_{ti}, \qquad \text{for } \forall i \in PD,\ t \in TH \qquad (1)$$

$$\sum_{i \in PD} w_i x_{ti} \le W \cdot y_t, \qquad \text{for } \forall t \in TH \qquad (2)$$

$$x_{ti} \le M \cdot z_{ti}, \qquad \text{for } \forall i \in PD,\ t \in TH \qquad (3)$$

$$I_{0i} = I_{Ti} = 0, \qquad \text{for } \forall i \in PD \qquad (4)$$

$$x_{ti}, I_{ti} \ge 0, \qquad \text{for } \forall i \in PD,\ t \in TH \qquad (5)$$

$$y_t \text{ is non-negative integer} \qquad \text{for } \forall t \in TH \qquad (6)$$

If we assume that all products have the same volume, the model (*P*) can be specialized to (*P1*) proposed by Kim and Lee (2012).

$$(P1)\ Minimize\ \sum_{t \in TH}\left\{\sum_{i \in PD} S_i \cdot z_{ti} + \sum_{i \in PD} h_{ti} \cdot I_{ti} + F \cdot y_t\right\}$$

Subject to $\sum_{i \in PD} w_i x_{ti} \le W \cdot y_t$, for $\forall t \in TH$ (7)

and (1), (3), (4), (5), and (6)

Constraint (2) ensures that the inventory level of each product in the current period balances the inventory level in the previous period, the ordering amount and the demand in the current period. Constraints (3) and (7) imply that the total ordering amount is restricted by the total carrying capacity associated with the number of containers used in the period. Constraint (4) ensures the relation between x_{ti} and z_{ti}. The constraints in the above model (*P*) and (*P1*) define a closed bounded convex set and the objective function is concave, so that the problem attains its minimum at an extreme point of the convex set. The extreme points will be further characterized in association with the optimal solution in the next section.

3. OPTIMAL SOLUTION PROPERTIES

The mathematical model (*P*) can be represented by a network model as Figure 1. In the network, two flow types are defined as follows. 1) The aggregate flow is defined as the flow between node 0 and nodes (t = 1, 2, ⋯, T). 2) The individual flow is defined as the flow between nodes (t = 1, 2, ⋯, T) and nodes ((1, 1), (1, 2), ⋯, (T, M)).

Here, the arcs in the aggregate flow are restricted by the capacities associated with the number of containers used whereas the arcs in the individual flow are not.

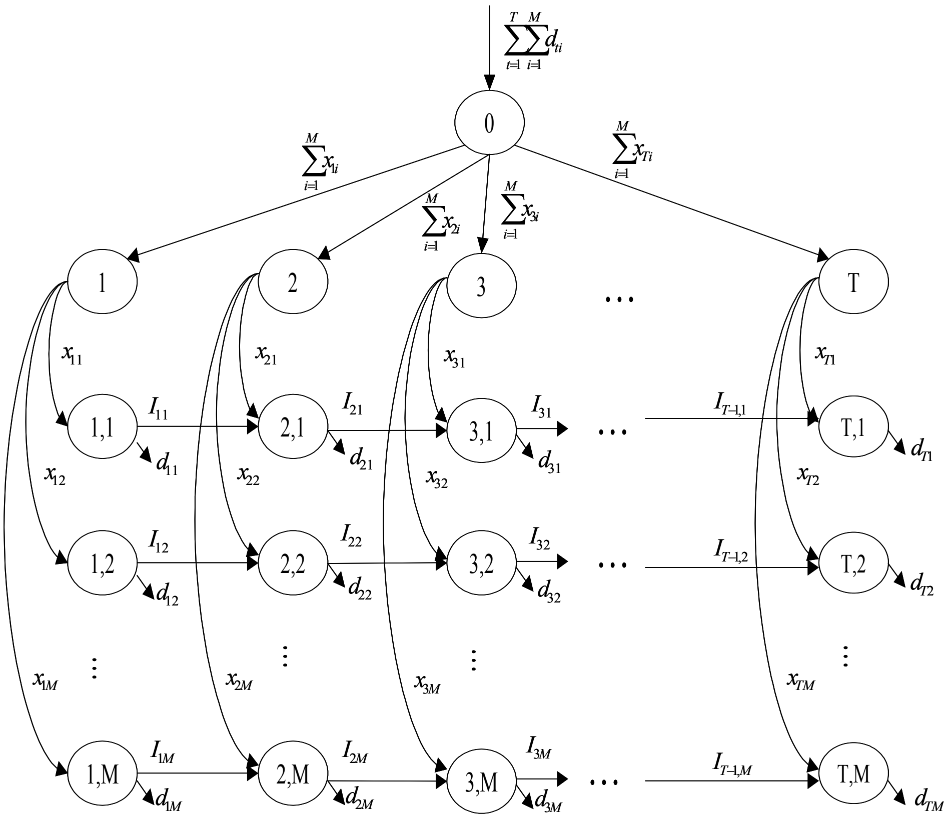

Figure 1. Network representation of the model *P1*.

The optimal solution of the model (*P*) occurs at extreme points. In a network theory, such an extreme point can be interpreted as an extreme flow (Florin *et al.*, 1980; Zangwill, 1969). In a network without arc capacities, a feasible flow is an extreme flow if it does not have a positive loop. Also, in a network with arc capacities, a feasible flow is an extreme flow if and only if each loop has at least one saturated arc.

In Figure 1, loops can be formed by two cases as follows. 1) Between the aggregate flow and the individual flow, for example, the loop can be formed by the sequences of nodes (0), (1), (1, 1), (2, 1), (2) and (0); 2) On the individual flows, for example, the loop can be formed by the sequences of nodes (1, 1), (1), (1, M), (2, M), (2), (2, 1), and (1, 1).

Ordering and shipment schedule satisfying the optimal solution property by Wagner and Whitin (1958), $x_{ti} \cdot I_{ti} = 0,$ is no longer the extreme flow. Such policies may form positive loops between the aggregate flows. The properties of Theorems 1 and 2 must be satisfied so as to have an extreme flow. To examine the properties, the production point, the partial aggregate point, and the inventory point are defined, respectively as follows. 1) Period *t* is a ordering point for product *i* if $x_{ti} \geq 0$; 2) Container type *j* is a partial aggregate point in period *t* if $n \cdot W < \sum_{i \in PD} w_i x_{ti} < (n+1) \cdot W$ (*n* is a non-negative integer); 3) Period *t* is an inventory point for product i if $I_{ti} = 0.$

Theorem 1: *In the model P, the optimal solution has at most one partial aggregate point for product i between two consecutive inventory points for product i.*

Proof: Suppose that there exists the optimal solution which has two partial aggregate point between two consecutive inventory points for product *i*. In the network of Figure 1, this case can have the loop formed by the sequences of nodes (0), (1), (1, 1), (2, 1), (2), and (0). The condition that this feasible flow will be the extreme flow is that at least one of the arcs (0, 1) and (0, 2) must be saturated. This feasible flow is not an extreme flow. Therefore, the proof is completed.

Theorem 2: *In the model P, the optimal flow must do not form the positive loop in the individual flow.*

Proof: Suppose that there exists a feasible flow that satisfies the properties of Theorems 1 and 2, which has a loop formed by the sequences of nodes (1, 1), (1), (1, 2), (2, 2), (2), (2, 3), (3, 3), (3), (3, 1), (2, 1) and (1, 1) in Figure 1. Because arcs in the individual flow are not capacitated, there exists a positive loop formed by unsaturated arcs. This feasible flow is not an extreme flow. Therefore, the proof is completed.

Van Norden and van de Velde (2005) proved that the single level non-capacitated multi-product multi-period lot-sizing with transportation cost model is NP-hard in the strong sense. Hence, the problem (*P*) is NP-hard. So, it is not easy to make optimization viable for large problems. In the next section, three meta-heuristic algorithms are presented based on the properties of Theorems 1 and 2.

4. META-HEURISTIC ALGORITHMS

In this section, we propose three meta-heuristics: a simulated annealing (SA) algorithm, a genetic algorithm (GA), and a self-evolution algorithm (SEA) based on the properties of Theorems 1 and 2.

4.1 Solution Representation

For the solution representation of three meta-heuristic algorithms, a solution is defined as two dimensional array of (*T*, *M*) of 0-1 genes as follows:

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{e}_{11} & \cdots & \boldsymbol{e}_{T1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{e}_{1M} & \cdots & \boldsymbol{e}_{TM} \end{bmatrix} \tag{8}$$

In this representation, e_{ti} can take a value 0, 1, 2 which is the key to find the value of x_{ti}. In this situation, the value of e_{ti} directly decides the value of x_{ti}. Then depending variable sets z_{ti}, l_{ti}, and y_t can be calculated, once the value of x_{ti} is determined by the following procedure:

Step 1: Set $i = 0$.
Step 2: Let *p* be the earliest period of product *i* after *t* that has nonzero e_{ti}. If all the period after *t* have zero values for e_{pi}, let *p* be *T*+1
Step 3: Set $X = \sum_{r=1}^{p-1} \sum_{r=1}^{p-1} d_{rk} - l_{t-1i}$
Step 4: If $e_{ti} = 1$, $x_{ti} = X$ and $z_{ti} = 1$. On the other hands, else if $e_{ti} = 2$, $x_{ti} = W\lceil X/W \rceil$ and $z_{ti} = 1$, otherwise $x_{ti} = z_{ti} = 0$
Step 5: If $i = M$, determine $y_t = \left\lceil \sum_{i \in PD} d_{rk} - I_{t-1i} \right\rceil$ then STOP. Otherwise, $i = i+1$

Property 1: For the model *P1*, the variable set x_{ti}, are determined, the other decision variable sets z_{ti}, l_{ti}, and y_t can be determined.

In this representation, one can easily find that the first nonzero value of $e_{ti}, i \in PD$ must be no later than the first positive value of d_{ti} to meet the constraint (4), i.e., the constraint of no-backlogging. Thus, a repairing process is required after genetic operations of GA and SEA and perturbations of SA.

4.2 Genetic Algorithm

The GA, which has been widely used in solving various optimization problems for the last three decades, is a stochastic search algorithm based on the mechanism of natural selection and natural genetics. Being different from conventional search techniques, it starts with an initial set of (random) solutions called a population. Each individual in the population is called a chromosome, representing a solution to the problem at hand. The chromosomes are evaluated, using some measures of fitness. Generally speaking, the GA is applied to spaces that are too large to be exhaustively searched (Goldberg, 1989). In this paper, the chromosomes that have higher fitness value (lower objective function value) than the average fitness of current population make a potential parent pool and new chromosomes in the next generation are randomly selected from the parent pool or selected from the roulette wheel method.

Two kinds of crossover operators are randomly used: partially-matched crossover (PMX) and uniform crossover (UX). Eight kinds of mutation operators are randomly used: the eight kinds of mutation operators (pull operator, swap operator, insert operator, inner random operator, outer random operator, uniform random operator, single change operator, and rest change operator) to make a new chromosome. For the operators, one, two or multiple points in the selected original chromosome are randomly selected and mutated. The pull operator is illustrated in Figure 2a. The genes on right side of point 2 (including point 2) are pulled to the position of point 1, and the genes between point 1 and 2 (including point 1) are placed after. The two genes at the points are interchanged for swap operator as shown in Figure 2b. Insert operator simply insert the gene at point 2 into the position of point 1 as shown in Figure 2c. Inner random operator and outer random operator are illustrated in Figure 2d and e. The inner or outer genes of point 1 and 2 are randomly replaced for the operators. Uniform random operator randomly select multiple points and change as shown in Figure 2f. Figure 2g and h describe single change and other change in simple way. Using the crossover operators, the mutation operators, and the selection operators, the selected parents reproduce new

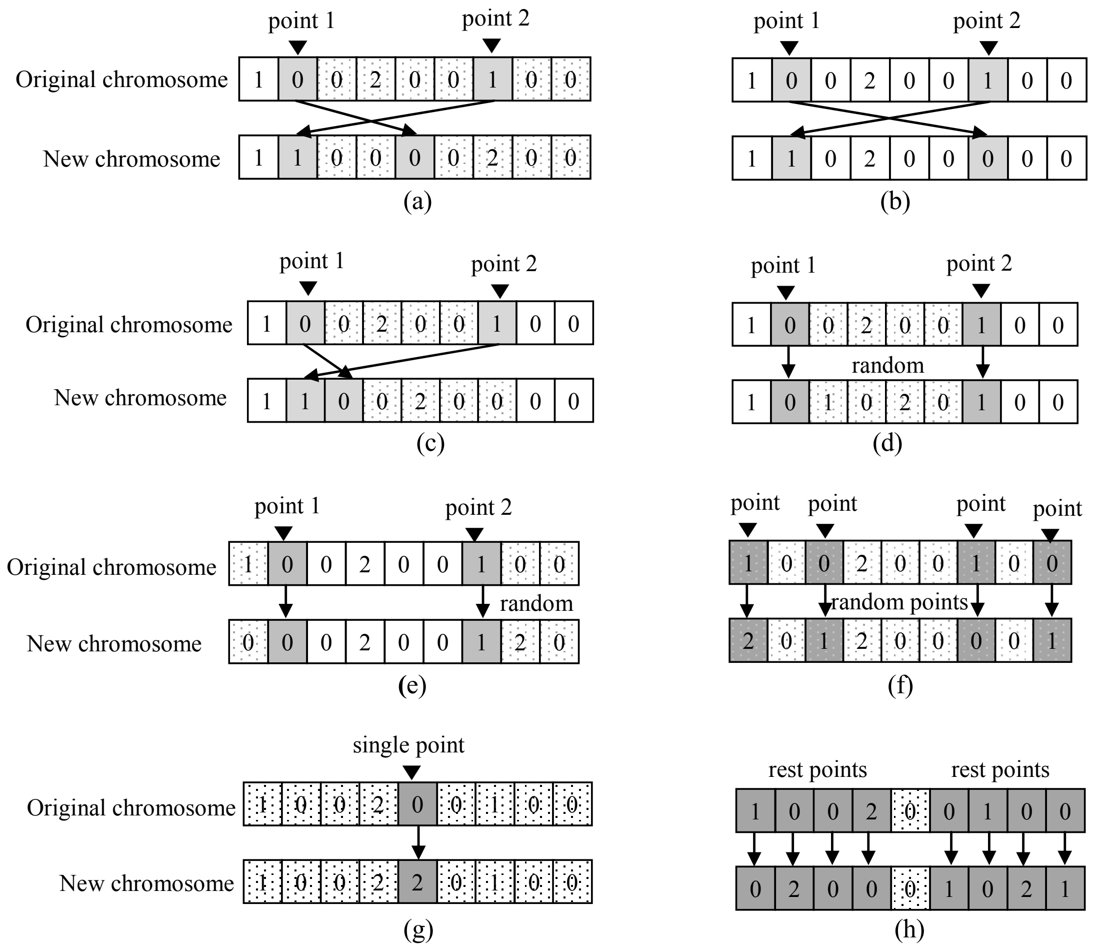

Figure 2. Mutation operators of genetic algorithm: (a) pull, (b) swap, (c) insert, (d) inner random, (e) outer random, (f) uniform random, (g) single change, and (h) rest change operator.

chromosomes (i.e., children) to generate a population for the next generation. The various parameters of the GA heuristic are summarized in Table 1. These parameters were selected based on extensive preliminary experimentations to the best combination with highest performance. GA evaluates a total of 1,000×(T+M+N) of fitness function.

Table 1. Parameters of the genetic algorithm heuristics

Parameter	Value/type
Population size	T+M+N
Number of generations	1,000
Crossover rate	0.8
Mutation rate	0.2

T: the size of planning horizon, M: the number of items, N: F/100

The following procedure outlines the pseudo code of the GA:

```
Procedure GA for multi-product lot-sizing problem with a freight container
 Begin
   Set crossover rate and mutation rate as $r_c = 0.8$ and $r_m = 0.2$
   Set a generation size as $g = 0$
   Generate an initial population $P_c$
   Evaluate $f(\cdot)$ all chromosomes in $P_c$
   Find the best chromosome $s_{cbest}$ in $P_c$
   Let $P_{best} = s_{cbest}$ and $f_{best} = f(s_{cbest})$
   Repeat (the generation loop)
     Repeat (the population loop)
       Randomly select two parents $s_{c1}$ and $s_{c2}$ among the population $P_c$ using roulette wheel selection
       Generate two new children $s_{n1}$ and $s_{n2}$ by following genetic operations:
         If (random [0, 1] < $r_c$)
         Then perform randomly PMX crossover or UX crossover
         If (random [0, 1] < $r_m$)
         Then perform randomly a mutation among 8 mutations
     Until (number of populations reaches a maximum of T+M+N)
     Evaluate $f(\cdot)$ in $P_n$
     Update $P_c$ by selecting an elitist and the rest of chromosomes in the population using randomly roulette wheel selection or random selection within a good solution pool from $P_n$
     If $f_{best} > f(s_{cbest})$
     Then
       $P_{best} > s_{cbest}$ and $f_{best} > f(s_{cbest})$ in the population $P_c$
   Until (number of generation reaches a maximum of 1,000 times)
 End
```

4.3 Simulated Annealing

SA was first proposed by Kirkpatrick *et al.* (1983) to solve combinatorial optimization problems. SA explores the solution space by successive moves from one potential solution to another that is a variation of its predecessor. At the start of the process, an initial solution is generated and evaluated based on the total cost (fitness) in Section 2. SA differs from local search algorithms in that the procedure uses random selection and will sometimes accept non-improving moves (interchanges) hoping to expand the search space and ultimately reach a better overall solution. The non-improving moves are probabilistically performed using the Boltzmann probability mass function as follows (Wolsey, 1998):

$$e(t) = e^{-(\Delta f / t)}, \tag{9}$$

where t is the current temperature, $\Delta f = f(P_n) - f(P_c)$ and p_n and p_c are the candidate fitness and the current fitness values, respectively.

In the implementation of this paper, the SA heuristic starts with a randomly generated chromosome and is controlled using two loops: an outer loop and an inner loop. The outer loop controls the temperature reduction according to the cooling schedule. In the inner loop, the temperature is held constant and a predetermined number of explorations are made. In the outer loop, explorations are made at each temperature. For fair comparison with other heuristics, the number of inner loops is 50 and the number of outer loops is 20, and the number of perturbations consists of (T+M+N) using random selection of the eight mutations in Figure 2 to increase the chances of obtaining a better solution. This leads a total of 1,000×(T+M+N) evaluations. This high number of perturbations was motivated by Barbarosoglu and Ozdamar (2000) who indicated that the increase in the number of search moves is carried out by SA significantly improves its performance. The various parameters of the SA heuristic are summarized in Table 2. These parameters were selected based on extensive preliminary experimentations to determine the best combination that leads to the highest frequency of hitting the optimal solution.

Table 2. Parameters of the simulated annealing heuristics

Parameter	Value/type
Initial temperature	1,000
Cooling schedule	Logarithmic
Size of outer loop	20
Size of inner loop	50
Number of perturbations	T+M+N

T: the size of planning horizon, M: the number of items, N: F/100

The following outlines the outlines pseudo code of the SA algorithm:

```
Procedure SA for multi-product lot-sizing problem with a freight container
  Begin
    $t = 0$
    $\theta_0 = 1,000$
    Randomly select a search point $P_c$
    Evaluate $f(P_c)$
    Let $P_{best} = P_c$ and $f_{best} = f(P_c)$
    Repeat (the outer cooling loop)
      Repeat (the inner cooling loop)
        Repeat
          Apply a specific perturbation on $P_c$ to produce $P_n$
          Evaluate $f(P_n)$ with a local search heuristic
            If $f(P_n) < f(P_c)$
            Then
              $P_c = P_n$
              If $f_{best} > f(P_n)$
              Then
                $P_{best} > P_n$ and $f_{best} > f(P_c)$
              Else if random [0, 1] < $e^{(f(P_c)-f(P_n))/\theta_t}$
              Then
                $P_c = P_n$
        Until (($T$+$M$+$N$) random trials are exhausted among 8 perturbations)
      Until (number of iterations reaches a maximum of 50 times)
      t = t+1
      $\theta_t = \frac{(t-1)\theta_{t-1}+\theta_0}{t(1+log\,t)}$
    Until (t reaches a maximum of 20 times)
  End
```

4.4 Self-Evolution Algorithm

SEA is a meta-heuristic algorithm which has a population (a set of solutions) based mechanism that uses the evolution of a solution by itself (self-evolution). Similar to GA, the set of chromosomes forms a population. Initial population is generated randomly, and the chromosomes in the population are evaluated by the measure of the fitness introduced in Section 2. A chromosome from the population is randomly selected and executes a self-reproduction using a random selection of one of mutation operators described in Section 4.2. Then the new chromosome is evaluated and it replaces the original chromosome, if the fitness value of the new chromosome is better than that of the original chromosome. The algorithm continues until the number of self-reproductions becomes a predetermined stopping value.

We propose the operators used in the eight mutations in GA. SEA is running without providing any parameters for the algorithm, because all the selection processes in SEA, such as selection of chromosome from the population for self-evolution, selection of evolution operator, and selection of points for the operator are randomly executed. SEA showed good performance for a machine scheduling problem compared with any other meta-heuristics (Joo and Kim, 2012).

For fair comparison with other heuristics, the number of generations is fixed as 1,000 for terminating, and the number of evolution operators consists of (T+M+N) using random selection of the eight mutations in Figure 2 to increase the chances of obtaining a better solution. This leads a total of 1,000×(T+M+N) evaluations with the same number of evaluations of GA and SA. The various parameters of the SEA heuristic are summarized in Table 3. These parameters were selected based on extensive preliminary experimentations to determine the best combination that leads to the highest frequency of hitting the optimal solution.

Table 3. Parameters of the self-evolution algorithm heuristics

Parameter	Value/type
Operator size	T+M+N
Number of generations	1,000

T: the size of planning horizon, M: the number of items, N: F/100

The following procedure outlines the pseudo code of the SEA:

```
Procedure SEA for multi-product lot-sizing problem with a freight container
  Begin
    Set a generation size as $g = 0$
    Generate an initial population $P_c$
    Evaluate $f(P_c)$
    Find the best solution $s_{cbest}$ in $P_c$
    Let $s_{best} = s_{cbest}$ and $f_{best} = f(s_{cbest})$
    Repeat (the generation loop)
      Randomly select a solution $s_c$ among the population $P_c$ using roulette wheel selection
      Apply a specific perturbation $s_c$ in the incumbent population $P_c$ to produce $s_n$ for the population $P_n$ of the next generation using randomly selected the perturbation operations a mutation among 8 mutations
      Evaluate $f(s_n)$
      If $f(s_n) < f(s_c)$
      Then
        $s_c = s_n$
      If $f_{best} = f(s_n)$
      Then
        $P_{best} = s_n$ and $f_{best} = f(s_n)$
    Until (number of generation reaches a maximum of 1,000×($T$+$M$+$N$) times)
  End
```

5. COMPUTATIONAL RESULTS

To analyze the performance of the proposed heuristic algorithm, the following experimental conditions were designed.

- Set M = 3, 6, 10 and T = 4 and 8 in small sized problems and T = 15, 18, and 24 in large sized problems
- Demands were generated from a normal distribution, $N(\mu_i, \sigma_i^2)$.
- Mean μ_i was generated from an uniform distribution, U(25, 100).
- Standard deviation σ_i was equally likely selected from μ_i or $\mu_i/5$.
- Setup cost was selected by $S_i = TS_i^2/2$ and TS_i = 1, 3, 6, where TS_i denotes EOQ time supply.
- h_{ti} = 1 was assumed without loss of generality.
- Set w_i were generated from an uniform distribution, U(0, 5)
- W = 100, 200, 300 and respective unit freight cost was proportionally selected by W as follows:

$$F = i \cdot W, i = 1, 3, 6.$$

To evaluate the performance of the heuristic, the C# computer code for the proposed heuristic was run on an Intel Core™2 Duo CPU with 2.00 GHz RAM. Also, CPLEX package for finding the optimal solution was run on the same computer and run so as to find the best solution within 700,000 node limits.

For the performance of the heuristics, the relative percentage deviation (RPD) calculated with the expression (10) and the computing time by GA, SA, and SEA

Table 4. RPD and CPU times in small sized problems

			$T = 6$				$T = 8$			
			RPD (%)		CPU time		RPD (%)		CPU time	
M	W	F	Heu	SEA	CPLEX	SEA	Heu	SEA	CPLEX	SEA
3	100	100	1.36	0.53	0.11	0.08	5.18	2.46	0.28	0.19
		300	1.79	0.57	0.13	0.08	4.72	1.78	0.54	0.19
		600	2.81	0.03	0.38	0.08	5.42	1.52	1.52	0.19
	200	200	2.07	1.04	0.13	0.08	4.23	2.41	0.36	0.19
		600	3.08	2.62	0.16	0.08	8.08	3.27	0.82	0.20
		1,200	8.33	0.12	0.28	0.08	8.84	4.53	1.61	0.19
	300	300	3.61	1.94	0.10	0.08	7.78	3.86	0.32	0.19
		900	4.87	2.93	0.09	0.08	8.83	4.49	0.36	0.39
		1,800	7.05	0.17	0.09	0.18	9.70	3.83	0.88	0.39
6	100	100	0.62	0.87	0.35	0.20	1.46	1.49	11.85	0.39
		300	1.29	0.93	0.94	0.19	2.66	1.21	24.95	0.40
		600	2.23	0.57	6.12	0.20	3.73	2.80	285.03	0.40
	200	200	2.00	0.26	0.36	0.18	3.21	2.12	7.64	0.40
		600	4.39	0.75	0.86	0.18	6.21	2.94	15.51	0.39
		1,200	5.49	0.31	1.46	0.18	7.06	4.16	62.29	0.39
	300	300	4.16	1.58	0.40	0.18	4.17	3.93	11.08	0.39
		900	6.50	0.55	0.59	0.18	7.25	4.22	27.83	0.76
		1,800	9.97	0.85	0.96	0.36	8.87	3.71	65.28	0.76
10	100	100	0.64	1.98	6.11	0.36	1.24	1.03	514.95	0.76
		300	1.27	1.93	27.28	0.36	2.07	1.28	866.55	0.76
		600	2.32	1.20	202.82	0.37	1.95	2.13	927.10	0.77
	200	200	2.27	1.48	4.30	0.37	2.61	3.77	479.79	0.76
		600	4.92	1.89	166.46	0.36	4.19	2.32	1,031.75	0.77
		1,200	4.99	1.42	182.15	0.37	4.49	2.42	1,064.13	0.78
	300	300	1.24	1.17	6.41	0.37	2.94	2.29	566.48	0.78
		900	0.81	0.38	174.72	0.37	4.11	3.06	673.24	0.39
		1,800	1.76	1.09	216.53	0.19	8.31	3.55	707.20	0.39
		Avg.	3.40	1.08	37.05	0.21	5.16	2.84	272.20	0.34

RPD: relative percentage deviation, Heu: RPD taken by the local search heuristic algorithm by Kim and Lee (2012), CPLEX: computational time taken by the local search heuristic algorithm by Kim and Lee (2012), SEA: self-evolution algorithm, T: the size of planning horizon, M: the number of items, W: capacity of a container, F: unit freight cost of container.

of 10 replications for each test problem.

$$\frac{z_H - z_B}{z_B} \times 100, \qquad (10)$$

Where z_B = objective value of the best solution by CPLEX in small sized test or the objective value of the best solution among the solutions of replications of GA, SA, and SEA and z_H = the average objective value of the heuristic.

In small sized problems, four instances were made for each combination of input parameters and 10 replicates for each instance are performed in the meta-heuristics (GA, SA, and SEA). In Table 4, we compare SEA shown in the best performance among SA and GA with the local search heuristic (Heu) proposed by Kim and Lee (2012). In the second column in the Table 4, SEA shown in the best performance offers on the no more than 5% from the optimal solution with small computational time for small sized test problems. The difference of RPDs between SEA and Heu increases as *T* (the size of planning horizon) and *M* (the number of items) increases, because Heu is easy to converge local optima as the problem size increases. The third and fourth columns in Table 4 show the average computing times taken by CPLEX and SEA that are measured in seconds. From the table, it is shown that SEA offers a good solution recorded in 0.34 seconds in average sense. Meanwhile, the computing time of CPLEX dramatically increases as *T* (the size of planning horizon) and *M* (the number of items).

In large sized problems, the absolute performance between CPLEX and the best solution of the heuristics (ARPD) is presented with the expression (11) and the relative performance of the heuristics (SA, GA, and SEA) is compared by the best value of the heuristics are shown in Table 5.

Table 5. Absolute RPDs between CPLEX and SEA and relative RPDs among SA, GA, and SEA in large sized problems

			T = 15				T =18				T = 24			
M	W	F	ARPD	SA	GA	SEA	ARPD	SA	GA	SEA	ARPD	SA	GA	SEA
3	100	100	6.12	15.29	13.00	1.05	9.50	12.27	17.60	1.62	12.21	7.94	10.18	0.67
		300	5.85	8.22	7.86	1.07	4.68	6.53	7.30	0.93	5.75	10.43	11.08	1.57
		600	5.99	5.28	7.30	1.52	7.09	3.46	7.25	1.41	9.13	9.20	5.73	0.61
	200	200	10.70	7.89	17.54	2.06	11.94	11.59	13.18	1.37	12.01	8.07	12.02	1.08
		600	8.02	5.35	10.13	1.27	14.02	8.84	11.74	1.75	20.54	5.99	11.14	1.57
		1,200	6.14	4.90	9.04	1.11	6.20	5.35	8.22	1.27	7.88	5.60	9.43	1.35
	300	300	10.01	17.04	19.57	2.69	10.65	21.14	19.93	3.17	NA	9.96	14.65	1.44
		900	8.63	5.61	9.32	1.08	17.35	5.63	9.29	1.75	23.25	8.41	11.03	1.83
		1,800	13.89	10.91	9.89	0.80	12.81	13.46	12.01	1.13	NA	5.46	10.38	0.95
6	100	100	9.83	9.63	11.73	1.26	9.36	10.35	9.71	1.10	9.97	6.31	10.10	0.86
		300	9.84	4.99	5.59	0.69	8.58	2.62	5.53	0.64	12.01	5.33	5.10	0.50
		600	6.61	2.43	3.67	0.42	8.47	2.68	5.22	1.08	NA	4.21	4.68	0.47
	200	200	14.46	6.14	10.94	1.28	18.04	7.31	13.77	1.79	25.01	8.57	13.94	2.44
		600	14.26	4.40	7.91	1.44	NA	4.32	7.80	1.47	NA	5.86	8.81	1.41
		1,200	12.88	3.68	5.90	1.33	NA	2.67	6.00	1.49	NA	2.98	6.47	1.02
	300	300	17.40	10.51	20.10	3.05	14.10	23.93	25.14	7.74	NA	13.46	14.07	1.74
		900	11.54	10.63	10.13	2.72	NA	5.57	8.15	2.17	NA	7.04	10.75	2.07
		1,800	15.24	3.80	7.11	1.65	18.55	4.11	6.80	1.23	19.76	6.27	8.57	1.85
10	100	100	13.00	5.87	9.13	0.78	13.00	5.62	8.38	0.67	NA	5.21	10.06	1.04
		300	9.05	3.46	5.49	1.11	9.05	1.80	4.28	0.62	NA	2.28	4.25	0.42
		600	7.49	1.30	3.35	0.52	7.49	1.83	2.71	0.18	NA	0.75	3.28	0.39
	200	200	14.19	8.32	13.61	1.39	NA	5.07	10.71	0.99	NA	4.85	12.94	1.69
		600	18.05	3.72	5.91	1.49	21.02	1.65	7.23	1.07	NA	2.06	6.63	1.44
		1,200	15.98	1.85	3.96	0.38	NA	1.41	3.84	0.67	NA	2.96	5.44	1.20
	300	300	21.02	10.58	11.77	2.11	NA	8.97	11.79	1.19	NA	9.29	14.50	3.18
		900	13.40	6.39	8.65	2.58	NA	2.83	6.31	1.08	NA	4.74	6.77	0.69
		1,800	8.36	2.10	5.35	1.20	NA	2.19	5.03	0.93	NA	3.19	4.78	1.35
										Avg.		6.67	9.33	1.40

GA: genetic algorithm, SA: simulated annealing, SEA: self-evolution algorithm, RPD: relative percentage deviation, CPLEX: computational time taken by the local search heuristic algorithm by Kim and Lee (2012), T: the size of planning horizon, *M*: the number of items, *W*: capacity of a container, *F*: unit freight cost of container.

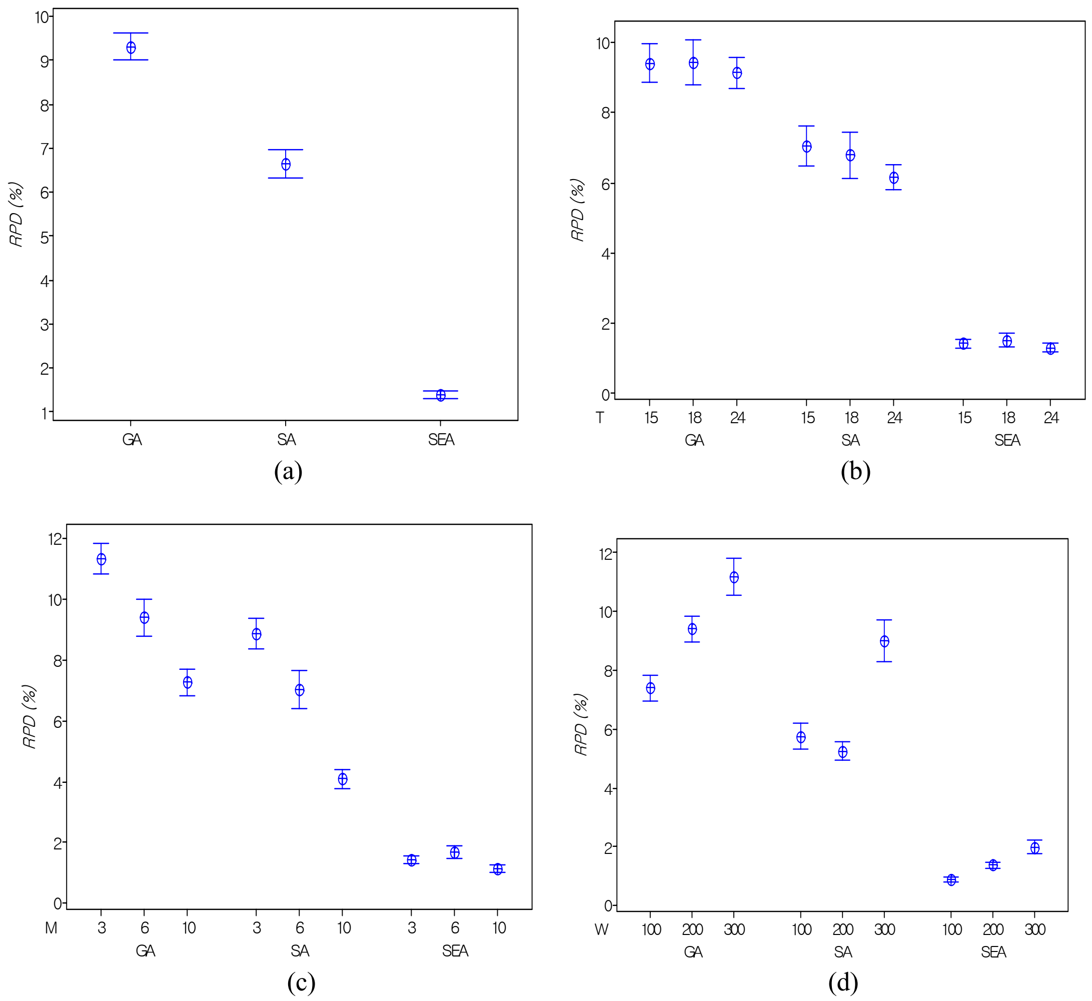

Figure 3. Mean plots and Tukey honest significant difference intervals at the 95% confidence level: (a) algorithm, (b) period (T), (c) products (M), and (d) container capacity (W). GA: genetic algorithm, SA: simulated annealing, SEA: self-evolution algorithm, RPD: relative percentage deviation.

$$\frac{z_{HB} - z_{Opt}}{z_{Opt}} \times 100, \tag{11}$$

Where z_{HB} = the best heuristic solution among SA, GA, and SEA and z_{Opt} = the optimal solution using CPLEX without node limits.

Table 5 shows that the optimal solution using CPLEX was not obtained within 1 hour for many large-sized test problems having more than 15 periods as shown in 'NA' because of the limitations of a computer performance. This result indicates that it is difficult to find the optimal solution as the problem size increases (T and M). For the relative comparison, SEA offers the best performance by showing 1.40% of RPD value in average sense compared to SA and GA. In order to validate the results, it is important to check if the observed differences in the RPD values of each meta-heuristic algorithm are statistically significant. Figure 3 shows the mean plots and Tukey honest significant difference (HSD) intervals at the 95% confidence level of all problems in Table 5. The superiority of SEA can be explained by the graphs in the graph in Figure 3a. The RPD value of SEA is consistent with small variance as T (the size of planning horizon) and M (the number of items) increase, but it increases as W (container capacity) increases. The computing time of SEA presents no more than 5 seconds.

6. CONCLUSION

In this paper, a dynamic lot sizing problem and shipment scheduling are simultaneously analyzed, where the order size of multiple products and a single container type are simultaneously considered. To address the problem, two different solution approaches are proposed. The first approach is based on a mixed integer programming model. Because the problem is NP-hard, three meta-heuristic algorithms are proposed to increase solution efficiency based on the optimal solution properties. SEA is a new meta-heuristic algorithm which has a population (a set of solutions) based self-evolution mechanism. Two problem groups are tested to verify the performance of proposed meta-heuristic algorithms. SEA offers the best performance by showing 1.40% of RPD value in average sense compared to SA and GA. Also, the RPD value of SEA is consistent with small variance as T (the size of planning horizon) and M (the number of items) increase, but it increases as W (container capacity) increases. Overall, the test results indicate that SEA is very effective and efficient algorithm with low variation for the dynamic inbound ordering problem in calculating the transportation cost.

Further study is required to assess the performance of SEA with other meta-heuristics (tabu-search and ant-colony optimization, etc.) in solving the inbound ordering problem.

REFERENCES

Anily, S. and Tzur, M. (2005), Shipping multiple items by capacitated vehicles: an optimal dynamic programming approach, *Transportation Science*, **39**(2), 233-248.

Barbarosoglu, G. and Ozdamar, L. (2000), Analysis of solution space-dependent performance of simulated annealing: the case of the multi-level capacitated lot sizing problem, *Computers and Operations Research*, **27**(9), 895-903.

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

Florian, M., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1980), Deterministic production planning: algorithms and complexity, *Management Science*, **26**(7), 669-679.

Fumero, F. and Vercellis, C. (1999), Synchronized development of production, inventory, and distribution schedules, *Transportation Science*, **33**(3), 330-340.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Pub., Reading, MA.

Hwang, H. and Sohn, K. I. (1985), An optimal policy for the dynamic transportation-inventory model with deteriorating items, *IIE Transactions*, **17**(3), 233-241.

Jaruphongsa, W., Cetinkaya, S., and Lee, C. Y. (2005), A dynamic lot-sizing model with multi-mode replenishments: polynomial algorithms for special cases with dual and multiple modes, *IIE Transactions*, **37**(5), 453-467.

Joo, C. M. and Kim, B. S. (2012), Non-identical parallel machine scheduling with sequence and machine dependent setup times using meta-heuristic algorithms, *Industrial Engineering and Management Systems*, **11**(1), 114-122.

Kim, B. S. and Lee, W. S. (2012), A multi-product dynamic inbound ordering and shipment scheduling problem at a third-party warehouse, *International Journal of Industrial Engineering*, Forthcoming.

Kirkpatrick, S., Gelatt, C. D. Jr., and Vecchi, M. P. (1983), Optimization by simulated annealing, *Science*, **220** (4598), 671-680.

Lee, C. Y. (1989), A solution to the multiple set-up problem with dynamic demand, *IIE Transactions*, **21**(3), 266-270.

Lee, C. Y., Cetinkaya, S., and Jaruphongsa, W. (2003), A dynamic model for inventory lot sizing and outbound shipment scheduling at a third-party warehouse, *Operations Research*, **51**(5), 735-747.

Lee, W. S., Han, J. H., and Cho, S. J. (2005), A heuristic algorithm for a multi-product dynamic lot-sizing and shipping problem, *International Journal of Production Economics*, **98**(2), 204-214.

Lippman, S. A. (1969), Optimal inventory policy with multiple set-up costs, *Management Science*, **16**(1), 118-138.

Van Norden, L. and van de Velde, S. (2005), Multi-product lot-sizing with a transportation capacity reservation contract, *European Journal of Operational Research*, **165**(1), 127-138.

Wagner, H. M. and Whitin, T. M. (1958), Dynamic version of the economic lot size model, *Management Science*, **5**(1), 89-96.

Wolsey, L. A. (1998), *Integer Programming*, Wiley, New York, NY.

Zangwill, W. I. (1969), A backlogging model and a multi-echelon model of a dynamic economic lot size production system: a network approach, *Management Science*, **15**(9), 506-527.