Computers & Industrial Engineering 135 (2019) 1–9

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Production planning in a two-level supply chain for production-time-dependent products with dynamic demands

Jun-Hee Han[a], Ju-Yong Lee[b,*], Yeong-Dae Kim[c]

[a] System Technology Team, Semiconductor Business, Samsung Electronics Co., Ltd., Hwaseong-Si, Gyeonggi-Do 18448, Republic of Korea
[b] Division of Business Administration & Accounting, College of Business Administration, Kangwon National University, Chuncheon-Si, Gangwon-Do 24341, Republic of Korea
[c] Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, Yuseong-gu, Daejeon 34141, Republic of Korea

ARTICLE INFO

Keywords:
Two-level supply chain
Production planning
Production-time-dependent products
Heuristics

ABSTRACT

We consider a two-level supply chain that consists of multiple suppliers and a manufacturing plant. Each supplier produces semi-finished products needed for several types of (finished) products and delivers the semi-finished products to the manufacturing plant, where the products are produced to satisfy dynamic demands. In each supplier, production times of semi-finished products for different product types may be different, but processing should be started at the same time if they are in the same production batch although they may be completed at different times. The problem considered here is to determine production plans of the suppliers with the objective of minimizing the sum of raw material purchasing costs, production costs, setup costs of the suppliers, transportation costs, and costs for outsourcing semi-finished products. We present a mixed integer programming model and develop a two-step heuristic algorithm in which the problem is decomposed into two subproblems and these subproblems are solved sequentially. We also present solution-improvement procedures. Performance of the algorithm is evaluated by comparing its solutions with optimal solutions and good solutions obtained from a simulated annealing algorithm developed in this study, and results are reported.

1. Introduction

In this paper, we consider a production planning problem in a two-level supply chain that consists of multiple suppliers and a manufacturing plant. Each supplier has a single processing line where can produce all types of semi-finished products to be used to produce finished products at the manufacturing plant. Before starting production at each supplier, a setup operation is needed. After being produced by the suppliers, the semi-finished products are delivered to the manufacturing plant, where finished products are produced. In this two-level supply chain, the decision maker makes a production plan to satisfy the demand for the finished products in each period. That is, decision maker determines suppliers to be used, quantities of each type of semi-finished products produced in each selected supplier, and outsourcing quantities of semi-finished products (if necessary) in each period with the objective of minimizing total costs associated with the supply chain.

This paper focuses on a case in which several semi-finished product types are produced through similar processes at the suppliers. Although the processes are not exactly the same, semi-finished products are produced in almost the same general processes except for the production times, which may be different for different types of semi-finished products. In this paper, it is assumed that processing of these semi-finished products should be started at the same time if those semi-finished products are to be produced simultaneously (in a batch) at some point of time. In other words, different semi-finished product types should be started at the same time, but they are completed at different times depending on their production times. Note that products with such characteristic are called *production-time-dependent products*. Examples of systems producing such products can be found in the poultry farming industry. There are a few types of chickens (semi-finished product types) produced in poultry farms and the chickens are distinguished by the weights. The weights of the chickens usually depend on the raising time (production time) in the farms. Therefore, the (semi-finished) product types are distinguished by the production time. We can also find the production-time-dependent products in other livestock industries such as hog-raising and cattle-raising industries. If we expand this problem into other areas, semiconductor manufacturing and steel industry have similar characteristics. In CMP process in semiconductor manufacturing, the depth of polishing depends on its processing time. Also, the products in steel industry are distinguished

* Corresponding author.
E-mail address: jy.lee@kangwon.ac.kr (J.-Y. Lee).

https://doi.org/10.1016/j.cie.2019.05.036
Received 4 February 2018; Received in revised form 20 May 2019; Accepted 29 May 2019
Available online 30 May 2019
0360-8352/

by adding cold rolling process after pressing rolling process.

We consider a system in which all the suppliers as well as the manufacturing plant are under the control of a managing unit (of a company). Hence, decisions to be made in the problem are whether to select suppliers to start production in each period, and production quantities of the semi-finished product types at those suppliers in each period. Moreover, the company can outsource semi-finished products, if currently opened suppliers cannot produce the amount needed for the manufacturing plant, or if production-related costs of producing the semi-finished products at the supplier are higher than outsourcing cost. In this paper, the problem under consideration is to be called a production planning problem for production-time-dependent products (PP-TDP). To the best of our knowledge, the PP-TDP is very rare except for Han and Kim (2016), in which they consider a supply chain design to establish suppliers among candidates at the beginning of the planning horizon for operation during the planning horizon.

In our problem, PP-TDP, not all suppliers produce semi-finished products and hence only the suppliers that produce products need the setup operations. In other words, we have to determine not only the quantity of products but also which suppliers are to be set up for operation in each period. Therefore, in each period, our problem is similar to a facility location problem (FLP). For each period, if the number of product types is one, production cost of each product is zero, and capacities of all suppliers are infinite, PP-TDP is reduced to the uncapacitated facility location problem (UFLP) (Han & Kim, 2016). Note that since the UFLP is known to be NP-hard (Cornuejols, Nemhauser, & Wolsey, 1997), PP-TDP is also NP-hard. However, there is difference between the typical FLP and our study. In the typical FLP, decisions are made on the openings of facilities at the beginning of the planning horizon and these decisions are not changed throughout the planning horizon, while in the PP-TDP, the decision on the setup operations of suppliers may be changed during the planning horizon.

In the literature, there have been many studies related to FLPs. Erlenkotter (1978) suggests a branch-and-bound algorithm for an uncapacitated facility location problem (UFLP), while Neebe and Rao (1983) and Yang, Chu, and Chen (2012) develop branch-and-bound algorithms for capacitated facility location problems (CFLPs). Also, Jia, Li, Shen, Wu, and Zhong (2012) and Zhang, Luo, and Huang (2013) suggest a branch-and-price algorithm and a branch-and-cut algorithm for the CFLP, respectively. In addition, there are many studies in which heuristic algorithms are proposed for CFLPs including those based on Lagrangian relaxation (Barcelo & Casanovas, 1984; Christofides & Beasley, 1983; Geoffrion & McBride, 1978; Han & Kim, 2016; Klincewocz & Luss, 1986; Li et al., 2013, 2014) and genetic algorithms (Babaie-Kafaki, Ghanbari, & Mahdavi-Amiri, 2012; Drezner, Brimberg, Mladenović, & Salhi, 2015; Fernandes et al., 2014; Wang, Sun, & Fang, 2008).

Our problem, PP-TDP, may look similar to the dynamic facility location problem (DFLP), which is an FLP with multi-period dynamic demands. However, like FLPs, the locations of the facilities, once determined, are maintained throughout the planning horizon in DFLPs. On the other hand, in the PP-TDP, the decisions of whether or not the facilities are to be operated, i.e., the decisions related to the locations of suppliers, may be changed during the planning horizon. Also, there may be positive inventory in DFLPs, while inventory of the semi-finished products is not allowed in the PP-TDP.

For DFLPs, Wesolowsky (1973), Wesolowsky and Truscott (1975) and Sweeney and Tatham (1976) suggest dynamic programming algorithms, while Van Roy and Erlenkotter (1982) suggest an optimal solution algorithm using a dual problem of the original problem. In addition, Canel and Khumawala (1997) develop a branch-and-bound algorithm for an extended version of the problem in which not only setup, transportation, and production costs but also taxes between two countries are considered. There also are various heuristics for DFLPs, such as those of Chardaire, Sutter, and Costa (1996), Canel and Khumawala (2001), Albareda-Sambola, Fernández, Hinojosa, and Puerto (2009), Guerrero, Prodhon, Velasco, and Amaya (2013), and Nadizadeh and Hosseini Nasab (2014). Differently from these studies, Torress-Soto and Üster (2011) allow relocation of facilities during the planning horizon in their model and develop heuristic algorithms based on Lagrangian relaxation and Bender's decomposition. They determine the opening and closing of the facilities in each period but opening can be done only one time in the planning horizon for each facility in their model.

Since our problem, PP-TDP, focuses on production planning of two-level supply chain with dynamic demands over the planning horizon, PP-TDP is also related to the dynamic lot sizing problem (DLSP). However, in PP-TDP, not only the lot sizing decisions but also the selection of suppliers are made in each period and it may be different period by period.

After the work of Wagner and Whitin (1958), there have been many studies related to DLSP. Among them, Aggarwal and Park (1993), Federgruen and Tzur (1991) and Wagelmans, Van Hoesel, and Kolen (1992), Fleischhacker and Zhao (2011) suggest optimal solution algorithms. There are also many heuristic algorithms for solving DLSPs and multi-level lot-sizing problems. Among others, Yelle (1979) and Veral and LaForge (1985) suggest heuristic algorithms for single-level lot-sizing models. Also, various heuristic algorithms have been developed for multi-level lot sizing problems, such as genetic algorithms (Alfares & Turnadi, 2018; Dellaert & Jeunet, 2000; Dellaert, Jeunet, & Jonard, 2000; Homberger, 2008; Zegordi, Kamal Abadi, & Beheshti Nia, 2010) and simulated annealing algorithms (Homberger, 2010; Tang, 2004). On the other hand, recently, there are studies on lot sizing problems in integrated supply chains with stochastic constraints (Gharaei and Pasandideh, 2017a, 2017b; Gharaei et al., 2017, 2018, 2019; Pasandideh, Niaki, & Gharaei, 2015; Shekarabi, Gharaei, & Karimi, 2018).

In this study, we consider a production planning problem in a two-level supply chain for production-time-dependent products with dynamic demands. For the objective of minimizing total costs associated with the supply chain, we develop a heuristic algorithm for not only selection of suppliers to produce semi-finished products in each period but also production quantities of the products in each period. In the heuristic algorithm, the problem is decomposed into two subproblems and solved with a two-step procedure: determining suppliers to be setup for operation in each period, and determining the production quantities for each semi-finished product type at the suppliers to be operated, which are solved sequentially. Then, the solution obtained with this two-step procedure is improved by solution-improve procedures.

The remainder of this paper is organized as follows. In Section 2, we describe the problem in detail and give a mixed integer programming formulation, and then we develop a heuristic algorithm and solution-improvement procedures in Section 3. The algorithm is evaluated through a series of computational experiments and results are reported in Section 4. Finally, Section 5 concludes the paper with a short summary and discussions on possible extensions.

2. Problem description

We consider a problem of production planning in a two-level supply chain that consists of multiple suppliers and a manufacturing plant. The problem is to select suppliers to be operated in each period and to determine production quantities of the semi-finished product types at the selected suppliers in each period. In addition, we determine outsourcing quantities of semi-finished products in each period. For the problem, several assumptions are made based on a real situation of the poultry farming company in Korea. Herein, we use the concept of *batch* of semi-finished products that can be processed at the same time in each supplier. Each supplier has an upper limit on the quantity for a batch of semi-finished products. However, the capacity of the manufacturing plant is large enough to produce finished products to meet the demand of the (external) customers. In addition, the locations of the

manufacturing plant as well as the suppliers are given. Additional assumptions are summarized in below.

(1) The supply chain consists of non-identical suppliers and one manufacturing plant.
(2) Locations of the suppliers and the manufacturing plant are given, and information associated with the suppliers such as production capacity, setup cost, purchase cost of raw materials, production costs, and transportation costs to the manufacturing plant for each semi-finished product type is given.
(3) Each supplier can produce several types of semi-finished products.
(4) Different semi-finished product types may be produced at the same supplier, but production times for different semi-finished product types may be different.
(5) In each supplier, semi-finished product types should be started at the same time if they are in the same production batch. (Different types may be, or often are, completed at different times.)
(6) Setup operations are needed at the suppliers before they start processing semi-finished products. Setup times are the same for all types of semi-finished products and at all suppliers. In addition, it is the same regardless of the size of a batch. Hence, setup time is included in the processing time of each semi-finished product.
(7) In each supplier, when a supplier is processing a batch, another batch cannot be started in the supplier (because of sanitary problems and for ease of management).
(8) Demand for each product type in each period is known but may vary period by period.
(9) If necessary (to satisfy demands of the customers, or retailers), semi-finished products can be outsourced with zero (or negligibly short) lead time.
(10) The transportation time from each supplier to the manufacturing plant is short enough (compared with production time) to be ignored.
(11) There is no shortage of raw materials for the semi-finished products.
(12) We find a plan (to start production) throughout the planning horizon based on information of demands for further periods. (Note that demand data for a few periods after the end of the planning horizon are needed to obtain production plan in the last periods of the planning horizon.)

For a clearer description of the problem, we present a mixed integer programming formulation for the production planning problem. This formulation is also used to find an optimal solution of the problem with a commercial integer program solver. First, we give notation used in the formulation and throughout the paper.

Indices and parameters

i	index for suppliers ($i = 1,\ldots, I$)
l	index for semi-finished product types ($l = 1,\ldots, L$)
t, t'	indices for time periods ($t, t' = 1,\ldots, T, \ldots, T + p_L$)
p_l	production time (in the number of periods) of semi-finished product type l at the suppliers ($p_1 \le p_2 \le \ldots \le p_L$)
S_i^{U}	setup cost for each batch at supplier i
K_i	capacity of supplier i
D_{lt}	demand quantity (at the manufacturing plant) of semi-finished product type l in period t
M_{lt}	outsourcing cost of a semi-finished product of type l in period t
C_{il}^{pro}	production cost of one unit of semi-finished product type l at supplier i
C_i^{pur}	purchase cost of the raw material needed to produce a semi-finished product at supplier i
C_{il}^{tra}	transportation cost of one unit of semi-finished product type l from supplier i to the manufacturing plant
B	a very large (positive) number

Decision variables

Q_{ilt}	processing quantity of semi-finished product type l, the quantity of which the processing is started at supplier i in period t
O_{lt}	outsourcing quantity of a semi-finished product of type l in period t
Y_{ilt}	$=1$ if semi-finished product type l is included in a production batch of which the processing is started by supplier i in period t, and 0 otherwise
Z_{it}	$=1$ if supplier i starts processing of a batch of semi-finished products in period t, and 0 otherwise

Now, we give a mixed integer programming formulation for the problem.

$$[P]\ \text{minimize}\ \sum_{t}^{T}\sum_{l}^{L}\sum_{i}^{I}(C_{il}^{\mathrm{pro}} + C_{il}^{\mathrm{tra}} + C_i^{\mathrm{pur}})Q_{ilt} + \sum_{t}^{T}\sum_{l}^{L} M_{lt}O_{lt} + \sum_{t}^{T}\sum_{i}^{I} S_i^{\mathrm{U}}Z_{it} \tag{1}$$

$$\text{subject to}\ \sum_{i} Q_{ilt} + O_{l,t+p_l} = D_{l,t+p_l}\quad \forall\ l,\ t \tag{2}$$

$$\sum_{l} Q_{ilt} \le K_i \cdot Z_{it}\quad \forall\ i,\ l,\ t \tag{3}$$

$$\sum_{t'=t+1}^{t+p_l} Z_{it'} \le B\cdot(1 - Y_{ilt})\quad \forall\ i,\ l,\ t \tag{4}$$

$$Q_{ilt} \le B\cdot Y_{ilt}\quad \forall\ i,\ l,\ t \tag{5}$$

$$\sum_{l} Y_{ilt} \le L\cdot Z_{it}\quad \forall\ i,\ t \tag{6}$$

$$Y_{ilt},\ Z_{it} \in \{0, 1\}\quad \forall\ i,\ t \tag{7}$$

$$O_{lt},\ Q_{ilt} \ge 0\quad \forall\ i,\ l,\ t \tag{8}$$

The objective function, which is to be minimized, represents production cost, transportation cost, and raw material purchasing cost at the suppliers, outsourcing cost for the semi-finished products, and setup cost of suppliers. Constraint (2) ensures that demand is satisfied by production and outsourcing, while (3) is a capacity constraint of the suppliers, and (4) prevents a supplier from starting a new batch if the supplier is processing a batch. Also, (5) and (6) show relationships of the values of the binary variables, Y_{ilt} and Z_{it}, and production quantities (Q_{ilt}).

3. Heuristic algorithm

The mixed integer programming formulation, [P], given in the previous section can be used to obtain an optimal solution with a commercial integer programming (IP) solver. However, for large of practical size problems, IP solvers may not give an optimal or even a feasible solution within a reasonable amount of time. Note that PP-TDP is NP-hard as stated in the first section. Thus, in this study, we present a heuristic algorithm to solve problems within a reasonably short computation time.

First, to simplify the problem, PP-TDP, we group the demands for semi-finished products according to the starting time of the semi-finished products required to meet the demand. That is, a demand group of period t is defined as the set of demands for all semi-finished product types for which the processing at suppliers should be started in period t to satisfy the demands on time. Before we describe the heuristic algorithm, we give additional notation used in the description.

$\widetilde{D}_t$	demand quantity of demand group t, i.e., the sum of demand quantities of semi-finished products for which the processing should be started in period t to satisfy their demands
V_i	profitability index of supplier i, $V_i = \frac{K_i}{S_i^{\mathrm{U}} + (C_i^{\mathrm{pur}} + C_{il}^{\mathrm{pro}} + C_{il}^{\mathrm{tra}})\cdot K_i}$, which is the reciprocal cost per one unit of a semi-finished product at supplier i (if the supplier is utilized up to its capacity)
r	random number between [1, 2]
E_t^{av}	set of suppliers that are available (to start production) in period t, those that are not processing a batch in period t
E_t^{st}	set of suppliers that are to start processing in period t
U_{it}	$=1$ if supplier i is selected to start processing in period t, i.e., if $i \in E_t^{\mathrm{st}}$, and 0 otherwise

In the heuristic suggested in this paper, a complete solution is obtained

through a two-step procedure. First, we determine suppliers that are to start processing of semi-finished products in each period from the first period to the end of the planning horizon, that is, we allocate suppliers to each period. Then, we determine production quantities of each semi-finished product type at the determined suppliers for the planning horizon. The complete solution is improved through improvement procedures. In the following, we give detailed descriptions of the two-step procedure and solution-improvement procedures.

3.1. Obtaining an initial solution

At the first step of the proposed two-step procedure, we determine the suppliers that are to start processing of semi-finished products in each period. In the problem, it is not cost-effective if the selected suppliers do not produce products fully up to their capacities since the setup costs at the suppliers are generally large. Thus, it is assumed that the selected suppliers use their capacities fully. Then, a knapsack problem can be considered if the demand quantity of demand group t, $\widetilde{D}_t$, and the profitability index of supplier i, V_i, are regarded as the capacity of a knapsack and the value of an item, respectively. That is, after solving this knapsack problem for t, $t = 1, 2, \ldots, T$, we can find the best combination of suppliers that will use their capacity fully. The formulation for the knapsack problem for period t is given below.

$$[\mathrm{KP}_t] \text{ maximize } \sum_{i \in E_t^{\mathrm{av}}} V_i \cdot U_{it} \tag{9}$$

$$\text{subject to } \sum_{i \in E_t^{\mathrm{av}}} K_i \cdot U_{it} \leqslant \widetilde{D}_t \tag{10}$$

$$U_{it} \in \{0,\ 1\} \quad \forall\ i \tag{11}$$

In $[\mathrm{KP}_t]$, the objective function is to maximize the sum of profitability indices of selected suppliers. Constraint (10) ensures that the sum of the capacities of selected suppliers is less than the demand quantity of the demand group. Due to constraint (10), demand of the demand group may not be satisfied. Thus, this study allows outsourcing to satisfy demand that is not satisfied by the production at the suppliers. Note that the total cost (setup cost, production and transportation cost) for one unit of a semi-finished product type is generally less than the outsourcing cost if the suppliers use their capacity fully. In this study, the above knapsack problem is solved with a dynamic programming recursion given by Horowitz and Sahni (1974).

The knapsack problems, $[\mathrm{KP}_t]$, $t = 1,\ldots,T$, are solved one by one from period 1 through period T. In the knapsack problem for period 1, $[\mathrm{KP}_1]$, all suppliers can be considered for being selected, that is, all are in the set of available suppliers (E_1^{av}). If suppler i is selected in a certain period, say in period t, in the solution of $[\mathrm{KP}_t]$, it is included in the set of selected suppliers (E_t^{st}) and is not considered for being selected in the subsequent periods until the end of production.

After determining suppliers to be operated at the first step, production quantities to be produced at the selected suppliers are determined at the second step to obtain a complete solution of PP-TDP. Subproblems of determining production quantities of the selected suppliers can be defined and formulated as the following linear program. Note that $\hat{U}_{it}$ is the value of U_{it} which is determined in $[\mathrm{KP}_t]$.

$$[\mathrm{LP}] \text{ minimize } \sum_t \sum_l \sum_i (C_{il}^{\mathrm{pro}} + C_{il}^{\mathrm{tra}} + C_i^{\mathrm{pur}}) Q_{ilt} + \sum_t \sum_l M_{lt} O_{lt} + \sum_t \sum_i S_i^{\mathrm{U}} \hat{U}_{it} \tag{12}$$

$$\text{subject to } \sum_{i \in E_t^{\mathrm{st}}} Q_{ilt} + O_{l,\,t+p_l} = D_{l,t+p_l} \quad \forall\ l,\ t \tag{13}$$

$$K_i \hat{U}_{it} \geqslant \sum_l Q_{ilt} \quad \forall\ i \in E_t^{\mathrm{st}},\ t \tag{14}$$

$$Q_{ilt},\ O_{lt} \geqslant 0 \quad \forall\ i \in E_t^{\mathrm{st}},\ l,\ t \tag{15}$$

The value of the third term of the objective function is already given from the solutions of the knapsack problems, and constraints (13), (14), and (15) correspond to (2), (3), and (8), respectively. In our solution procedure, [LP] is solved with a commercial software package for linear programming. Solutions of $[\mathrm{KP}_t]$, $t = 1, \ldots, T$ and [LP] give a complete solution of our problem, PP-TDP.

3.2. Solution improvement

3.2.1. Iterative algorithm

In the PP-TDP, there may be cases in which demand quantity in the middle of the planning horizon, say in period t^*, is very large. In such cases, there may not be enough suppliers that can be selected (to be operated) in that period, i.e., in $E_{t^*}^{\mathrm{av}}$ (because many suppliers are already selected in previous periods). In these cases, the outsourcing cost of period t may become very large, and so may the total cost. Therefore, it would be better to consider that period first when selecting suppliers.

In this study, we add the following iterative improvement procedures. First, we select a period with the largest outsourcing cost, say period t', from the initial or current solution. Then, we solve the knapsack problem for period t', $[\mathrm{KP}_{t'}]$. In $[\mathrm{KP}_{t'}]$, all suppliers are assumed to be available in period t'. When suppliers are selected in period t', $E_{t'+1}^{\mathrm{av}}$ is defined and we select suppliers to be operated in period $t' + 1$ among $E_{t'+1}^{\mathrm{av}}$ by solving $[\mathrm{KP}_{t'+1}]$. In the same way, we select suppliers for period $t' + 2$ through period T. After then, we select suppliers for the remaining periods, i.e., from period 1 to period $t' - 1$. A complete solution obtained with this procedure is compared with the best solution obtained so far. If the new solution is better, the current best solution is replaced with this new solution. This improvement procedure is performed iteratively until the solution cannot be improved within a predetermined number of iterations.

The overall procedure of the iterative algorithm can be summarized as follows. Here, g and G are iteration count and the maximum number of iterations allowed without solution improvement, respectively.

Procedure 1. (*Iterative Algorithm*)

Step 0. Set $g = 0$, $E_0^{\mathrm{av}} = \{1,\ 2,\ \ldots,\ I\}$, $E_1^{\mathrm{st}} = \emptyset$, and $TC^* = \infty$. Compute $\widetilde{D}_t$ for $t = 1,\ldots,\ T$.

Step 1. Solve the knapsack problems, $[\mathrm{KP}_t]$, for $t = 1,\ldots,\ T$, with a dynamic programming method, such as the one of Horowitz and Sahni (1974).

Step 2. Solve the linear program, [LP], defined by the solution of the knapsack problems (using a commercial solver). Let the optimal solution value of [LP] be TC^0, and let $g \leftarrow g + 1$. If $TC^0 < TC^*$, let $TC^* \leftarrow TC^0$.

Step 3. Find a period with the largest outsourcing cost, say period t'. Solve $[\mathrm{KP}_t]$, for $t = t',\ldots,\ T$, and then, for $t = 1,\ldots,\ t'-1$. Solve [LP] defined by the solutions of these knapsack problems. Let the optimal solution value of [LP] be TC^g.

Step 4. If$TC^g < TC^*$, let $TC^* \leftarrow TC^g$, $g = 1$ and go to Step 3. Otherwise, let $g \leftarrow g + 1$ and if $g = G$, stop; otherwise, go back to Step 3.

3.2.2. Repeated-perturbation algorithm

By the iterative algorithm described above, we can obtain a solution for a given set of profitability indexes (V_i). If other sets of V_i values are considered, we may obtain other solutions. That is, by changing the V_i values, one can obtain other solutions, hopefully those that are better than the solution of the iterative algorithm. Therefore, after obtaining an initial solution from the iterative algorithm, we perturb V_i values for a new set of V_i values by multiplying a random number r, $1 \leq r < 2$, to the original V_i value of each supplier and obtain a complete solution with the iterative algorithm. In the repeated-perturbation algorithm, complete solutions are generated with N sets of V_i values, and the best

solution is selected as the final solution of the heuristic.

The overall procedure of the repeated-perturbation algorithm is summarized below. Here, n and N are the repetition count and the maximum allowed number of repetitions. Also, g and G defined to describe the iterative algorithm are also used here.

Procedure 2. (*Repeated-Perturbation Algorithm*)

Step 0. Set $n = 0$, $g = 0$, $E_0^{\text{av}} = \{1, 2, \ldots, I\}$, $E_1^{\text{st}} = \varnothing$, and $TC^* = \infty$. Compute $\tilde{D}_t$ for $t = 1,\ldots, T$.

Step 1. Let $n \leftarrow n + 1$ and solve the knapsack problems, $[\text{KP}_t]$, $t = 1, \ldots, T$, with a dynamic programming method.

Step 2. Solve the linear program, [LP], defined by the solution of the knapsack problems (using a commercial solver). Let the optimal solution value of [LP] be TC_n^0, and let $g \leftarrow g + 1$. If $TC_n^0 < TC^*$, let $TC^* \leftarrow TC_n^0$.

Step 3. Find a period with the largest outsourcing cost, say period t'. Solve $[\text{KP}_t]$, for $t = t',\ldots, T$, and then, for $t = 1,\ldots, t'-1$. Solve [LP] defined by the solutions of these knapsack problems. Let the optimal solution value of [LP] be TC_n^g.

Step 4. If $TC_n^g < TC^*$, let $TC^* \leftarrow TC_n^g$, reset $g = 0$ and go to Step 3. Otherwise, let $g \leftarrow g + 1$ and if $g = G$, go to Step 5; otherwise, go back to Step 3.

Step 5. If $n = N$, stop. Otherwise, reset $V_i \leftarrow rV_i$ for all i and $g \leftarrow 0$. Go back to Step 1.

4. Computational results

We performed computational experiments to evaluate the performance of the heuristic algorithms proposed in this study. For the experiments, the heuristics are evaluated in two ways. For small sized problem instances, solutions obtained by the heuristics are compared with optimal solutions obtained by a commercial software package for linear and integer programs, CPLEX 12.1. For large sized instances, the heuristics are compared with a simulated annealing (SA) algorithm, which is a kind of meta-heuristic algorithm widely used in optimization problems. Especially in production-related applications such as production planning, SA is a common discipline (Bakar et al., 2016). Note that in real-world systems, i.e., in the aforementioned poultry company, production plans are determined by rule of thumb or based on managers' experience without any systematic method, so there is no (existing) rule or algorithm with which the suggested heuristics can be compared.

In the SA algorithm used in the experiments, we generate an initial solution by randomly selecting suppliers to be opened in each period, and solve [LP] to determine production and outsourcing quantities. A neighborhood solution, $\boldsymbol{\sigma}'$, from the current solution, $\boldsymbol{\sigma}$, is generated using an insertion method, as suggested in Park and Kim (1997). In the current solution, if there are suppliers more than necessary for demand in a period, it may costs a lot to setup the suppliers. Thus, we select a supplier randomly among those to be opened in a period with the largest number of opened suppliers in the current solution, and determine when to open the selected supplier randomly from period 1 to period T. If $\Delta = f(\boldsymbol{\sigma}') - f(\boldsymbol{\sigma}) < 0$, where $f(\bullet)$ is the solution value (total cost) of solution $\bullet$, the neighborhood solution $\boldsymbol{\sigma}'$ is accepted, that is, the current solution, $\boldsymbol{\sigma}$, is replaced by $\boldsymbol{\sigma}'$. If $\Delta \geq 0$, $\boldsymbol{\sigma}'$ is accepted with a specified probability given with a function, $e^{-\Delta/t}$, where t is a parameter called the *temperature*. The temperature is initially set to T_0 and decreased by multiplying $c < 1$, called the *cooling ratio*, if the solution cannot be improved for Λ iterations. (Λ is called the *epoch length*.) The algorithm is terminated when the computation time reaches 4800 s. Note that the solution needs to be found within 4800 s in real situations in a poultry farming company because 4800 s is the maximum time that they can wait after preparing the data for production planning for a week in the morning on Monday. After a series of preliminary tests on several candidate values for each parameter, we set the values as $(T_0, c) = (1.0, 0.98)$ and Λ is set to the product of the number of suppliers and the number of periods.

For the experiment, we generated two groups of instances with different lengths of the planning horizon: short and long planning horizons. For instances with shorter planning horizons, 180 instances were generated, 10 instances for each of all combinations of three levels (20, 40, and 60) for the number of suppliers, three levels (12, 16, and 20 periods) for the length of the planning horizon, and two levels (low and high) for setup costs. For instances with longer planning horizons, a total of 300 instances were generated, 10 instances for each of all combinations of five levels (20, 30, 40, 50, and 60) for the number of suppliers, three levels (24, 36, and 48 periods) for the length of the planning horizon, and two levels (low and high) for setup costs. These parameters used for generating instances were selected based on information of a real poultry company in Korea. In the poultry company, they produce three types of products (and semi-finished products), and the production times of the three semi-finished types are 3, 4, and 5 weeks. Also, demand per a period, raw material purchasing cost, transportation cost, outsourcing cost were generated based on data of a real poultry company, and the capacities of the suppliers and the setup cost were generated from information of suppliers in the company. These data were generated as follows. Here, $DU(a, b)$ denotes the discrete uniform distribution with range $[a, b]$.

(1) Demand for each semi-finished product in each period was generated from $DU(500, 800)$.
(2) The capacity of a supplier was generated from $DU(200, 400)$.
(3) The raw material purchasing cost was generated from $DU(300, 400)$.
(4) The unit production costs of semi-finished products of types 1, 2, and 3 were generated from $DU(75, 100)$, $DU(100, 125)$, and $DU(125, 150)$, respectively.
(5) Transportation costs for each unit of semi-finished products of types 1, 2, and 3 from a supplier to the plant were generated from $DU(25, 50)$, $DU(50, 75)$, and $DU(75, 100)$, respectively.
(6) The outsourcing costs of each outsourced semi-finished product of types 1, 2, and 3 were generated from $DU(500, 600)$, $DU(600, 700)$, and $DU(700, 800)$, respectively.
(7) The setup costs for instances with low setup costs were generated randomly from $DU(2000, 2500)$, while those of high setup costs were generated from $DU(3000, 3500)$.

In the heuristic algorithms, G, the maximum number of iterations allowed without solution improvement in the iterative algorithm (and in the repeated-perturbation algorithm), and N, the maximum number of repetitions made in the repeated-perturbation algorithm, were set to 10 and 100, respectively. The heuristics and the SA algorithm were coded in Java and [LP] is solved with CPLEX 12.1, a commercial software package for linear and integer programs. Also, optimal solutions were obtained by CPLEX applied to the mixed integer programming formulation given in Section 2. The tests were done on a personal computer with an AMD 3.0-GHz processor and 3 GB RAM.

Results of the tests for instances with shorter planning horizons and low-level setup costs are given in Table 1, which shows averages and standard deviations of percentage errors (PEs) of heuristic solutions from optimal solutions and computation times. The overall average PE of the two-step procedure (TSP) for the initial solution, iterative algorithm (IA), and repeated-perturbation algorithm (RPA) were 3.52%, 2.87%, and 1.88%, respectively. It means that the suggested improvement algorithms are useful to improve solution qualities. Also, when the planning horizon is longer, the percentage errors of all algorithms are larger. The average PE of solutions from the best-working algorithm, RPA, was less than 1.3% in the tests on instances with $T = 12$, and the average PE is less than 2% in the tests on instances with $T = 16$ and $T = 20$.

As can be seen in Table 1, PEs decrease as the number of suppliers increases. This may be due to outsourcing. We mentioned that the

Table 1
Results of tests on short-term instances for low-level setup cost.

T	I	PE (%)[a]			CPU time[b]			
		TSP[c]	IA[d]	RPA[e]	TSP	IA	RPA	CPLEX
12	20	3.01 (0.62)	2.41 (0.49)	1.46 (0.68)	0.06	3.89	7.37	3216.29
	40	2.69 (0.97)	2.17 (0.90)	0.86 (0.67)	0.06	4.14	7.78	5578.54
	60	2.47 (1.00)	1.97 (0.77)	1.54 (0.70)	0.09	5.84	15.21	4599.65
	Average	2.72 (0.88)	2.18 (0.76)	1.29 (0.67)	0.07	4.62	10.12	4464.83
16	20	4.95 (2.24)	3.92 (1.71)	2.06 (1.15)	0.07	5.45	8.64	6063.40
	40	4.12 (1.75)	3.06 (1.13)	1.28 (0.69)	0.10	4.89	13.13	8829.14
	60	3.37 (1.46)	2.41 (1.04)	2.09 (1.06)	0.13	6.69	18.02	8343.99
	Average	4.15 (1.82)	3.13 (1.26)	1.81 (0.97)	0.10	5.68	13.26	7745.51
20	20	6.10 (2.97)	5.94 (2.94)	4.66 (1.87)	0.08	4.42	9.66	8673.77
	40	2.84 (1.69)	2.04 (0.86)	1.12 (0.54)	0.13	8.63	15.01	11786.71
	60	2.20 (1.43)	1.98 (1.03)	1.82 (0.93)	0.15	9.15	19.13	15042.58
	Average	3.71 (2.16)	3.32 (1.25)	2.53 (1.17)	0.12	7.41	14.60	11834.35
Overall		3.52 (1.54)	2.87 (1.45)	1.88 (0.99)	0.10	5.90	12.66	8014.90

[a] Average (and standard deviation in parenthesis) percentage error of the heuristic solutions from the optimal solution.
[b] Average CPU time required for an instance.
[c] Two-step procedure for the initial solution.
[d] Iterative algorithm.
[e] Repeated-perturbation algorithm.

outsourcing cost is greater than the production cost at a supplier. Thus, we may need to reduce outsourcing quantity in order to decrease the total cost. When $I = 20$, the number of suppliers is not large enough for the demands to be covered without outsourcing. In these cases, every supplier may have to be operated to reduce outsourcing quantities, and hence selecting suppliers in our heuristics is less important than those of cases where there are more suppliers. However, if there are a large number of suppliers ($I = 40$, 60) in the supply chain, a majority of demand can be satisfied from production at selected suppliers. That is, the selection of suppliers is more important for reducing the total cost.

CPU times for the heuristic algorithms and CPLEX (used for optimal solutions) are also shown in Table 1. Even in instances with the shortest planning horizon ($T = 12$), CPLEX required more than 30 min, although RPA required less than 20 s. In general, companies need production plans not only for short planning horizons ($T = 12$, 16) but also for longer planning horizons ($T > 20$). For instances with T and I larger than those in the instances used in the above test, it is hard to find optimal solutions in a reasonable time. Thus, the suggested heuristic, RPA, may be considered as a viable tool in such longer-term planning problems.

Results of tests on instances with high-level setup costs are given in Table 2. These results are similar to those given in Table 1. The suggested heuristics give good solutions regardless of levels of setup costs. In this case too, IA improved the solution of TSP, and RPA improved the solutions of IA significantly. Because of the aforementioned reason, when the planning horizon is longer, the performance of the heuristics is worse. The overall average PEs of TSP, IA, and RPA were 3.26%, 2.45%, and 1.63%, respectively. The suggested algorithms also worked well for instances with the largest number of suppliers; PEs of RPA were less than 2% in all instances with $T = 20$, $I = 60$, and the average PE was less than 1%.

Computation time required for the heuristics to solve an instance did not exceed 0.2 s, 18 s, and 25 s for TSP, IA, and RPA, respectively. On the other hand, CPLEX required more than 1 h for the smallest instances ($T = 12$, $I = 20$), and 5 h for the largest instances ($T = 20$, $I = 60$). Among the heuristics, RPA required the longest computation time. However, it required only 5 or 6 s more than IA, meaning that the repeated-perturbation procedure improved solutions within a very short time.

Results of tests on instances of larger sizes, i.e., those with longer planning horizons are given in Tables 3 and 4. In these tables, results of CPLEX are not given, since optimal solutions were not obtained by CPEX in a reasonable amount of CPU time. Table 3 shows results of tests on instances with low-level setup costs. Here, NB denotes the number of

Table 2
Results of tests on short-term instances for high-level setup cost.

T	I	PE (%)[a]			CPU time[b]			
		TSP[c]	IA[d]	RPA[e]	TSP	IA	RPA	CPLEX
12	20	2.71 (0.55)	2.24 (0.62)	1.24 (0.42)	0.07	3.88	8.44	3844.71
	40	2.38 (0.64)	2.13 (0.48)	1.16 (0.41)	0.06	4.23	8.95	6435.19
	60	2.47 (0.51)	1.96 (0.49)	1.04 (0.47)	0.10	5.61	14.16	7041.89
	Average	2.52 (0.63)	2.11 (0.52)	1.15 (0.44)	0.08	4.57	10.52	5773.93
16	20	4.24 (1.87)	2.88 (1.53)	1.95 (0.97)	0.09	5.04	12.65	7236.45
	40	3.99 (1.45)	2.36 (1.29)	1.63 (0.88)	0.11	6.17	15.77	9014.62
	60	3.07 (1.24)	2.06 (0.84)	1.57 (0.62)	0.14	6.96	19.21	9908.14
	Average	3.77 (1.48)	2.43 (1.32)	1.72 (0.90)	0.11	6.06	15.88	8719.74
20	20	5.30 (2.37)	4.73 (2.49)	3.46 (1.64)	0.08	5.08	9.30	10894.90
	40	3.11 (1.20)	2.60 (0.84)	1.61 (0.41)	0.15	6.95	18.36	14874.16
	60	2.09 (1.29)	1.47 (1.01)	0.97 (0.45)	0.16	10.14	19.04	18443.38
	Average	3.50 (1.57)	2.93 (1.39)	2.01 (0.72)	0.13	7.39	15.57	14737.48
Overall		3.26 (1.06)	2.49 (1.28)	1.63 (0.53)	0.10	6.01	11.41	9743.72

a, b, c, d, e See the footnotes of Table 1.

Table 3
Results of tests on long-term instances for low-level setup cost.

T	I	Solutions						CPU time[f]		
		NB[a]			RPD(%)[b]					
		TSP[c]	IA[d]	RPA[e]	TSP	IA	RPA	TSP	IA	RPA
24	20	10	10	10	−4.37	−4.83	−5.23	0.09	4.61	26.92
	30	9	9	10	−4.05	−4.29	−4.91	0.09	6.04	29.62
	40	8	9	10	−0.26	−0.42	−0.75	0.12	6.12	35.86
	50	2	3	4	1.83	1.51	1.04	0.13	7.17	37.22
	60	0	0	0	2.87	2.42	1.96	0.15	8.04	39.92
	Total/average	29	31	34	−0.80	−1.12	−1.58	0.12	6.40	33.80
36	20	9	10	10	−4.16	−4.94	−5.61	0.16	8.57	30.14
	30	10	10	10	−3.94	−4.03	−4.98	0.16	9.60	40.05
	40	5	7	9	−0.63	−1.20	−1.34	0.18	9.86	43.70
	50	2	3	5	1.41	1.17	0.81	0.21	10.44	48.30
	60	0	1	2	2.20	2.04	1.71	0.23	11.09	55.05
	Total/average	26	31	36	−1.02	−1.40	−1.88	0.19	9.91	43.44
48	20	8	10	10	−4.56	−5.12	−5.90	0.29	12.03	35.14
	30	7	9	10	−3.38	−4.40	−4.91	0.32	12.62	46.51
	40	5	7	8	−0.47	−1.00	−1.72	0.36	13.47	49.15
	50	1	2	3	1.64	1.08	0.66	0.38	13.36	57.36
	60	0	0	0	3.73	3.05	2.41	0.47	15.96	60.55
	Total/average	21	28	31	−0.61	−1.28	−1.89	0.36	13.49	49.74
Overall		76	90	101	−0.81	−1.26	−1.78	0.22	9.93	41.89

c, d, e See the footnote of Table 1.
[a] Number of instances (out of 10 instances, 150 instances for overall) for which the heuristic algorithm found better solutions than the SA algorithm.
[b] Average of percentage deviations of heuristic solutions from those of SA.
[f] Average CPU time required for an instance.

instances out of 10 (out of 50 for overall) for which the suggested heuristics found better solutions than those of the SA algorithm, and RPD denotes the relative percentage deviation of the heuristic solutions from the solutions of the SA algorithm.

The overall RPDs were −0.81%, −1.26%, and −1.78% for TSP, IA, and RPA, respectively. This means that the suggested heuristic algorithms gave better solutions than the SA algorithm. Even TSP, without improvement procedures, gave better solutions than the SA algorithm in 76 among 150 instances. RPA, which gives the best solutions among the heuristics, worked better than the SA algorithm in 101 instances among 150. CPU time required to solve an instance did not exceed 1 s, 16 s, and 61 s for TSP, IA, and RPA, respectively, while the SA algorithm required 4800 s, which is the time limit specified for the termination criterion.

Similar results were obtained in the cases with high-level setup costs, as shown in Table 4. The overall RPDs were −2.04%, −2.14%, and −2.38% for TSP, IA, and RPA, respectively. In these cases too, the suggested heuristics gave better solutions than the SA algorithm. The overall RPDs were smaller and the heuristics found the best solutions in more instances than those of the cases with low-level setup costs. CPU

Table 4
Results of tests on long-term instances for high-level setup cost.

T	I	Solutions						CPU time[f]		
		NB[a]			RPD(%)[b]					
		TSP[c]	IA[d]	RPA[e]	TSP	IA	RPA	TSP	IA	RPA
24	20	10	10	10	−5.31	−5.41	−5.81	0.12	6.01	31.23
	30	10	10	10	−4.81	−4.74	−5.36	0.13	6.07	32.37
	40	9	9	10	−2.09	−1.96	−2.26	0.15	6.86	43.60
	50	4	6	6	1.27	1.08	0.94	0.17	7.61	54.57
	60	1	2	2	1.78	1.48	1.30	0.18	8.13	66.09
	Total/average	34	37	38	−1.83	−1.91	−2.24	0.15	6.94	45.57
36	20	10	10	10	−6.24	−6.57	−6.74	0.17	8.11	30.14
	30	10	10	10	−4.77	−4.93	−5.21	0.22	10.57	40.05
	40	10	10	10	−1.37	−1.54	−1.75	0.23	11.13	53.70
	50	4	4	4	0.38	0.36	0.33	0.24	11.61	68.33
	60	0	1	1	2.10	1.94	1.69	0.27	13.10	85.05
	Total/average	34	35	35	−1.98	−2.15	−2.34	0.23	10.90	55.45
48	20	10	10	10	−6.05	−6.14	−6.58	0.36	15.34	45.37
	30	10	10	10	−5.33	−5.38	−5.76	0.44	16.12	57.76
	40	8	8	10	−1.42	−1.56	−1.63	0.57	16.01	65.05
	50	3	4	6	0.22	0.20	0.20	0.62	19.14	84.90
	60	0	0	0	1.11	1.08	0.96	0.81	22.66	99.35
	Total/average	31	32	36	−2.30	−2.36	−2.56	0.56	17.85	70.49
		99	103	109	−2.04	−2.14	−2.38	0.31	11.90	57.17

a, b, c, d, e, f See the footnotes of Table 3.

Table 5
Comparison of RPA and SA2 (results of tests on instances with $T = 48$).

Level	I	RPD (%)[a]	CPU time[b]	
			RPA	SA2
Low	20	0.21	35.14	7739.18
	30	0.35	46.51	13654.06
	40	0.84	49.15	15642.73
	50	2.64	57.36	19195.52
	60	3.05	60.55	24463.41
	Average	1.42	49.74	16138.98
High	20	0.34	45.37	7780.30
	30	0.41	57.76	13702.18
	40	1.02	65.05	15643.32
	50	2.56	84.90	19443.94
	60	2.95	99.35	24031.23
	Average	1.46	70.49	16120.19
Overall		1.44	60.11	16129.59

[a] Average percentage deviations of solutions of RPA from those of SA2.
[b] Average CPU time required for an instance.

time required for the suggested heuristics were longer in the cases with high-level setup costs. For example, in cases of the largest instances ($I = 60$, $T = 48$), TSP, IA, and RPA required less than 1 s, 23 s, and 100 s, respectively. The average CPU time of RPA, which worked best, was 57.17 s.

In the SA algorithm used in the above test, a random solution is used as the initial solution. To evaluate the performance of the proposed heuristics with better solutions for large-size instances (with $T = 48$), of which the optimal solutions cannot be obtained with CPLEX, we use another SA algorithm, in which the solution from RPA is used as the initial solution. This new SA algorithm, denoted by SA2, was let to be terminated when the temperature decreases down to $T_f = 0.001$ so that sufficiently good, possibly near optimal, solutions can be obtained. Note that SA2 is used to evaluate the performance of the suggested heuristics comparing solutions with near optimal solutions. Results of the tests are given in Table 5. Although SA2 improved the solutions of RPA by 1.44%, it required a very long computation time. The average CPU time for SA2 was 16129.59 s, and this is too long for SA2 to be used in practice for a poultry company.

In large-size instances tested in this study, which reflect practical situations, optimal solutions could not be obtained within 24 h. However, the company has to make a production plan within one hour, and they modify their production plan every day or every week with updated demand forecasts or production and outsourcing costs. Hence, the optimal solution approach cannot be used in practice. On the other hand, the proposed heuristic algorithms give a reasonably good solution in 20 s. Thus, they can obtain good solutions very quickly even though the parameters of the problem are changed. In addition, by using outsourcing quantity in the solution, one can adjust the number of suppliers in their supply chain. For example, if the outsourcing quantity is large, the company can establish more suppliers or increase the capacity of existing suppliers. Also, if there are suppliers that are never selected for production or selected in only a few periods, the company may completely shut down such suppliers.

5. Concluding remarks

In this paper, we considered a production planning problem for production-time-dependent products with dynamic demands in a supply chain composed of multiple suppliers and a single manufacturing plant. What need to be determined in the problem are whether each of the suppliers should be operated in each period and the production quantities of each supplier as well as the outsourcing quantities. We presented a mixed integer programming formulation and suggested a heuristic algorithm to solve the problem in a reasonable time. In the heuristic, the problem is decomposed into two subproblems, which have the forms of the knapsack problem and linear program (LP), which are solved with a dynamic programming method and a commercial LP solver, respectively. To improve the solution, we presented an iterative algorithm and a repeated-perturbation algorithm.

This research can be extended in several ways. For example, we can consider a supply chain with multiple manufacturing plants instead of a single manufacturing plant. In such a case, transportation quantities between suppliers and plants should be determined as well as the production quantities at the suppliers. Also one may need to consider cases involving design of the supply chain, such as a case in which locations of the suppliers and the manufacturing plant(s) need to be determined.

References

Aggarwal, A., & Park, J. K. (1993). Improved algorithms for economic lot size problems. *Operations Research, 41*, 549–571.

Albareda-Sambola, M., Fernández, E., Hinojosa, Y., & Puerto, J. (2009). The multi-period incremental service facility location problem. *Computers and Operations Research, 36*, 1356–1375.

Alfares, H. K., & Turnadi, R. (2018). Lot sizing and supplier selection with multiple items, multiple periods, quantity discounts, and backordering. *Computer and Industrial Engineering, 116*, 59–71.

Babaie-Kafaki, S., Ghanbari, R., & Mahdavi-Amiri, N. (2012). An efficient and practically robust hybrid metaheuristic algorithm for solving fuzzy bus terminal location problems. *Asia-Pacific Journal of Operational Research, 29*, 1250009 (25 pages).

Bakar, M. R. A., Bakheet, A. J. K., Kamil, F., Kalaf, B. A., Abbas, I. T., & Soon, L. L. (2016). Enhanced simulated annealing for solving aggregate production planning. *Mathematical Problems in Engineering, 2016*, 1679315 (9 pages).

Barcelo, J., & Casanovas, J. (1984). A heuristic Lagrangian algorithm for the capacitated plant location problem. *European Journal of Operational Research, 15*, 212–226.

Canel, C., & Khumawala, B. M. (1997). Multi-period international facilities location: An algorithm and application. *International Journal of Production Research, 35*, 1891–1910.

Canel, C., & Khumawala, B. M. (2001). International facilities location: A heuristic procedure for the dynamic uncapacitated problem. *International Journal of Production Research, 39*, 3975–4000.

Chardaire, P., Sutter, A., & Costa, M. C. (1996). Solving the dynamic facility location problem. *Networks, 28*, 117–124.

Christofides, N., & Beasley, J. E. (1983). Extensions to a Lagrangian relaxation approach for the capacitated warehouse location problem. *European Journal of Operational Research, 12*, 19–28.

Cornuejols, G., Nemhauser, G. L., & Wolsey, L. A. (1997). The uncapacitated facility location problem. In P. Mirchandani, & R. Francis (Eds.). *Discrete location theory* (pp. 119–171). New York: John Wiley and Sons.

Dellaert, N., & Jeunet, J. (2000). Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm. *International Journal of Production Research, 38*, 1083–1099.

Dellaert, N., Jeunet, J., & Jonard, N. (2000). A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs. *International Journal of Production Economics, 68*, 241–257.

Drezner, Z., Brimberg, J., Mladenović, N., & Salhi, S. (2015). New heuristic algorithms for solving the planar *p*-median problem. *Computers and Operations Research, 62*, 296–304.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research, 26*, 992–1009.

Federgruen, A., & Tzur, M. (1991). A simple forward algorithm to solve general dynamic lot sizing models with n periods in O(nlogn) or O(n) time. *Management Science, 37*, 909–925.

Fernandes, D. R. M., Rocha, C., Aloise, D., Ribeiro, G. M., Santos, E. M., & Silva, A. (2014). A simple and effective genetic algorithm for the two-stage capacitated facility location problem. *Computers & Industrial Engineering, 75*, 200–208.

Fleischhacker, A. J., & Zhao, Y. (2011). Planning for dynamic failure: A dynamic lot size model for clinical trial supply chains. *European Journal of Operational Research, 211*, 496–506.

Geoffrion, A. M., & McBride, R. (1978). Lagrangian relaxation applied to capacitated facility location problems. *AIIE Transactions, 10*, 40–47.

Gharaei, A., Karimi, M., & Hoseini Shekarabi, S. A. (2019). Joint economic lot-sizing in multi-product multi-level integrated supply chains: Generalized benders decomposition. *International Journal of Systems Science: Operations & Logistics*. https://doi.org/10.1080/23302674.2019.1585595.

Gharaei, A., & Pasandideh, S. H. R. (2017a). Four-echelon integrated supply chain model with stochastic constraints under shortage condition: Sequential Quadratic Programming. *Industrial Engineering & Management Systems, 16*, 316–329.

Gharaei, A., & Pasandideh, S. H. R. (2017b). Modeling and optimization of four-level integrated supply chain with the aim of determining the optimum stockpile and period length: Sequential quadratic programming. *Journal of Industrial and Production Engineering, 34*, 529–541.

Gharaei, A., Pasandideh, S. H. R., & Khamseh, A. A. (2017). Inventory model in a four-echelon integrated supply chain: Modeling and optimization. *Journal of Modelling in*

Management, 12, 739–762.
Gharaei, A., Pasandideh, S. H. R., & Niaki, S. T. A. (2018). An optimal integrated lot-sizing policy of inventory in a bi-objective multi-level supply chain with stochastic constraints and imperfect products. *Journal of Industrial and Production Engineering, 35*, 6–20.
Guerrero, W. J., Prodhon, C., Velasco, N., & Amaya, C. A. (2013). Hybrid heuristic for the inventory location-routing problem with deterministic demand. *International Journal of Production Economics, 146*, 359–370.
Han, J.-H., & Kim, Y.-D. (2016). Design and operation of a two-level supply chain for production-time-dependent products using Lagrangian relaxation. *Computers & Industrial Engineering, 96*, 118–125.
Homberger, J. (2008). A parallel genetic algorithm for the multilevel unconstrained lot-sizing problem. *INFORMS Journal on Computing, 20*, 124–132.
Homberger, J. (2010). Decentralized multi-level uncapacitated lot-sizing by automated negotiation. *4OR – A Quarterly Journal of Operations Research, 8*, 155–180.
Horowitz, E., & Sahni, S. (1974). Computing partitions with application to the knapsack problem. *Journal of the Association for Computing Machinery, 21*, 277–292.
Jia, S., Li, Z., Shen, H., Wu, T., & Zhong, W. (2012). A logistics network design model with vendor managed inventory. *International Journal of Production Economics, 135*, 754–761.
Klincewocz, J. G., & Luss, H. (1986). A Lagrangian relaxation heuristic for capacitated facility location with single-source constraints. *Journal of the Operational Research Society, 37*, 495–500.
Li, J., Chu, F., Prins, C., & Zhu, Z. (2014). Lower and upper bounds for a two-stage capacity facility location problem with handling costs. *European Journal of Operational Research, 236*, 957–967.
Li, Q., Zeng, B., & Savachkin, A. (2013). Reliable facility location design under disruptions. *Computers and Operations Research, 40*, 901–909.
Nadizadeh, A., & Hosseini Nasab, H. (2014). Solving the dynamic capacitated location-routing problem with fuzzy demands by hybrid heuristic algorithm. *European Journal of Operational Research, 16*, 458–470.
Neebe, A. W., & Rao, M. R. (1983). An algorithm for the fixed-charge assigning users to sources problem. *Journal of the Operational Research Society, 34*, 1107–1113.
Park, M. W., & Kim, Y. D. (1997). Search heuristics for a parallel machine scheduling problem with ready times and due dates. *Computers and Industrial Engineering, 26*, 183–192.
Pasandideh, S. H. R., Niaki, S. T. A., & Gharaei, A. (2015). Optimization of a multiproduct economic production quantity problem with stochastic constraints using sequential quadratic programming. *Knowledge-Based Systems, 84*, 98–107.
Shekarabi, S. A. H., Gharaei, A., & Karimi, M. (2018). Modelling and optimal lot-sizing of integrated multi-level multi-wholesaler supply chains under the shortage and limited warehouse space: Generalised outer approximation. *International Journal of Systems Science: Operations & Logistics,* 1–21. https://doi.org/10.1080/23302674.2018.1435835.
Sweeney, D. J., & Tatham, R. L. (1976). An improved long-run model for dynamic facility location. *Management Science, 22*, 748–758.
Tang, O. (2004). Simulated annealing in lot sizing problems. *International Journal of Production Economics, 88*, 173–181.
Torress-Soto, J. E., & Üster, H. (2011). Dynamic-demand capacitated facility location problems with and without relocation. *International Journal of Production Research, 49*, 3979–4005.
Van Roy, T. J., & Erlenkotter, D. (1982). A dual-based procedure for dynamic facility location. *Management Science, 28*, 1091–1105.
Veral, E. A., & LaForge, R. L. (1985). The performance of a simple incremental lot sizing rule in a multilevel inventory environment. *Decision Sciences, 16*, 57–72.
Wagelmans, A., Van Hoesel, S., & Kolen, A. (1992). Economic lot sizing: An O(nlogn) algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research, 40*, S145–S156.
Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science, 5*, 89–96.
Wang, X. F., Sun, X. M., & Fang, Y. (2008). Genetic algorithm solution for multi-period two-echelon integrated competitive/uncompetitive facility location problem. *Asia-Pacific Journal of Operational Research, 25*, 33–56.
Wesolowsky, G. O. (1973). Dynamic facility location. *Management Science, 19*, 1241–1248.
Wesolowsky, G. O., & Truscott, W. G. (1975). The multiperiod location-allocation problem with relocation of facilities. *Management Science, 22*, 57–65.
Yang, Z., Chu, F., & Chen, H. (2012). A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research, 221*, 521–532.
Yelle, L. E. (1979). Materials requirements lot sizing: A multilevel approach. *International Journal of Production Research, 17*, 223–232.
Zhang, A., Luo, H., & Huang, G. Q. (2013). A bi-objective model for supply chain design of dispersed manufacturing in China. *International Journal of Production Economics, 146*, 48–58.
Zegordi, S. H., Kamal Abadi, I. N., & Beheshti Nia, M. A. (2010). A novel genetic algorithm for solving production and transportation scheduling in a two-stage supply chain. *Computers & industrial Engineering, 58*, 373–381.