Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 670528, 12 pages
http://dx.doi.org/10.1155/2013/670528

Research Article

Solving a Novel Inventory Location Model with Stochastic Constraints and (R, s, S) Inventory Control Policy

Guillermo Cabrera,[1,2] Pablo A. Miranda,[2] Enrique Cabrera,[2,3] Ricardo Soto,[2,4] Broderick Crawford,[2,5] Jose Miguel Rubio,[2,6] and Fernando Paredes[7]

[1] *Department of Engineering Science, University of Auckland, Auckland 1010, New Zealand*
[2] *Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile*
[3] *CIMFAV, Facultad de Ingeniería, Universidad de Valparaíso, Valparaíso 2340000, Chile*
[4] *Universidad Autónoma de Chile, Santiago 7500000, Chile*
[5] *Universidad Finis Terrae, Santiago 7500000, Chile*
[6] *Universidad de Playa Ancha, Valparaíso 33449, Chile*
[7] *Escuela de Ingeniería Industrial, Universidad Diego Portales, Santiago 8370179, Chile*

Correspondence should be addressed to Guillermo Cabrera; guillermo.cabrera@ucv.cl

Received 3 May 2013; Revised 9 August 2013; Accepted 9 August 2013

Academic Editor: Vishal Bhatnaga

We solve a novel inventory-location model with a stochastic capacity constraint based on a periodic inventory control (ILM-PR) policy. The ILM-PR policy implies several changes with regard to other previous models proposed in the literature, which consider continuous review as their inventory policy. One of these changes is the inclusion of the undershoot concept, which has not been considered in previous ILM models in the literature. Based on our model, we are able to design a distribution network for a two-level supply chain, addressing both warehouse location and customer assignment decisions, whilst taking into consideration several aspects of inventory planning, in particular, evaluating the impact of the inventory control review period on the network configuration and system costs. Because the model is a very hard-to solve combinatorial nonlinear optimisation problem, we implemented two heuristics to solve it, namely, Tabu Search and Particle Swarm Optimisation. These approaches were tested over small instances in which they were able to find the optimal solution in just a few seconds. Because the model is a new one, a set of medium-size instances is provided that can be useful as a benchmark in future research. The heuristics showed a good convergence rate when applied to those instances. The results confirm that decision making over the inventory control policy has effects on the distribution network design.

1. Introduction

Distribution network design (DND) is one of the most important problems for companies that distribute products to their customers. The problem consists of selecting specific sites to install plants, warehouses, and distribution centres, assigning customers to serving and interconnecting facilities by flow assignment decisions. DND is typically solved as part of a sequential approach that simplifies associated tactical and operational issues. Hence, the decisions that have been omitted are tackled only after the DND problem has already been solved. This means that decisions related to the location of warehouses and allocation of customers are made without taking into account operational aspects such as transportation and inventory. This situation leads to suboptimal designs because operational decisions are restricted to the current network design.

This paper considers a two-level supply chain, in which a single plant serves a set of warehouses, which in turn serve a set of end customers or retailers. Unlike traditional approaches and in accordance with the recent inventory-location literature, in this paper, we incorporate the inventory control policy as a relevant factor that directly affects DND. A distinctive feature of our model is that it is based on a periodic inventory control policy (R, s, S) for each distribution centre (DC) in a single-product scenario, which is important for

those industries in which a continuous revision policy is not possible. Because our model is a very hard-to-solve nonlinear one, we have solved it using well-known heuristic approaches, namely, Tabu Search (TS) and Particle Swarm Optimisation (PSO). Moreover, as the model was recently developed by us, there is no set of instances that can be used as a benchmark. Thus, in this paper, we present two types of medium-sized instances. The first type considers uniformly distributed customers. The other one considers two clusters of customers. Warehouses are uniformly distributed in both types of instances.

The remainder of the paper is organised as follows. First, a literature review of the main ILM is presented in Section 2. Then, in Section 3, we present and analyse the stochastic capacity constraint under periodic review (R, s, S). At the end of that section, the formulation of the model is presented. Section 4 briefly presents a review of both TS and PSO algorithms and some features of the corresponding implementations. Section 5 starts explaining the procedure used to generate the set of instances presented in this paper. In Section 5.1, the obtained results are presented. Finally, in the last section, some conclusions based on the numerical results are outlined.

2. Literature Review

Many authors have been focused on the DND problem and on ILM in particular during the last 15 years. For instance Simchi-Levi and Zhao [1], Mourits and Evers [2], Bradley and Arntzen [3], Miranda and Garrido [4], and Miranda [5] all analysed the levels of decision making related to DND and supply chain management (SCM). Daskin [6], Simchi-Levi et al. [7], and Drezner and Hamacher [8] present detailed FLP reviews and analysis. However, the traditional structure of FLP is not useful for considering interactions between facility location and inventory decisions or the impact of the latter on network configurations. For example, the risk pooling effect states that the total system safety stock is reduced when customers are served by a smaller number of warehouses. Daskin et al. [9] and Shen et al. [10] incorporate an (Q, RP) inventory control policy into the widely studied uncapacitated facility location problem (UFLP), establishing a safety stock at each site. Daskin et al. [9] employ Lagrangian relaxation to solve the model, whereas Shen et al. [10] reformulate the model as a set-covering problem and solve it using a column generation method. Based on the same inventory control policy, Miranda and Garrido [4] consider the order quantity for each warehouse as a decision variable and the capacitated facility location problem (CFLP) as a base framework. Finally, in Miranda and Garrido [11] and Ozsen et al. [12] authors handle capacity constraints by using previous inventory-location models.

More recently, Kumar and Tiwari [13] have presented an ILM that incorporates the risk pooling effect for both safety stock and running inventory. Additionally, in their model, the authors consider the effect produced when warehouses and end customers work jointly. Tancrez et al. [14] have presented a three-level supply chain nonlinear ILM that integrates location, allocation, and shipment sizes. In [15], the authors developed a model for the DND problem that considers the short lifetime of perishable products. To solve their model, the authors implemented a Lagrangian relaxation. One paper that has addressed the DND problem using periodic 3 inventory review is presented in [16]. In their paper, the authors consider a (R, S) inventory policy, which is slightly different from the one that is considered in this work.

The authors have presented different techniques to solve these models. For instance, Bard and Nananukul [17] proposed a branch and price algorithm for an integrated production and inventory routing problem. In [18], Lagrangian relaxation was used to solve a mixed integer linear programming model for multiple echelon and multiple commodity supply chain network design. Heuristics have also been used to solve DND problems. For instance, Armentano et al. [19] TS is used to minimise the production and inventory cost in a model that integrates production and distribution decisions by considering the capacity constraints of the plant. Askin et al. [20] implemented an evolutionary algorithm to solve an ILM considering multicommodity and distribution planning decisions. In their paper, the authors present a very comprehensive description of their genetic algorithm. To the best of our knowledge, PSO has not been considered to solve ILM problems.

As we stated previously, in this paper, we have considered a two-level supply chain with a single plant, a set of warehouses, and a set of end customers or retailers. Because we incorporate the inventory control policy as a relevant factor that directly affects DND in this paper, based on a periodic inventory control policy (R, s, S) for each Distribution Center (DC), this model can be considered as a variant of the models presented previously in the literature [4, 9, 12, 21], which considers a policy of continuous inventory review (Q, R). In next section, the new model proposed in this work is presented.

3. Inventory-Location Model with Stochastic Capacity Constraint

The model presented in [11] optimises warehouse location and customer assignment decisions, taking into account fixed installation, transportation, inventory, and fixed ordering costs. The authors assume that each warehouse i operates a continuous inventory review policy based on a (Q, RP) policy to meet a stochastic demand, with mean D (units of product per time unit) and variance V. It is also assumed that the plant takes a lead time, LT, to fill incoming orders from warehouse i. Stochastic constraint on inventory capacity is proposed, assuming a maximum inventory level for each warehouse I^{cap}. This constraint is based on chance constrained programming [22] and it fixes a maximum probability, β, of violating inventory capacity at peak times, which occurs only when orders arrive at the warehouses. This inventory level corresponds to the reorder point RP, which is stated in order to satisfy demand during LT with at least a probability of $1-\alpha$, minus stochastic demand during LT, plus order quantity, $\mathrm{RP} - \mathrm{SD}(\mathrm{LT}) + Q$. Thus, the inventory capacity constraint can be written as a deterministic nonlinear constraint as follows:

$$Q + \left(Z_{1-\alpha} + Z_{1-\beta}\right) \cdot \sqrt{\mathrm{LT}} \cdot \sqrt{V} \leq I^{\mathrm{cap}}. \tag{1}$$

It may be noted that a similar and more conservative capacity constraint is proposed in [12], which ensures that inventory capacity is observed in 100% of the cases. However, when periodic review is considered, particularly assuming a (R, s, S) inventory control policy [23], capacity constraint cannot be stated at any moment and does not take the same form as in (1). In a (R, s, S) inventory control policy, inventory levels are reviewed only every R period, and if the inventory level is lower than s, then an order is submitted to reach the objective level S. Consequently, order size must consider the well-known undershoot magnitude (US), which is the amount of items required in addition to s to reach S units of inventory. The average US as a function of demand mean and variance, D and V, and for a review period R, can be computed as

$$\mathrm{US}(D, V) = \frac{V}{2 \cdot D} + \frac{R \cdot D}{2}. \tag{2}$$

In terms of inventory capacity constraints, peak inventory levels are not controlled at any time, but only at specific times for each review period. The peak inventory level is reached only when orders arrive at the warehouse LT time units after the last order and naturally only if an order was submitted to the central warehouse or plant. Consequently, when an order arrives at a warehouse, the inventory level is

$$\begin{aligned} &\underbrace{(s - \mathrm{US})}_{\text{Inventory level when an order is submitted}} \\ &+ \underbrace{(S - s + \mathrm{US})}_{\text{Submitted order size}} - \underbrace{\mathrm{SD}(\mathrm{LT})}_{\text{Stochastic demand during LT}} \\ &= \underbrace{S - \mathrm{SD}(\mathrm{LT})}_{\text{Maximum inventory level when an order arrives}}. \end{aligned} \tag{3}$$

This expression is not surprising, as when an order is submitted to the plant, it is necessary that the total inventory position (on hand plus on order inventory) reaches level S and that LT time units later, the inventory level is reduced by the demand during the LT, SD(LT). Similar to [11], in this paper, we propose that this constraint must be observed for each peak inventory instant (i.e., for each order period) with a fixed and known probability of $1 - \beta$, but now assuming a periodic review as follows:

$$S \leq I^{\mathrm{cap}} + D \cdot \mathrm{LT} - Z_{1-\beta} \cdot \sqrt{V \cdot \mathrm{LT}}. \tag{4}$$

We then define the minimum order size, Q, as

$$S = s + Q. \tag{5}$$

Consequently, constraint (4) can be rewritten as follows:

$$Q + s \leq I^{\mathrm{cap}} + D \cdot \mathrm{LT} - Z_{1-\beta} \cdot \sqrt{V \cdot \mathrm{LT}}. \tag{6}$$

Finally, the reorder point s is set to ensure that, for each time an order is not submitted (inventory level larger than s), the inventory level is large enough to fill the demand until the next order arrives in $R + \mathrm{LT}$ time units, with a probability or service level of $1 - \alpha$:

$$s = D \cdot (\mathrm{LT} + R) + Z_{1-\alpha} \cdot \sqrt{R + \mathrm{LT}} \cdot \sqrt{V}. \tag{7}$$

Finally, substituting (7) into (6), the inventory capacity constraint is

$$\begin{aligned} &Q + D \cdot R + \left(Z_{1-\alpha} \cdot \sqrt{R + \mathrm{LT}} + Z_{1-\beta} \cdot \sqrt{\mathrm{LT}}\right) \\ &\quad \cdot \sqrt{V} \leq I^{\mathrm{cap}}. \end{aligned} \tag{8}$$

According to the previous inventory control assumption, we describe the proposed Inventory-Location Model with Stochastic Constraints on Inventory Capacity under Periodic Review (ILM-SCC-PR) as a stochastic nonlinear mixed-integer programming model (SNL-MIP). Based on a periodic (R, s, S) inventory control policy, the safety stock to be included in the objective function is the average inventory level just before an order arrives at the warehouse:

$$\begin{aligned} &(s - \mathrm{US}) - D \cdot \mathrm{LT} \\ &= D \cdot R + Z_{1-\alpha} \cdot \sqrt{R + \mathrm{LT}} \cdot \sqrt{V} - \left(\frac{V}{2 \cdot D} + \frac{D \cdot R}{2}\right). \end{aligned} \tag{9}$$

Additionally, inventory and ordering costs related to order size or inventory cycle are evaluated in terms of the minimum order size Q as a decision variable, as in the well-known EOQ model:

$$\frac{\mathrm{OC} \cdot D}{(Q + \mathrm{US})} + \frac{\mathrm{HC} \cdot (Q + \mathrm{US})}{2}. \tag{10}$$

The variables and parameters considered in the mathematical formulation are as follows.

Parameters

N: Number of available sites to install warehouses

M: Number of customers that must be served by the installed warehouses

F_i: Fixed location cost of a warehouse on site i (\$/day)

C_{ij}: Transportation unit cost from the warehouse on site i to customer j (\$/unit)

OC_i: Fixed ordering cost at site i (\$/order)

HC_i: Holding cost per time unit at site i (\$/day)

μ_j: Mean demand of customer j per day

σ_j: Standard deviation of the demand of customer j per day

I_i^{cap}: Capacity at warehouse i

Q_i^{cap}: Maximum order size at warehouse i

R_i: Inventory check period at warehouse i in days

LT_i: Average leadtime at warehouse i in days

$Z_{1-\alpha}$: Value of the Standard Normal Distribution, which accumulates a probability of $1 - \alpha$

$Z_{1-\beta}$: Value of the Standard Normal Distribution, which accumulates a probability of $1 - \beta$.

Variables

X_i: Binary variable. It is equal to 1 if a warehouse is installed on site i and 0 otherwise

Y_{ij}: Binary variable. It is equal to 1 if warehouse i serves customer j and 0 otherwise

Q_i: Order size at warehouse i. It is greater than 0 if $D_i > 0$, and 0 otherwise

D_i: Total mean demand at warehouse i. It is greater than 0 if there exist at least one $Y_{ij} > 0$ and 0 otherwise

V_i: Total variance of the demand at warehouse i. It is greater than 0 if $D_i > 0$ and 0 otherwise.

Then, the optimisation problem is

Min

$$\sum_{i=1}^{N}\left(F_i \cdot X_i\right) + \sum_{i=1}^{N}\sum_{j=1}^{M}\left(C_{ij} \cdot Y_{ij}\right) + \sum_{i=1}^{N}\left(\frac{\mathrm{OC}_i \cdot D_i}{\left(Q_i + \mathrm{US}_i\right)} + \frac{\mathrm{HC}_i \cdot \left(Q_i + \mathrm{US}_i\right)}{2}\right) + \sum_{i=1}^{N}\left(\mathrm{HC}_i\left(D_i \cdot R_i + Z_{1-\alpha} \cdot \sqrt{R_i + \mathrm{LT}_i} \cdot \sqrt{V_i} - \mathrm{US}_i\right)\right) \tag{11}$$

s.t.:

$$\sum_{i=1}^{N} Y_{ij} = 1 \quad \forall j = 1, 2, \ldots, M, \tag{12}$$

$$Y_{ij} \le X_i \quad \forall i = 1, 2, \ldots, N, \ \forall j = 1, 2, \ldots, M, \tag{13}$$

$$Q_i + D_i \cdot R_i + \left(Z_{1-\alpha} \cdot \sqrt{R_i + \mathrm{LT}_i} + Z_{1-\beta} \cdot \sqrt{\mathrm{LT}_i}\right) \cdot \sqrt{V_i} \le I_i^{\mathrm{cap}} \cdot X_i \quad \forall i = 1, 2, \ldots, N, \tag{14}$$

$$Q_i + \mathrm{US}_i \le Q_i^{\mathrm{cap}} \cdot X_i \quad \forall i = 1, 2, \ldots, N, \tag{15}$$

$$D_i = \sum_{j=1}^{M} \mu \cdot Y_{ij} \quad \forall i = 1, 2, \ldots, N, \tag{16}$$

$$V_i = \sum_{j=1}^{M} \sigma_j^2 \cdot Y_{ij} \quad \forall i = 1, 2, \ldots, N, \tag{17}$$

$$\mathrm{US}_i = \begin{cases} \dfrac{V_i}{2 \cdot D_i} + \dfrac{R_i \cdot D_i}{2} & \text{if } D_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$X_i, Y_{ij} \in \{0, 1\} \quad \forall i = 1, \ldots, N; \quad \forall j = 1, \ldots, M. \tag{19}$$

Equation (11) is the total system cost. The first term is the fixed setup and operating cost when opening warehouses. The second term is the daily transport cost between warehouse and customers. The third term contains fixed order and inventory costs related to warehouse order size. The fourth term represents the storage cost associated with safety stock at each warehouse. Equation (12) ensures that retailers are served by only one warehouse. Inequality (13) ensures that customers are only assigned to installed warehouses ($X_i = 1$). Inequality (14) ensures that the inventory capacity of each warehouse is respected at least with a probability $1 - \alpha$. Inequality (15) ensures that the order size is below the maximum order size Q_i^{cap} allowed to warehouse i. Equations (16) and (17) compute the mean and variance of the total demand served by each warehouse i, respectively. Equations (18) determine the value of US_i. Finally, (19) states integrality (0 – 1) for the variables Y_{ij} and X_i. This model is NP hard because it is clearly an extension of the UFLP, which is already NP hard (UFLP can be obtained just by using HC = 0 and OC = 0). In addition, the objective function and one constraint are nonlinear, resulting in a model that is very difficult to solve to optimality using classical mathematical programming techniques. Therefore, in this paper, we attempt to solve medium-size instances by mean of two well-known heuristics called Tabu Search (TS) and Particle Swarm Optimisation (PSO).

4. Heuristic Methods

Heuristic methods are a common approach to solve hard combinatorial optimisation problems. Despite the fact that heuristics do not guarantee optimality, the solutions provided by them can be considered as good suboptimal ones. In contrast exact methods guarantee optimality; however, they usually fail when dealing with medium- and large-sized problems. In this paper, two heuristics have been separately considered to solve our DND problem. The first one corresponds to a well-known local search called Tabu Search. The second one is an evolutionary algorithm called Particle Swarm Optimisation. In the next subsections, an overview of these two heuristics is provided.

4.1. Tabu Search. We can describe the TS approach as a local search technique guided by the use of adaptive or flexible memory structures. However, such a general definition fails to show the specificity of TS and could be confused with other types of Greedy Random Adaptive Search Procedure (GRASP) algorithms. The variety of the tools and search principles introduced and described in [24] are such that the TS can be considered as the seed of a general framework for modern heuristic search [25]. TS has been applied to several combinatorial optimisation problems (see, for instance, [26–29]). So too other techniques [30–32] have been used to solve different variants of the FLP. The TS is essentially a local search algorithm; that is, it needs to "exchange information" with its neighbours. To do that, first, the neighbourhood must be defined. Typically, TS algorithm has only one neighbourhood move that defines a set of possible candidate solutions. This move is used across all executions of the TS algorithm. Because different neighbourhood definitions can lead to different results in this paper, we have implemented two types of neighbourhoods. The first one is defined by a change in the allocation of a customer from one facility to another without

any restriction; that is, the new facility could be either open or closed at the allocation moment. The second one is defined by a change in the allocation of a customer from one facility to another in which the new facility was opened previous to the movement 90% of the time; in other words, the probability of moving a customer to a closed facility is equal to 10%. The TS heuristic provides a diversification mechanism that allows it to get out of low-quality neighbourhoods and "jump" to explore new neighbourhoods. The diversification mechanism implemented in this study is a restart method, which re-initialises a current solution without losing the best solution found by the algorithm. As stated above, the restart method is used in case the TS is unable to move out of low-quality neighbourhoods. The TS algorithm requires the following structures to be implemented.

(i) *solutionVector*: a vector with a size equal to the number of customers. For each customer this vector contains the DC to which the customer is allocated.

(ii) *bestSolutionVector*: a vector corresponding to the best *solutionVector* found during the TS execution.

(iii) *candidateSolutionVector*: a vector resulting from application of the neighbourhood movement to a *solutionVector*.

(iv) *candidateList*: a list of candidateSolutionVectors. From this list, we obtain the next *solutionVector*, which corresponds to the best candidateSolution from the list.

(v) *tabuList*: an $N \times M$ matrix that contains those neighbourhood movements that are prohibited in the current TS iteration.

Additionally, our TS implementation also requires the following parameter list.

(i) *iterationNumber*: Total iteration number to end the algorithm.

(ii) *diversificationBound*: total iteration number to apply diversification criterion (restart method).

(iii) *tabuListSize*: size of *tabuList*. This means the number of iterations for which a specific movement remains banned.

(iv) *listOfCandidates*: size of the set of *candidateList*.

The TS algorithm starts with a random solution in which each customer is allocated to a randomly selected DC. When all customers are assigned, TS validates the model constraints. If the constraints are satisfied, the initial solution (S_0) is assigned as a best solution (S_{best}). A list of candidate solutions (NS_0) is generated after that. They are neighbours of S_0. The size of the NS_0 set corresponds to the *listOfCandidates* parameter. Once the set of NS_0 has been generated, the TS algorithm selects the best candidate solution (NS_0^{best}), which should not stay on the tabu list. If the NS_0^{best} actually is in the *tabuList*, it cannot be selected unless its cost is less than the cost of the S_{best} found until this moment (*aspiration criterion*). When a new current solution S_1 is selected, the tabuList is updated, and a new iteration starts. The algorithm stops when the number of iterations k is equal to the parameter *iterationNumber*. Below, we can see the general framework of our TS implementation.

4.2. Particle Swarm Optimisation. The PSO algorithm was proposed and developed by Kennedy and Eberhart [33–37]. Although the PSO algorithm was initially designed to tackle continuous optimisation problems, during the last decade, it has proven to be a very good alternative for solving combinatorial optimisation problems. In fact, several articles have used this technique to solve complex combinatorial optimisation problems; see, for example, [38, 39]. For a comprehensive analysis of publications concerning PSO approaches, see [40].

Unlike other evolutionary algorithms, PSO does not use the "survival" concept. This is because all particles are kept "alive" throughout the algorithm execution time and at no time is their survival threatened. The algorithm starts when a set of particles $\mathcal{S}$ (or *swarm*) is initialised. Each particle $p^k \in \mathcal{S}$, with $k = 1, \ldots, P$, is represented by a vector in which the value of each element p_j^k with $j = 1, \ldots, M$ corresponds to the warehouse that has been set to a customer j. In our algorithm, the initialisation is a random process. For each particle, we know its current position (denoted by p^k) and its previous best position (denoted by $p^{k(\text{best})}$). We also calculate for each particle the best current neighbour (denoted by $p^{k(\text{neighbour})}$) and the best particle (denoted by p^{best}) found so far. The position of each individual particle is adjusted at each iteration by considering its previous best positions, the position of the best neighbour, and the position of the global best solution found so far (see Algorithm 1).

The general frame for our PSO algorithm is presented below.

The most important step in Algorithm 2 is the particle updating. As we have said before, the PSO algorithm was designed for continuous optimisation problems. Therefore, we need to adapt some of its steps to solve combinatorial problems. One important change is in the strategy used to move particle p^k from its position in the iteration t to its next position in iteration $t+1$. To do that, we have considered three possible alternatives: the first one (namely s_1) corresponds to a change in the allocation of one of the customers (randomly selected) to a new (randomly selected) warehouse. The second alternative (s_2) is a change in the allocation of one of the customers (randomly selected) to a new warehouse following the allocation of either the bestNeighbour ($p^{k(\text{neighbour})}$) or the prior best position ($p^{k(\text{best})}$). Finally, the third alternative (s_3) corresponds to the same movement but now considering the allocation from the global best solution reached so far p^{best}. Therefore, the position of particle p^k in the iteration $t + 1$ will be the best particle among s_1, s_2, and s_3. Below we present the pseudocode for our particle updating process (see Algorithm 3).

As with local search algorithms, the PSO approach can also get trapped in local optima if the global best and local best positions are equal to the position of the particle over

```
begin
    k = 0;
    Generate initialSolution S_k;
    S_best = S_k;
    while k < iterationNumber do
        NS_k = Generate Neighbourhood (S_k);
        find best (NS_k);
        while NS_k^best isTabu do
            if cost of NS_k^best < cost of S_best then
                aspirationCriterion;
            reeplace NS_k^best;
        If cost of ns_k^best < cost of S_best then
            S_best = ns_k^best;
            iteration without improvements = 0;
        iteration without improvements + 1;
        S_k = ns_k^best;
        update (tabuList);
        check (diversificationCriterion);
        k = k + 1
end
```

Algorithm 1: General algorithmic frame for tabu search.

```
begin
    Initialise set S;
    Calculate p^best;
    while maxIteration not reached do
        foreach p^k ∈ S do
            Calculate p^k(best);
            Calculate p^k(neighbour);
        Update p^k;
        Calculate p^best;
end
```

Algorithm 2: General algorithmic frame for particle swarm optimisation.

```
begin
    Particles s_1, s_2, s_3;
    If inertia > rand[0, 1] then
        s_1 = LocalMove(p^k);
    If cognitiveFactor > rand[0, 1] then
        s_2 = LocalMove(p^k(best));
    s_2 = LocalMove(p^k(neighbour));
    If globalCognitiveFactor > rand[0, 1] then
        s_3 = LocalMove(p^best);
    p^k = best(s_1, s_2, s_3);
end
```

Algorithm 3: Particle update.

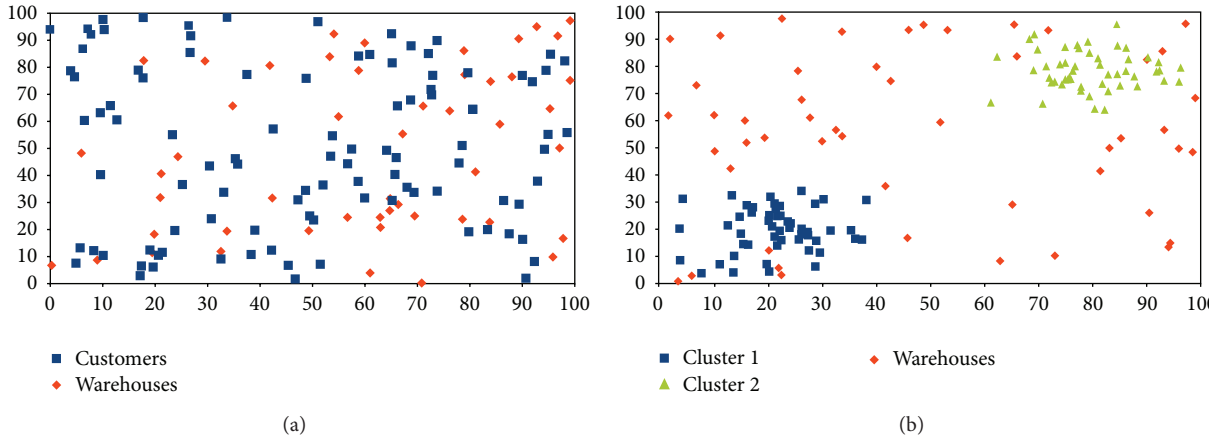

FIGURE 1: Customer and warehouse distributions. (a) Shows that customers (blue squares) are uniformly distributed. (b) Shows that customers are distributed over two clusters (blue squares and green triangles). In both cases, warehouses (orange diamonds are uniformly distributed).

a number of iterations [41]. One distinctive feature of our algorithm is the neighbourhood definition. The idea is that we know that every particle p^k is able to see the best particle at the current iteration. Then, we assume that each particle k can see, at least, any other particle l for which the distance d_{k-l} from p^k to p^l is less than or equal to distance $d_{k\text{-best}}$ from p^k to p^{best}. Distance d_{k-l} is calculated as follows:

$$\text{dist}_{k-l} = \sum_{j=0}^{M} \sqrt{\left(p_j^k - p_j^l\right)^2}. \tag{20}$$

It is clear that in this paper, we do not attempt to develop an original strategy to solve our problem. Instead, we are proposing very simple procedures to obtain solutions for our new model that can be used as a baseline in future research. Moreover, validating some of the main model assumptions is also one of the goals of this work.

5. Experiments and Computational Results

In this section, we present the experiments we have carried out, and we draft some conclusions about our numerical results. As we have noted before in this papaer, our goal is not to state whether the TS or PSO approach is the best way to solve this problem. In fact, we firmly believe that other approaches might be more effective than our approach, and, therefore, applying different strategies to solve our model appears to be a fruitful area for future research into artificial intelligence field. In spite of this, due to the simplicity that both TS and PSO offer, we present these results as a baseline for future studies. We have selected these two algorithms as samples of local search and evolutionary strategies, respectively. Therefore, two of the most important approaches used to solve large and complex combinatorial optimisation problems are covered with these two algorithms.

Moreover, in this section, we present a set of instances that can be used as a benchmark in future studies. The benchmark set consists of two base instances, namely, C_1 and C_2, which means that customers are distributed uniformly and in clusters, respectively. Therefore, for instances of type C_1 customers and DCs are randomly distributed, whereas, for instances of type C_2, only DCs are distributed over the entire area. Customers are distributed randomly along a specific perimeter around the centre of the cluster. The maximum distance allowed between a customer and the centre of the cluster is a parameter. In our case, the number of clusters was equal to 2 and the ratio was 25[km]. These two base instances (C_1 and C_2) were used for all the corresponding instances. Customer demand d_j is calculated using a uniform distribution $U[30, 100]$. Transportation costs (TC) Y_{ij} correspond to 1/5 of the Euclidean distance between a customer j and a warehouse i. Fixed costs (FC) are calculated as the sum of a constant value b (in our case 5000) and a factor dependent on the distance between the warehouse and its closest cluster ($10\times b/\text{mindist}$). With this expression we make those facilities that are close to the "city centre" more expensive. In the same way, the capacity of the warehouse I^{cap} will decrease as the distance to the centre of the cluster becomes small. I_i^{cap} is calculated as the sum of a base value $c = 100$ plus $10\times\text{mindist}$. The order cost (OC) and holding cost (HC) depend on the I^{cap}. The larger the capacity, the smaller the inventory costs. LT is a random number between 2 and 4. The size of the order Q^{cap} is determined as the sum of a constant equal to 100 plus a percentage of the capacity of the warehouse I^{cap}. In our case, this percentage was 50%. FC, HC, OC, Service Levels ($Z_{1-\alpha}$), LT, I^{cap}, and Q^{cap} for each DC are the same for the two base instances. Figure 1 shows these configurations.

From these two main configurations, we generated a set of 22 instances based on them (11 for each one). To do that, we modified values of FC, HC, and OC in ±25%, and additionally we have varied parameter R among values 1, 2 and 3. Then, each instance attempts to evaluate the impact on the network configuration (and on its costs) produced by either increasing or decreasing the parameter values, one at a time. As base instances, we have C_1 and C_2 and based on them we have a set of instances such as $C_1\text{FC}_{.75}$, which means that the FCs from base instance C_1 have been reduced to 75% of their original value.

5.1. Computational Results and Discussion. The computational experiments were performed on an Intel Core Duo processor CPU T2700 2.33 GHz with 2 GB of RAM, using the Ubuntu 12.04 operating system. Both the TS and PSO algorithms were implemented in the JAVA programming language using NetBeans IDE. To validate the algorithms and measure their convergence, we used a set of small instances from [42] that have an optimal solution that was obtained by means of an enumerative method. This enumerative algorithm was executed for approximately 4 hours for each instance, whereas the heuristic algorithms took less than 5 seconds to solve each one. Furthermore, they found the optimal solution all 10 times that they were executed over each instance. We also ran a random search algorithm to determine the real improvement provided by our heuristic. On average, savings when using heuristics algorithms were approximately 300% in all instances.

We fixed both the nonstockout probability and the nonviolate capacity constraint probability at 95%. Hence, the values for α and β are fixed at 1.648. As we mentioned in Section 4 of this paper, we implemented two types of neighbourhood movements for our TS algorithm. Table 1 shows a summary of the results obtained using both movements for all 22 instances.

Column *Instance* corresponds to the evaluated instance. Columns $\overline{x_{m1}}$ and $\overline{x_{m2}}$ correspond to the average value obtained after executing the TS algorithms 10 times using movements $M1$ and $M2$, respectively. In the same way, columns σ_{m1} and σ_{m2} correspond to the standard deviation of the results. Finally, columns Time_{m_1} (sec) and Time_{m_2} (sec) show the average time needed by our TS implementation to find the best solution.

Table 1 shows how, predictably, the average cost increases with the value of period review R. Furthermore, the DND is also affected by the change in the value of R. In several instances, when R increases, the number of open DCs follows it. Figures 2(a) and 2(b) illustrate this situation.

These two situations are produced for the same reason: when the value of R increases, the safety stock must be greater to avoid stockout during the period without review. Thus, DCs cannot attend the same number of customers when the safety stock is increased because of their limited capacity. For this reason, the model is forced to open an extra DC to satisfy total customer demand. This relation is especially important because it shows that the value of R not only affects the cost of the distribution network but also its design.

We now compare the best solutions obtained by the TS algorithm with the ones obtained by the PSO method. Table 2 shows the results of this comparison.

According to the results the PSO algorithm seems to perform slightly better than the TS. That is more evident when we look at C_2 instances where PSO shows a better performance in 9 out of 11 cases. On the other hand, TS is clearly better than PSO when C_1 instances are considered. These are quite interesting results because they suggest that we should take into account how customers are distributed to decide among different heuristic techniques. It is clear that further research is needed in this field.

Because we do not have any bound for our instances, we cannot say anything about the quality of our solutions. Another interesting research line would be to find lower bounds for this model to determine how good the solutions provided by the heuristic methods are.

Moreover, when we examine the components of our objective function, we can see how they behave. In Table 3, columns FC (%), TC (%), Inv (%), and SS (%) show the change (on average) of *Location*, *Allocation*, *Inventory*, and *Safety Stock* cost as a percentage of the base instance C_iR_1.

Another interesting finding is that individual subcosts (FC, TC, Inventory, and SS) do not increase to the same extent that parameters do. In other words, when, for example, FC is increased (decreased) by 25%, the portion of the total cost corresponding to FC does not increase by 25% when compared with its corresponding base instance. This can be associated with the ability of the model to seek other location/allocation alternatives to keep the total cost as low as possible.

Regarding our algorithms, they showed good performance in terms of time and convergence. As we mentioned before, we cannot compare our results with other works because this is a very new model. Figure 3 shows the convergence of the TS algorithm for a specific instance. Here, we can see how the diversification method was invoked several times (peaks in the solid line) during the execution after finding a (presumably) local optima.

Despite the fact that restart method proved to be effective in providing some diversification to our algorithm (and consequently allowing us to move away from local optima), other diversification strategies can be implemented to exploit information from previous iterations and, consequently, provide the algorithm a higher exploration level. With regard to the *aspiration criterion*, this was invoked 30 times on average for each execution with a lower bound of 4 and an upper bound of 71. These values show how the frequency-based memory of the TS works.

6. Conclusions and Future Work

In this paper, we solve a novel inventory-location model that integrates inventory decisions at the strategic level. As a first contribution, our model considers a periodic inventory review policy (R, s, S) unlike previous models that assume continuous inventory review policies. We have also generated a set of 22 medium-sized instances for both *uniformly* and *2-clustered* distributed customers. These medium-sized instances consider 50 DCs and 100 customers. This set of instances was solved using both TS and PSO heuristics independently which yielded an approximation to the optimal solution for each instance. Despite the simplicity of both approaches, good solutions (w.r.t. randomly generated ones) were found in an acceptable amount of time. The results obtained confirm the importance of the value of R, which affects both the cost and configuration of the distribution network. A sensitivity analysis over the key parameters of the model was performed. The numerical results showed the effect on the network produced by changes in both inventory cost (HC and OC) and TC. As future work it is possible to

Table 1: Summary results. Comparison between the results obtained by movements m_1 and m_2.

Instance	$\overline{x_{m_1}}$	σ_{m_1} (%)	Time$_{m_1}$ (sec)	$\overline{x_{m_2}}$	σ_{m_2} (%)	Time$_{m_2}$ (sec)	Gap (%)
C_1FC_{-25}	151541.81	1.57%	305.93	142772.05	0.69%	94.62	5.79%
C_1FC_{+25}	176910.48	0.76%	175.37	172204.30	0.50%	46.83	2.66%
C_1TC_{-25}	155167.37	7.69%	97.86	125920.71	9.85%	86.94	18.85%
C_1TC_{+25}	173094.71	7.54%	289.57	164272.51	9.61%	104.37	5.10%
C_1HC_{-25}	144888.67	8.96%	223.16	134974.87	11.98%	79.43	6.84%
C_1HC_{+25}	185303.83	6.05%	166.17	173147.78	7.57%	88.09	6.56%
C_1OC_{-25}	164188.72	1.48%	155.14	155644.07	0.67%	64.94	5.20%
C_1OC_{+25}	163959.14	1.39%	60.80	157637.04	0.30%	67.82	3.86%
C_1R_1	164081.02	7.75%	203.51	152860.21	16.25%	87.88	6.84%
C_1R_2	226197.19	5.71%	239.07	228169.26	11.13%	249.08	−0.87%
C_1R_3	300915.27	3.25%	217.26	301631.38	6.89%	198.21	−0.24%
C_2FC_{-25}	142217.21	0.43%	235.52	123147.56	2.31%	114.04	13.41%
C_2FC_{+25}	168438.53	0.33%	121.61	142471.58	1.52%	90.45	15.42%
C_2TC_{-25}	146509.00	4.64%	218.86	124372.47	14.23%	199.12	15.11%
C_2TC_{+25}	161718.90	6.10%	294.69	136471.36	15.16%	67.86	15.61%
C_2HC_{-25}	137153.56	7.16%	240.17	114363.85	17.62%	82.98	16.62%
C_2HC_{+25}	172214.43	4.57%	94.59	148850.52	11.69%	65.94	13.57%
C_2OC_{-25}	155132.44	1.61%	181.48	133414.40	0.83%	53.26	14.00%
C_2OC_{+25}	156367.93	1.39%	62.21	133270.49	0.73%	68.75	14.77%
C_2R_1	154938.46	9.34%	216.25	132817.52	10.40%	69.38	14.28%
C_2R_2	197953.21	7.59%	327.43	198309.61	7.06%	164.28	−0.18%
C_2R_3	267045.25	4.54%	254.97	266821.85	4.58%	259.85	0.08%

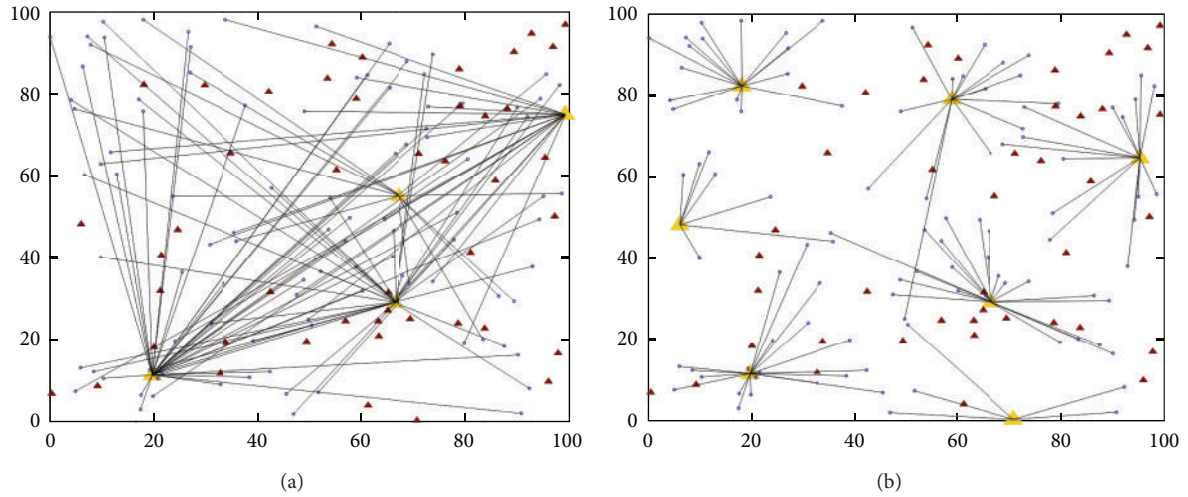

Figure 2: (a) Shows the DND obtained when parameter $R = 1$ and (b) shows the DND obtained when parameter $R = 3$.

integrate other tactical elements such as routing or layout decisions to our ILM, to make them even more representative of reality and therefore improve the quality of the solutions. Examples of these elements are the transportation decisions or the DC layout decisions. Based on the obtained results, we can state (preliminary) that evolutionary algorithms and particularly those based on swarm intelligence such as PSO should be implemented to obtain better solutions for those instances that involve clusters. Local search strategies should be considered for those instances with uniformly distributed customers. However, further investigation is needed in this area.

Moreover, some *matheuristics* approaches (ones that mix mathematical programming and metaheuristics) seem to be another interesting research area to explore. Finally, it would also be interesting to calculate some lower bounds for this model in order to measure how far from the optimal solution our approximations are.

Table 2: Comparison between results obtained by the local search (TS) and the evolutionary algorithm (PSO).

Instance	TS	PSO	Dif (%)
C_1FC_{+25}	142,772.05	142,849.31	0.054%
C_1FC_{+25}	172,204.30	166,232.44	−3.468%
C_1TC_{-25}	125,920.71	147,587.20	17.206%
C_1TC_{+25}	164,272.51	164,450.20	0.108%
C_1HC_{-25}	134,974.87	141,538.00	4.862%
C_1HC_{+25}	173,147.78	178,852.42	3.295%
C_1OC_{-25}	155,644.07	157,435.47	1.151%
C_1OC_{+25}	157,637.04	157,176.59	−0.292%
C_1R_1	152,860.21	156,823.91	2.593%
C_1R_2	226,197.19	226,450.57	0.112%
C_1R_3	300,915.27	301,078.22	0.054%
C_2FC_{-25}	123,147.56	119,074.55	−3.307%
C_2FC_{+25}	142,471.58	145,809.30	2.343%
C_2TC_{-25}	124,372.47	122,065.28	−1.855%
C_2TC_{+25}	136,471.36	135,172.27	−0.952%
C_2HC_{-25}	114,363.85	108,741.32	−4.916%
C_2HC_{+25}	148,850.52	142,216.14	−4.457%
C_2OC_{-25}	133,414.40	127,997.33	−4.060%
C_2OC_{+25}	133,270.49	133,320.84	0.038%
C_2R_1	132,817.52	119,972.02	−9.672%
C_2R_2	198,309.61	197,929.53	−0.192%
C_2R_3	266,821.85	262,059.09	−1.785%

Table 3: Variation of cost distribution at each instance as a percentage of base instances C_iR_1.

Instance	FC (%)	TC (%)	Inv (%)	SS (%)
C_1FC_{-25}	13.75%	−3.59%	−8.11%	−7.35%
C_1FC_{+25}	−24.85%	15.95%	10.29%	10.88%
C_1FC_{-25}	8.15%	16.00%	−9.89%	−10.61%
C_1FC_{+25}	−4.25%	−11.52%	6.07%	6.62%
C_1FC_{-25}	−16.76%	−11.70%	13.72%	14.86%
C_1FC_{+25}	9.79%	12.35%	−9.87%	−10.42%
C_1FC_{-25}	−5.81%	6.78%	1.70%	1.41%
C_1FC_{+25}	−7.10%	7.48%	1.29%	2.53%
C_1R_1	**0.00%**	**0.00%**	**0.00%**	**0.00%**
C_1R_2	31.66%	31.96%	−28.61%	−31.50%
C_1R_3	48.95%	47.11%	−44.14%	−47.62%
C_2FC_{-25}	19.18%	−7.42%	−7.93%	−7.88%
C_2FC_{+25}	−14.05%	2.19%	6.82%	6.87%
C_2FC_{-25}	2.38%	16.64%	−7.00%	−6.63%
C_2FC_{+25}	5.64%	−17.16%	1.94%	2.95%
C_2FC_{-25}	−17.46%	−8.89%	11.43%	12.80%
C_2FC_{+25}	13.25%	12.61%	−11.09%	−11.40%
C_2FC_{-25}	−2.13%	0.07%	2.11%	0.62%
C_2FC_{+25}	0.97%	−1.34%	−1.04%	0.41%
C_2R_1	**0.00%**	**0.00%**	**0.00%**	**0.00%**
C_2R_2	37.02%	36.05%	−30.89%	−32.30%
C_2R_3	52.70%	53.47%	−44.24%	−46.91%

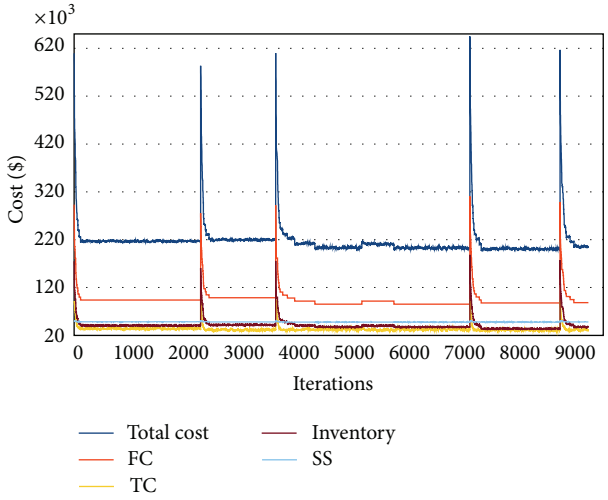

FIGURE 3: Current solution costs. The peaks correspond to the values of the current solution when the restart method is invoked.

Acknowledgment

The authors are indebted to anonymous referee comments and the editor for their valuable comments through the review process. The paper remarkably improved through their recommendations; yet, the authors are responsible for any remaining errors. Fernando Paredes is supported by FONDECYT-Chile Grant 1130455.

References

[1] D. Simchi-Levi and Y. Zhao, "The value of information sharing in a two-stage supply chain with production capacity constraints," *Naval Research Logistics*, vol. 50, no. 8, pp. 888–916, 2003.

[2] M. Mourits and J. J. Evers, "Distribution network design: an integrated planning support framework," *International Journal of Physical Distribution Logistics Management*, vol. 25, pp. 43–57, 1995.

[3] J. R. Bradley and B. C. Arntzen, "The simultaneous planning of production, capacity, and inventory in seasonal demand environments," *Operations Research*, vol. 47, no. 6, pp. 795–806, 1999.

[4] P. A. Miranda and R. A. Garrido, "Incorporating inventory control decisions into a strategic distribution network design model with stochastic demand," *Transportation Research E*, vol. 40, no. 3, pp. 183–207, 2004.

[5] P. Miranda, *Un enfoque integrado para el diseno estrategico de redes de distribucion de carga [Ph.D. thesis]*, Pontificia Universidad Catolica de Chile, 2004.

[6] M. S. Daskin, *Network and Discrete Location: Models, Algorithms, and Applications*, Wiley-Interscience, New York, NY, USA, 1st edition, 1995.

[7] D. Simchi-Levi, X. Chen, and J. Bramel, *The Logic of Logistics*, Springer, New York, NY, USA, 2005.

[8] Z. Drezner and H. Hamacher, *Facility Location. Applications and Theory*, Springer, Berlin, Germany, 2002.

[9] M. S. Daskin, C. R. Coullard, and Z.-J. M. Shen, "An inventory-location model: formulation, solution algorithm and computational results," *Annals of Operations Research*, vol. 110, pp. 83–106, 2002.

[10] Z.-J. M. Shen, C. Coullard, and M. S. Daskin, "A joint location-inventory model," *Transportation Science*, vol. 37, no. 1, pp. 40–55, 2003.

[11] P. A. Miranda and R. A. Garrido, "Valid inequalities for Lagrangian relaxation in an inventory location problem with stochastic capacity," *Transportation Research E*, vol. 44, no. 1, pp. 47–65, 2008.

[12] L. Ozsen, C. R. Coullard, and M. S. Daskin, "Capacitated warehouse location model with risk pooling," *Naval Research Logistics*, vol. 55, no. 4, pp. 295–312, 2008.

[13] S. K. Kumar and M. Tiwari, "Supply chain system design integrated with risk pooling," *Computers Industrial Engineering*, vol. 64, pp. 580–588, 2013.

[14] J.-S. Tancrez, J.-C. Lange, and P. Semal, "A location-inventory model for large three-level supply chains," *Transportation Research E*, vol. 48, no. 2, pp. 485–502, 2012.

[15] Z. Firoozi, N. Ismail, Sh. Ariafar, S. H. Tang, M. K. A. M. Ariffin, and A. Memariani, "Distribution network design for fixed lifetime perishable products: a model and solution approach," *Journal of Applied Mathematics*, vol. 2013, Article ID 891409, 13 pages, 2013.

[16] O. Berman, D. Krass, and M. M. Tajbakhsh, "A coordinated location-inventory model," *European Journal of Operational Research*, vol. 217, no. 3, pp. 500–508, 2012.

[17] J. F. Bard and N. Nananukul, "A branch-and-price algorithm for an integrated production and inventory routing problem," *Computers & Operations Research*, vol. 37, no. 12, pp. 2202–2217, 2010.

[18] H. Badri, M. Bashiri, and T. H. Hejazi, "Integrated strategic and tactical planning in a supply chain network design with a heuristic solution method," *Computers & Operations Research*, vol. 40, no. 4, pp. 1143–1154, 2013.

[19] V. A. Armentano, A. L. Shiguemoto, and A. Løkketangen, "Tabu search with path relinking for an integrated production-distribution problem," *Computers & Operations Research*, vol. 38, no. 8, pp. 1199–1209, 2011.

[20] R. G. Askin, I. Baffo, and M. Xia, "Multi-commodity warehouse location and distribution planning with inventory consideration," *International Journal of Production Research*, 2013.

[21] P. A. Miranda and R. A. Garrido, "A simultaneous inventory control and facility location model with stochastic capacity constraints," *Networks and Spatial Economics*, vol. 6, no. 1, pp. 39–53, 2006.

[22] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management Science*, vol. 6, pp. 73–79, 1959/1960.

[23] S. Axsater, "Approximate optimization of a two-level distribution inventory system," *International Journal of Production Production Economics*, vol. 81-82, pp. 545–5553, 2003, Proceedings of the 7th International Symposium on Inventories.

[24] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic, Norwell, Mass, USA, 1997.

[25] M. Pirlot, "General local search methods," *European Journal of Operational Research*, vol. 92, no. 3, pp. 493–511, 1996.

[26] V. Vails, M. A. Perez, and M. S. Quintanilla, "A tabu search approach to machine scheduling," *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 277–300, 1998.

[27] S. Hanafi and A. Freville, "An efficient tabu search approach for the 0-1 multidimensional knapsack problem," *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 659–675, 1998.

[28] J. Brandão and A. Mercer, "A tabu search algorithm for the multi-trip vehicle routing and scheduling problem," *European Journal of Operational Research*, vol. 100, no. 1, pp. 180–191, 1997.

[29] K. S. Al-Sultan and M. A. Al-Fawzan, "A tabu search approach to the uncapacitated facility location problem," *Annals of Operations Research*, vol. 86, pp. 91–103, 1999.

[30] G. Guillermo Cabrera, E. Cabrera, R. Soto, L. J. M. Rubio, B. Crawford, and F. Paredes, "A hybrid approach using an artificial bee algorithm with mixed integer programming applied to a large-scale capacitated facility location problem," *Mathematical Problems in Engineering*, vol. 2012, Article ID 954249, 14 pages, 2012.

[31] L. Dupont, "Branch and bound algorithm for a facility location problem with concave site dependent costs," *International Journal of Production Economics*, vol. 112, no. 1, pp. 245–254, 2008, Special Section on Recent Developments in the Design, Control, Planning and Scheduling of Productive Systems.

[32] M. A. Arostegui Jr., S. N. Kadipasaoglu, and B. M. Khumawala, "An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location problems," *International Journal of Production Economics*, vol. 103, no. 2, pp. 742–754, 2006.

[33] R. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools*, Academic Press, San Diego, Calif, USA, 1996.

[34] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th IEEE International Symposium on Micro Machine and Human Science*, pp. 39–43, October 1995.

[35] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

[36] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[37] J. Kennedy, "Particle swarm: social adaptation of knowledge," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '97)*, pp. 303–308, April 1997.

[38] G. Guillermo Cabrera, D. Silvana Roncagliolo, J. P. Riquelme, C. Cubillos, and R. Soto, "A hybrid particle swarm optimization-simulated annealing algorithm for the probabilistic travelling salesman problem," *Studies in Informatics and Control*, vol. 21, pp. 49–58, 2012.

[39] L. Cagnina, S. Esquivel, and C. A. Coello, "Hybrid particle swarm optimizers in the single machine scheduling problem: an experimental study," *Studies in Computational Intelligence*, vol. 49, pp. 143–164, 2007.

[40] R. Poli, *Analysis of the Publications on the Applications of Particle Swarm Optimisation, Journal of Artificial Evolution and Applications*, vol. 2008, Article ID 685175, 14 pages, 2008.

[41] *Empirical Study of Particle Swarm Optimization*, vol. 3, 1999.

[42] P. A. Miranda and G. Guillermo Cabrera, "Inventory location problem with stochastic capacity constraint under perodic review (r, s, s)," in *International Conference on Industrial Logistic (ICIL '10)*, vol. 1, pp. 289–296, 2010.