Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 3112–3125
computers & operations research
www.elsevier.com/locate/cor

A scatter search-based heuristic to locate capacitated transshipment points

Burcu B. Keskin, Halit Üster*

Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843-3131, USA

Available online 4 January 2006

Abstract

We consider a fixed charge two-stage location problem in which a given number of intermediate transshipment points are to be located between the supply plants and the customer locations. Both plants and transshipment points are capacitated. Scatter search is a population-based heuristic that has been applied to several combinatorial optimization problems. We develop an efficient scatter search-based heuristic approach with hybrid improvements including local search and path-relinking routines. Computational results demonstrate the effectiveness of the heuristic even for realistic problems with larger instances and tighter capacities.

Keywords: Two-stage location; Transshipment; Scatter search; Heuristics

1. Introduction

Freight transportation plays a key role in today's economies as it allows production and consumption to take place at different locations. The classical transportation problem [1] considers how much of a product should be sent from a set of capacitated suppliers to a set of customers with known demands so that total transportation cost is minimized. On the other hand, in the transshipment problem, the transportation process takes place in two stages: transportation from supply points to transshipment points and transportation from transshipment points to demand points. Regional transshipment points may perform a variety of functions. They can provide reductions in overall shipment costs by permitting bulk shipments from production facilities to transshipment points. This type of cost efficiency cannot be fully utilized when shipments are made directly to demand points from production facilities or from distant storage locations. In such systems, the emphasis is mostly on decreasing the transportation costs. Usually, inventories are not kept at the intermediate transshipment points. However, if the transshipment points are allowed to carry inventories or to perform postponed processing such as final packaging, testing, etc., they help to improve customer relations by decreasing delivery times relative to direct factory shipment, thereby permitting customers to reduce their inventories.

In this paper, we consider an extension of the transshipment problem where the locations of transshipment points are to be decided. We are interested in locating p transshipment points and determining shipment patterns from suppliers to customers through these points. We realistically assume that locating a transshipment point incurs a fixed cost and that capacity limitations exist at these locations as well as at main supply locations. The problem at hand can be considered as a two-stage capacitated location problem. In the first stage, the product is transported from the production plants

* Corresponding author. Tel.: +1 979 845 9573; fax: +1 979 847 9005.
E-mail addresses: burcu@tamu.edu (B.B. Keskin), uster@tamu.edu (H. Üster).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.11.020

to the capacitated depots (transshipment points) whose locations are to be determined and, in the second stage, the customer demands are satisfied from these depots.

The most significant contribution of this paper to the existing literature is to propose an efficient scatter search-based hybrid heuristic framework for the problems discussed above. Scatter search is a population-based metaheuristic that combines existing feasible solutions to create new feasible solutions. It has been successfully applied to various hard optimization problems [2]. In general, heuristics are evaluated based on *speed* (how fast the solutions are obtained) and *effectiveness* (how close they get to optimal) criteria [3]. We present computational analysis results, including the optimality gaps and the durations of the heuristic, to show the performance of our scatter search algorithm.

The remainder of this paper is organized as follows. In the following section, we provide a review of the related literature. In Section 3, we provide a mixed integer programming formulation of the problem to obtain the optimal solutions, and discuss a reduced transshipment problem which is solved within the scatter search heuristic. In Section 4, we provide a scatter search heuristic approach for the problem of interest and we discuss hybrid improvements. In Section 5, we present the results of some computational tests regarding the performance of the overall heuristic approach, and in Section 6, we discuss our on-going and future research directions.

2. Related literature

The problem of interest is related to various logistics models previously studied and reported in the literature. One of them is the well-known p-median facility location problem where the objective is to locate p facilities among a set of possible candidates so that customer demands are satisfied at a minimum total transportation cost. The p-median problem is known to be NP-hard [4], and extensive treatment and review can be found in [5]. In classical p-median formulation, candidate facilities do not possess fixed location costs. When fixed location costs exist, a fixed charge p-median problem is obtained in which case the trade-off between fixed setup costs and variable transportation costs is explicitly addressed in the model. In general, in the p-median problem, the potential facilities do not have capacity restrictions. Two recent studies [6,7] consider capacity limitations at new facility locations; the former develops a genetic algorithm and the latter provides a solution approach with column generation. For the p-median problem, recent research in developing heuristic methodologies, mostly metaheuristics, include interchange heuristics [8] and heuristic concentration [9], genetic algorithms [10–12], tabu search [13], simulated annealing [14], scatter search [15,16], greedy randomized adaptive search [17] and a dynamic programming based heuristic [18].

Our problem can be seen as an extension of the p-median problem. Although we consider locating a predetermined number of facilities, these facilities play the role of intermediaries between the supply and demand points. Furthermore, we consider fixed setup costs for new facilities and limited supply and throughput capacities at the plants (first stage) and at the transhipment points locations (second stage), respectively. In this respect, the problem of interest is also directly related to the two-stage logistics network design studies in the literature.

Although there is a large body of literature on single-stage facility location problems, the studies on two-stage, or multi-stage location problems seem to be very limited. The seminal work by Geoffrion and Graves [19] consider the problem of locating intermediate warehouses in a two-stage, capacitated, multi-commodity distribution system. They assume each customer is serviced by only one facility (i.e. single sourcing). They present an MIP formulation employing 3-index transportation variables which represent the flow of products from the plants through the facilities to their final destinations. Benders decomposition is successfully applied to that specific formulation to solve a real-world problem. Later, a multi-sourcing extension to a similar multi-commodity distribution system design problem and a Lagrangian relaxation solution approach are provided by Hindi and Basta [20]. These studies are extended by Pirkul and Jayaraman [21] and Jayaraman and Pirkul [22]. In [21], Pirkul and Jayaraman consider a capacitated three-tier system consisting of one or more suppliers, warehouses and retailers. It is assumed that the locations of the plants and the warehouses are unknown and that the optimum set of plants and warehouses are selected from a potential set. A heuristic procedure is developed to solve this problem. Similarly, in [22], a three-tier system is considered, where both the plants and the warehouses are located. The model is solved by Lagrangian relaxation. The remainder of the review focuses on single-item, two-stage location problems, because our work falls into this classification.

A two-stage, capacitated, single-sourcing facility location problem is considered by Tragantalerngsak et al. [23]. They consider location decisions in both stages where each facility is supplied by only one uncapacitated plant and each customer is served by only one capacitated facility. The problem is formulated as an IP problem with 3-index transportation variables. A branch-and-bound algorithm is presented based on the most efficient Lagrangian heuristic

found in [24]. To obtain better bounds in a Lagrangian relaxation approach, in [25,26] Klose investigates the structure of an MIP formulation of a two-stage, single-source, capacitated facility location problem with 2-index transportation variables. He shows that the quality of the upper bound deteriorates with a decrease in the capacity of facilities. His computational results show that near-optimal solutions and good lower bounds can be obtained in short computational times even for large problem instances.

The two works most relevant to our study are [27,28]. Marín and Pelegrín [27] consider the location of p transshipment points using 2-index transportation variables. They assume ample supply in the plants and the warehouses, whereas we consider limited capacities at the warehouses and plants. They develop an algorithm using Lagrangian relaxation and branch-and-bound. Marín and Pelegrín [28] consider two-stage, multi-sourcing, capacitated location problems and compare the formulations with 2- and 3-index transportation assignment variables using Lagrangian relaxation. They note that the efficiency of a 3-index formulation is limited to smaller instances; thus it is lower than the efficiency of the 2-index formulation due to the need for greater storage space. Our formulation combines the problem settings of [27,28], and we provide an efficient scatter search-based heuristic for this general problem.

3. Problem formulation

In this section, we provide a mixed integer programming formulation of our problem. For this purpose, we define the following notation for the parameters:

$\mathcal{I}$ set of customers, $i = 1, \ldots, m$
$\mathcal{J}$ set of potential transshipment points, $j = 1, \ldots, n$
$\mathcal{K}$ set of plants, $k = 1, \ldots, K$
p number of transshipment points to be located
D_i demand at customer $i \in \mathcal{I}$
W_j capacity limit at transshipment point $j \in \mathcal{J}$
P_k capacity limit at plant $k \in \mathcal{K}$
f_j fixed cost of locating a transshipment point at $j \in \mathcal{J}$
c_{ij} unit transportation cost from transshipment point $j \in \mathcal{J}$ to customer $i \in \mathcal{I}$
t_{jk} unit transportation cost from plant $k \in \mathcal{K}$ to transshipment point $j \in \mathcal{J}$

We define the following decision variables for this formulation:

z_j 1 if transshipment point $j \in \mathcal{J}$ is located, 0 o.w.
x_{ij} amount of product transported from transshipment point $j \in \mathcal{J}$ to customer $i \in \mathcal{I}$
y_{jk} amount of product transported from plant $k \in \mathcal{K}$ to transshipment point $j \in \mathcal{J}$

Then, the problem (P) can be formulated as follows:

$$\text{Min} \quad \sum_{j\in\mathcal{J}} f_j z_j + \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} c_{ij} x_{ij} + \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} t_{jk} y_{jk}$$

subject to

$$\sum_{j\in\mathcal{J}} x_{ij} = D_i \quad \forall i \in \mathcal{I}, \tag{1}$$

$$\sum_{i\in\mathcal{I}} x_{ij} \leqslant W_j z_j \quad \forall j \in \mathcal{J}, \tag{2}$$

$$\sum_{i\in\mathcal{I}} x_{ij} = \sum_{k\in\mathcal{K}} y_{jk} \quad \forall j \in \mathcal{J}, \tag{3}$$

$$\sum_{j\in\mathcal{J}} y_{jk} \leqslant P_k \quad \forall k \in \mathcal{K}, \tag{4}$$

$$\sum_{j\in\mathcal{J}} z_j = p, \tag{5}$$

$$x_{ij} \geqslant 0, \ y_{jk} \geqslant 0, \ z_j \in \{0,1\} \quad \forall i \in \mathcal{I}, \ \forall j \in \mathcal{J}, \ \forall k \in \mathcal{K}, \tag{6}$$

In the objective function, the first term represents the fixed cost of locating transshipment points; the second term is the total transportation cost from the transshipment points to the customers; and the third term is the total transportation cost from the plants to transshipment points. Constraints (1) ensure that demand at each customer i is satisfied. Constraints (2) and (4) ensure the capacity restrictions at the transshipment points and the plants, respectively. Constraints (3) are for flow conservation at each transshipment point. Constraint (5) specifies the number of transshipment points to be located. Finally, constraints (6) are non-negativity and integrality constraints.

Although they are redundant, in order to obtain a stronger formulation, we add the following constraints to the formulation (P):

$$x_{ij} \leqslant S_{ij} z_j \quad \forall i \in \mathscr{I}, \ \forall j \in \mathscr{J}, \tag{7}$$

$$y_{jk} \leqslant R_{jk} z_j \quad \forall j \in \mathscr{J}, \ \forall k \in \mathscr{K}. \tag{8}$$

In constraint set (7), S_{ij} is given by $\min\{D_i, W_j\}$ for a customer location $i \in \mathscr{I}$ and transshipment point $j \in \mathscr{J}$ pair. These constraints state that a customer can only be assigned to an open transshipment point. In constraint set (8), R_{jk} is given by $\min\{W_j, P_k\}$ for a transshipment point $j \in \mathscr{J}$ and plant $k \in \mathscr{K}$ pair. They are implied by constraints (2), (3) and (4). The new formulation with the added constraints provides stronger lower bounds for solving the problem to optimality. We note that an uncapacitated problem formulation can be obtained by simply excluding the capacity constraints (2), (4), (8), and replacing S_{ij} with D_i in constraints (7). Then, the problem reduces to the generalized p-median problem since the first stage variables can be assigned based on their proximity to the facilities (i.e. lower transportation cost). Formulation (P) is used to find the optimal solution of a problem instance by using CPLEX 9.0 (a trademark of ILOG, Inc.).

Let $\hat{\mathscr{J}} \subset \mathscr{J}$ be the set of open transshipment points such that $|\hat{\mathscr{J}}| = p$ and $\sum_{j \in \hat{\mathscr{J}}} W_j \geqslant \sum_{i \in \mathscr{I}} D_i$. When the open transshipment points are known, the problem (P) reduces to a capacitated transshipment problem, which is relatively easier to solve since it is a linear programming formulation. We call this reduced model (TrP), and it is stated as follows:

$$\text{Min} \quad \sum_{i \in \mathscr{I}} \sum_{j \in \hat{\mathscr{J}}} c_{ij} x_{ij} + \sum_{j \in \hat{\mathscr{J}}} \sum_{k \in \mathscr{K}} t_{jk} y_{jk}$$

subject to

$$\sum_{j \in \hat{\mathscr{J}}} x_{ij} = D_i \quad \forall i \in \mathscr{I}, \tag{9}$$

$$\sum_{i \in \mathscr{I}} x_{ij} \leqslant W_j z_j \quad \forall j \in \hat{\mathscr{J}}, \tag{10}$$

$$\sum_{i \in \mathscr{I}} x_{ij} = \sum_{k \in \mathscr{K}} y_{jk} \quad \forall j \in \hat{\mathscr{J}}, \tag{11}$$

$$\sum_{j \in \hat{\mathscr{J}}} y_{jk} \leqslant P_k \quad \forall k \in \mathscr{K}, \tag{12}$$

$$x_{ij} \geqslant 0, \ y_{jk} \geqslant 0 \quad \forall i \in \mathscr{I}, \ \forall j \in \hat{\mathscr{J}}. \tag{13}$$

In our solution procedure, we will determine the set of open transshipment points in the course of the scatter search heuristic. Given a set of open transshipment points $\hat{\mathscr{J}}$, we solve the corresponding capacitated transshipment problem, (TrP), by using CPLEX.

Alternatively, this problem can be formulated by employing 3-index variables for product flow from plant k via transshipment point j to customer i, as in [23,28,29]. Since a 3-index formulation needs greater storage space as reported by Marin and Pelegrin [28], its use is limited to smaller instances. Therefore, we conclude that the formulation with 2-index transportation assignment variables is preferable to obtain optimal results using exact methods for comparison purposes. In the rest of the paper, the 2-index formulation is employed.

4. A hybrid scatter search framework

Our hybrid scatter search heuristic operates on a binary vector $\mathbf{z}$ of size n representing the locations of the transshipment points. Given the locations of the transshipment points, i.e. an instance of $\mathbf{z}$, we solve the corresponding

capacitated flow problems (TrP) to optimality as mentioned above. For a given solution vector $\mathbf{z}$, we find the corresponding objective function value Z by adding the optimal objective value of (TrP) and the fixed costs of locating open transshipment points (facilities).

Scatter search, which was first introduced by Glover [30], is a population based metaheuristic. It has been implemented in various combinatorial optimization problems such as the traveling salesman problem, the arc routing problem, and the quadratic assignment problem [2]. In very general terms, we can describe the steps of scatter search as follows. We first generate a population of solutions that satisfy a critical level of diversity using a *diversification generation method*. An *improvement method* transforms a trial solution into an enhanced feasible trial solution. Next, we select quality and diverse solutions from the population of solutions to create a reference set. Diversity is generally measured by the distance between the solutions, while the quality of a solution is measured by its objective value. Using a *subset generation method*, we generate subsets of the reference set. These subsets provide a basis for creating new combined solutions. A *solution combination method* transforms a given subset of solutions produced by the subset generation method into one or more combined new solutions. For further improvement of the new solutions, heuristic methods can be applied again. If the new improved solutions are better in either quality or diversity than the solutions in the reference set, we update the reference set with a *reference set update method* and repeat generating new subsets and creating new solutions until the reference set does not change. Since each of the above methods mentioned in italics can be implemented in a variety of ways and with different degrees of complexity, the scatter search procedure is very adaptable to different problems. In a recent paper, Martí et al. [31] examine basic and advanced scatter search designs including a path-relinking methodology.

Only four of the five components are strictly required in the scatter search. The improvement method, as the only exception, is used to generate high quality solutions if they are not provided by other components. Here, we develop scatter search procedures with and without the improvement method. The procedure with improvement uses a local search routine with an *exchange* neighborhood on the binary solution vector which provides us with a hybrid scatter search procedure. We also incorporate a path relinking approach to further improve the solution provided by the scatter search (both with and without improvement approaches). Next, we present the details of our overall hybrid scatter search heuristic with the path relinking procedure.

The parameters used in the scatter search are population size $2 * h_{\max}$ where $h_{\max}$ is the diversification parameter and the number of quality and diverse solutions in the reference set, which are denoted by b_1 and b_2, respectively. The general framework of the scatter search heuristic employed here is given in Display 1.

Display 1. Pseudo-code of the scatter search-based heuristic

Step 0: Initialize the scatter search parameters.
Step 1: Generate the initial population using the *diversification generation method.*
Step 2: Attain feasibility in all the trial solutions of the population.
Step 3: Use the *reference set update method* to build a ReferenceSet $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_b\}$.
Step 4: Generate a collection of subsets $\mathscr{C}$ with the *subset generation method.*
while $\mathscr{C} \neq \emptyset$ **do**
Step 5: Select the next $s \in \mathscr{C}$
Step 6: Apply the *solution combination method* to s to obtain a new solution $\mathbf{V}$.
Step 7: Convert $\mathbf{V}$ to a feasible solution if necessary, and calculate its objective function value using (TrP).
Step 7*: Improve $\mathbf{V}$ using local search (optional)
Step 8: Apply the *reference set update method* to include $\mathbf{V}$ in ReferenceSet.
if the ReferenceSet is changed **then**
Go to Step 4
end if
$\mathscr{C} \leftarrow \mathscr{C} \backslash s$
end while
Step 9: Apply path relinking

As can be seen in this framework, our particular implementation can be divided into two parts. The first part consists of Steps 1, 2 and 3, which is the initialization phase where we produce an initial population and an initial reference set using a seed solution. Its role is to generate high quality and diverse solutions as an input to part 2 (Steps 4–9) where we perform the solutions evolution phase via recombination and improvement in the reference set. The complete scatter search heuristic can be repeated by starting from different initial seed solutions for a wider exploration of the solution space. The details of the implementation for each step are given below.

4.1. Step 0: Initial parameters

In Step 0, the parameters of scatter search, i.e. population size, reference set size, number of quality and diverse solutions, and the initial seed solution are initialized. The selection of the parameters has an important effect on the performance of the heuristic. For the best performance, computational analyses are conducted to find the best set of parameters.

The initial seed solution is used as an input to the diversification generation method. We find our initial seed solution (a binary vector $\mathbf{z}$) using one of three different construction heuristics. The construction heuristics determine which p transshipment points to open from the set of potential transshipment points, that is, to determine a good $\mathbf{z}$. We consider the following construction heuristics:

The Random Heuristic (R): We open p facilities randomly from the set of potential transshipment points $\mathscr{J}$.

The Random + Greedy Heuristic (RG): In this heuristic, we select a fraction $\alpha \in [0, 1]$ (an input parameter) of the p transshipment points to open at random, and then, the rest of the open transshipment points are found in a greedy fashion. If $\alpha = 0$, the heuristic selects p transshipment points greedily, and if $\alpha = 1$, the heuristic is equivalent to the random heuristic. As a greedy ordering criteria, we use the fixed cost of opening a facility. In other words, we sort the set $\mathscr{J}$ according to the fixed cost in an increasing manner. The greedy heuristic selects the first p facilities from the sorted set. This heuristic is suggested by Resende and Werneck in [17] for the p-median problem.

The Randomized Heuristic (RZ): This algorithm is similar to the greedy heuristic, but while selecting the ith facility, instead of selecting the best option among all $n - i + 1$ options, we choose randomly from the $\lceil \alpha(n - i + 1) \rceil$ best facilities, where $\alpha \in (0, 1]$ is an input parameter. Note that, as in the random + greedy heuristic, if $\alpha = 0$, the heuristic is completely greedy, and if $\alpha = 1$, the heuristic is equivalent to the random heuristic. This algorithm is also used in [17] to solve the p-median problem.

During the initial seed construction, we are not concerned with the feasibility of the binary vector $\mathbf{z}$, since feasibility will be attained after generating a diverse population of trial solutions. Among these three heuristics, the random+greedy heuristic performs the best with an input parameter of $\alpha = 0.5$. In Section 5, we present the results from a scatter search algorithm that is initiated with a seed solution generated by RG with an α value of 0.5.

4.2. Step 1: Initial population generation

The aim of diversification is to produce trial solutions that differ from each other as significantly as possible. This process can be viewed as sampling the solution space in a systematic fashion to identify high quality and diverse solutions. Glover [32] presents two different systematic (i.e. non-random) procedures for creating binary solution vectors as trial solutions. We summarize the ideas behind the one that we use in our implementation (see [32] for details).

Let z^i be an element of a binary solution vector $\mathbf{z}$. Given $\mathbf{z}$ as the seed solution, the generator creates $2 * h_{\max}$ new solutions. The first $h_{\max}$ solutions $\mathbf{z}_1, \ldots, \mathbf{z}_{h_{\max}}$ are generated using an incremental parameter k and the following expression:

$$\mathbf{z}_h^{1+hk} = 1 - \mathbf{z}^{1+hk} \quad k = 0, 1, \ldots, \lfloor n/h \rfloor$$

The maximum value of the incremental variable k changes for each value of h. The second $h_{\max}$ solutions are obtained using the relation $\mathbf{1} - \mathbf{z}_h$ where $h = 1, \ldots, h_{\max}$. Although it is recommended to assume $h_{\max}$ to be less than, or equal to, $n/5$, when a large number of diverse solutions is needed, the value of $h_{\max}$ can approach $n - 1$.

4.3. Step 2: Attaining feasibility in trial solutions of the population

The infeasibility of trial solutions is due to violations of two constraints, (2) and (5).

4.3.1. Satisfying constraint (5)

It is likely that some of these initial solutions will not satisfy constraint (5) since the number of open facilities in a trial solution can be lower or higher than the prespecified p value. We can correct this infeasibility via opening or closing some of the facilities (transshipment points) in a trial solution until the total number of open facilities is equal to p. The total number of open transshipment points in a trial solution $\mathbf{z}_o$ is given by $t = \sum_{i=1}^{n} \mathbf{z}_o^i$. If t is less than p, then we need to open $p - t$ transshipment points to satisfy constraint (5). Alternatively, if t is greater than p, we need to close $t - p$ transshipment points. We consider three different heuristics to achieve feasibility with respect to constraint (5):

(1) *Fixed cost heuristic*: This is a greedy heuristic which considers transshipment points according to their fixed costs. First, the transshipment points are ranked in an increasing order of their fixed costs. If t is less than p, we open $p - t$ of the unused facilities from the beginning of the sorted list, i.e. facilities with lower fixed costs. Otherwise, we close $t - p$ of the open facilities from the end of the sorted list, i.e. facilities with higher costs.
(2) *Fixed cost/capacity heuristic*: This is also a greedy heuristic which is similar to the fixed cost heuristic. In this heuristic, we use the fixed cost/capacity ratio of the facility, instead of fixed cost, as a greedy ordering criteria.
(3) *Random location heuristic*: If p is greater than t, we select $p - t$ facilities to open randomly among the unused facilities. Otherwise, $t - p$ of the open facilities are closed randomly.

In order to explore different areas of the solution space, we need a diverse population of trial solutions. However, using either a fixed cost heuristic or a fixed cost/capacity heuristic disturbs the diversity of the population. They result in premature convergence of the scatter search algorithm. Therefore, we choose to implement the random location heuristic to attain feasibility with respect to constraint (5).

4.3.2. Satisfying constraints (2)

After the number of open facilities in a trial solution is corrected to p, we consider its feasibility in terms of the total capacity it provides. If the total capacity of the trial solution, i.e. the total capacity provided by the open transshipment points, is less than the total demand, we can employ one of the following two heuristics to ensure feasibility without violating constraint (5):

(1) *Greedy capacity heuristic*: We manipulate the binary solution vector of a trial solution by swapping the open facility with the lowest capacity with an unused facility that provides just enough capacity to close the gap between the total demand and the total available capacity. First, we close the open facility with the lowest capacity thus increase the current gap between the total demand and total available capacity. Then, we open the facility among the unused ones that has the lowest capacity and that closes the current gap. If no such unused facility exists, we open the unused facility with the largest capacity in order to decrease the current gap as much as possible. We continue in this fashion by swapping open and unused facilities until the overall demand can be satisfied by total capacity of open transshipment points.
(2) *Random capacity heuristic*: In this heuristic, we open a facility chosen randomly among the unused ones and close a randomly chosen open facility. We continue in this manner until the capacity constraint is satisfied.

Note that in both methods, we swap only a pair of facilities between the open and unused facility groups until the total capacity feasibility is obtained. Hence, we do not distort the feasibility with respect to constraint (5). Among these two heuristics, we observed that the random capacity heuristic performs better than the greedy capacity heuristic in attaining the feasibility of the initial population. A scatter search algorithm with the greedy capacity heuristic introduces relatively better quality solutions in the initial population, and thus, has an adverse effect on the diversity of the newly generated solutions. To avoid premature convergence, we employ the random capacity heuristic. After the feasibility of each binary vector $\mathbf{z}$ is attained in the population, the transshipment subproblem (TrP) for each $\mathbf{z}$ is solved to optimality to obtain the corresponding objective value.

4.4. Step 3: Generating a reference set

The main purpose of the reference set is to collect a set of good quality and most diverse solutions from the existing population. Let $b = b_1 + b_2$ be the cardinality of this set. The b_1 quality solutions are selected based on the objective function values, i.e. the solutions with lower total cost values are selected as quality solutions. The objective value of a solution is found by adding the fixed cost of the open transshipment points and the cost of transshipment between stages, i.e. objective value obtained by solving the problem (TrP). In order to find b_2 diverse solutions, we need to define a diversity measure. We use the distance calculated via the ℓ_1-norm as a diversity measure between two solutions, $\mathbf{z}_1$ and $\mathbf{z}_2$. That is

$$d(\mathbf{z}_1, \mathbf{z}_2) = \sum_{i=1}^{n} |z_1^i - z_2^i|.$$

We first include the quality solutions in the reference set. Then, in the population, we look for a solution that is not currently in the reference set and that maximizes the minimum distance to all solutions currently in the reference set. We include that solution in the reference set as the most diverse solution. We continue in this manner until we choose a total of b_2 diverse solutions.

4.5. Step 4: Generating the subsets from the reference set

This step consists of generating subsets of reference set solutions. The solutions in each subset are combined to generate new solutions in the next step. The general scatter search framework considers different sizes of subsets. The subsets may contain as few as two solutions of the reference set and as many as all of the solutions of the reference set. As stated in [30], there are several systematic ways to generate these subsets. The number of subsets has an impact on the overall duration of the algorithm as well as on the performance of the heuristic. Since the heuristic generates as many new solutions as the number of subsets, when this number is high, the heuristic takes a longer time to converge. However, the chances of generating better and more diverse solutions increase. In our implementation, we use two subset generators to obtain subsets of size 3 and 4, known as Type III and Type IV, respectively. Subsets of Type III are formed by including the best quality solution and a pair of other solutions from the reference set. The subset generator Type IV selects the two highest quality solutions from the reference set and joins them with pairs of remaining solutions. In the earlier analysis, we included subsets of Type II which correspond to the pairs of solutions taken from the reference set. We observed that dropping Type II subsets helped us achieve lower heuristic durations while sustaining solution quality. In Section 5, the results we report are generated using only Type III and Type IV subsets.

4.6. Steps 5 and 6: Creating new solutions from the subsets

We obtain a new solution from a subset by taking a linear combination of the solutions it includes. We use the objective values of the solutions as coefficients in this calculation. Let s be a subset under consideration for generating a new solution $\mathbf{V}$, and assume that it contains the solutions $\{\mathbf{s}_1, \ldots, \mathbf{s}_j\}$ with objective values $\{Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_j)\}$. In order to find the value of each element V^k in $\mathbf{V}$, we use the following formula:

$$V^k = \frac{\sum_{i=1}^{j} s_i^k * Z(\mathbf{s}_i)^{-1}}{\sum_{i=1}^{j} Z(\mathbf{s}_i)^{-1}}, \quad k = 1, \ldots, n, \tag{14}$$

where s_i^k is the value of the kth element in solution $\mathbf{s}_i$. Note that $Z(\mathbf{s}_i)^{-1}$ is the reciprocal of $Z(\mathbf{s}_i)$. Since our problem is a minimization problem, we use $Z(\mathbf{s}_i)^{-1}$ as multipliers so that a solution with a lower objective value receives higher weight. Clearly, an element of a combined solution can be a fractional value. In order to obtain a legitimate solution, we round a V^k to its closest binary value.

4.7. Steps 7 and 7*: Feasibility and improvement

The combination mechanism may generate infeasible solutions violating constraint (5) and/or may not provide minimum required total capacity, i.e. violating constraint (2). The heuristics we mentioned in Step 2 can be utilized

to attain feasibility of the new solution vector. Although, in Step 2, we implement the random location and random capacity heuristics to attain feasibility of the population of solutions, in Step 7, we use the fixed cost heuristic and the greedy capacity heuristic to satisfy constraints (5) and (2), respectively. The main reason is that, at this step of the heuristic, evolution of the solution is more important than its diversity. We retain the better quality solutions instead of the random solutions to achieve higher effectiveness.

After obtaining a new feasible solution, we can proceed in one of two ways. We can either directly proceed to Step 8 with the new solution or we can improve this new solution via local search before proceeding to Step 8. For improvement, we implement a local search routine that uses an *exchange* neighborhood on a solution vector $\mathbf{z}$. In the course of local search, we update a solution with the first better solution that we find in the solution space by randomly choosing pairs of elements to be exchanged. The general framework of local search is given in Display 2.

Display 2. Pseudo-code of the local search in the improvement method

Step 0: Let $\mathbf{z}_0 = (z_0^l \ldots, z_0^n)$ be a newly generated solution. Set $Z_{\text{best}} = Z(\mathbf{z}_0)$.
Step 1: $\mathbf{z} \leftarrow \mathbf{z}_0$
Step 2: Choose an element z_0^k
Step 3: Choose another element z_0^l where $z_0^l = 1 - z_0^k$
Step 4: $z^k \leftarrow 1 - z_0^k$
Step 5: $z^l \leftarrow 1 - z_0^l$
Step 6:
if $\sum_{j=1}^{n} W_j z^j < \sum_{i \in \mathscr{I}} D_i$ **then**
Go to Step 1
else
if $Z(\mathbf{z}) \leqslant Z_{\text{best}}$ **then**
$Z_{\text{best}} \leftarrow Z(\mathbf{z})$
$\mathbf{z}_0 \leftarrow \mathbf{z}$
Go to Step 1
else
Terminate with $\mathbf{z}_0$ and Z_{best} as the local optimum
end if
end if

4.8. Step 8: Updating the reference set

Once we obtain a new feasible solution (improved or not improved), we decide on its inclusion into the reference set, thus updating the reference set. There are three possible cases that we consider sequentially:

(1) If the objective value of the new solution is better (i.e. lower objective value) than any of the quality solutions in the reference set, we drop the worst solution among the quality solutions and add the new solution to the reference set.
(2) We calculate the distance between the new solution and each solution in the reference set. If it is more diverse (the most distant) than any of the diverse solutions in the reference set, we drop the worst solution (with respect to the distance criteria) among the diverse solutions and add the new solution to the reference set.
(3) If the new solution does not satisfy the first two conditions, then it is not eligible to be a member of the reference set. Then, we move to the next subset to generate another new solution.

4.9. Step 9: Path relinking

In general, a scatter search procedure is terminated after Step 8 when the reference set cannot be updated further and the best quality solution is reported. In order to improve this solution, we use a path relinking procedure. Path relinking

was originally suggested as an approach to integrate intensification and diversification strategies in the context of tabu search [33,34]. This approach generates new solutions by exploring trajectories that connect two high quality solutions. It starts with one of these solutions $\mathbf{z}_I$ (initiating solution), and generates a path (trajectory) of solutions towards the other solution $\mathbf{z}_G$ (guiding solution).

Display 3. Pseudo-code of the path relinking

Initialize: Choose the best quality solution $\mathbf{z}_G$ and choose one other quality solution $\mathbf{z}_I$
for $k = 1$ to n **do**
if $z_I^k = z_G^k$ **then**
$k \leftarrow k + 1$
else
$z_I^k \leftarrow z_G^k$
if $z_G^k = 0$ **then**
Find an intermediate solution **T** by changing the first zero entry to the right of k to 1 in $\mathbf{z}_I$ such that the capacity constraint (2) is not violated.
Calculate $Z(\mathbf{T})$.
if $Z(\mathbf{T}) \leqslant Z(\mathbf{z}_G)$ **then**
Add **T** to *Elite Set*.
end if
else
Find an intermediate solution **T** by changing the first non-zero entry to the right of k to 0 in $\mathbf{z}_I$ such that the capacity constraint (2) is not violated.
Calculate $Z(\mathbf{T})$.
if $Z(\mathbf{T}) < Z(\mathbf{z}_G)$ **then**
Add **T** to *Elite Set*.
end if
end if
end if
end for
if Elite Set $= \emptyset$ *then*
Return $\mathbf{z}_G$ as the solution of the heuristic
else
Return the solution **z** with minimum objective value in *Elite Set*
end if

In our path relinking framework, given in Display 3, we pick the best quality solution (the solution with the minimum objective value) as the guiding solution $\mathbf{z}_G$ and one of the other quality solutions in the reference set as the initiating solution $\mathbf{z}_I$. At each step of path relinking, we consider an element in $\mathbf{z}_I$ to convert it to the corresponding element in $\mathbf{z}_G$. For this purpose, at each iteration k, where $k = 1, \ldots, n$, where n is the size of $\mathbf{z}_I$ and $\mathbf{z}_G$, if $z_I^k \neq z_G^k$, then we make the assignment $z_I^k \leftarrow z_G^k$. With this change, we have either $p - 1$ transshipment points (if z_I^k was 1) or $p + 1$ transshipment points (if z_I^k was 0). We need to establish feasibility by changing one of the elements z_I^t, where $t \in \{k + 1, \ldots, n\}$. In this neighborhood, we evaluate all possible solutions via updates $z_I^t \leftarrow 1 - z_I^k$. At this point, we know that this new solution satisfies the constraint requiring p transshipment points. However, we check to see if it also satisfies the total transshipment capacity requirement. If it does not, we ignore the solution generated and move to the next t. Otherwise, we obtain its objective value by solving (TrP). If the objective value is better than the guiding solution, then we add this solution to the *Elite Set*, increase the iteration number k and continue in this manner until $\mathbf{z}_I$ is equal to $\mathbf{z}_G$. After termination, if the *Elite Set* is empty, then the guiding solution is the best solution. Otherwise, we pick the best solution in the *Elite Set*.

Although it is not a necessary component of the scatter search procedure, path relinking among the best quality solution and the quality solutions improve the results in most cases. Implementing scatter search without path relinking provides inferior results; therefore, we choose to keep this component in our search framework.

5. Computational results

We perform two different sets of experiments to show the effectiveness of the heuristic in terms of solution quality and speed. In the first experiment, we generate 120 random data sets to represent small, medium and large instances of the problem. In these first experiments, we observe that the solution quality of the hybrid scatter search algorithm is significant. In the second experiment set, we focus on relatively realistic problems with larger instances and tighter capacities. For these cases, our heuristic performs well, both in terms of speed and solution quality.

5.1. Experiment set 1

We consider 6 classes of problems with 20 instances in each, thus a total of 120 instances. These instances are solved using the scatter search heuristic both without and with the local search improvement method. We abbreviate these methods as SS and SSLS, respectively. We implement and run the algorithms using C++ on a Pentium IV 3.2 GHz machine.

We construct 20 random problems according to the ranges of parameters in each of the 6 classes given in Table 1. The parameters are generated randomly using uniform distributions with corresponding ranges. In Table 1, the first column represents the range for the number of plants; the second column represents the range for the number of candidate transshipment points; the third column represents the range for the number of customers; and the fourth column represents the range for the number of required open transshipment points. The last column of Table 1 reports the average durations that CPLEX takes to reach optimality for the corresponding problem classes.

We generate the capacity parameters randomly according to the following criteria:

(1) Capacity of each plant is generated according to $U[0.5 * \text{avCap}, 2 * \text{avCap}]$ where $\text{avCap} = (\sum_{i \in \mathscr{I}} D_i)/K$ and D_i is the demand for customer $i \in \mathscr{I}$.
(2) Capacity of each transshipment point is generated according to $U[0.5 * \text{avWcap}, 2 * \text{avWcap}]$ where $\text{avWcap} = (\sum_{i \in \mathscr{I}} D_i)/p$.

The other parameters are generated using the following distributions:

(1) Demand for each customer is drawn from $U[90, 100]$.
(2) Unit transportation costs from plants to transshipment points and from transshipment points to customers are drawn from $U[1, 10]$.
(3) The fixed cost of locating a transshipment point is given by $f_j = TC * W_j/m$, $j \in \mathscr{J}$, where TC is the total transportation cost over all the linkages.

Table 1
Test problem classes (20 instances solved for each set)

Data set	K	n	m	p	Avg. CPLEX time
Class 1	[2,3]	[10,20]	[30,40]	[1,5]	0.513
Class 2	[2,5]	[20,30]	[30,40]	[1,5]	6.988
Class 3	[2,5]	[20,30]	[30,40]	[6,10]	10.898
Class 4	[2,5]	[30,50]	[40,100]	[1,5]	25.066
Class 5	[2,5]	[30,50]	[40,100]	[6,10]	112.042
Class 6	[2,5]	[30,50]	[100,300]	[6,10]	736.177

Table 2
Optimality gaps (%) and runtimes (s) for Experiment 1 (20 instances solved for each set)

Data set	Without improvement (SS)				With improvement (SSLS)			
	Ave. gap	Max. gap	SS time	No. opt	Ave. gap	Max. gap	SSLS time	No. opt
Class 1	0.00	0.00	4.48	20	0.00	0.00	3.95	20
Class 2	0.15	0.80	15.25	13	0.00	0.00	11.40	20
Class 3	0.57	1.77	35.99	9	0.12	0.66	33.99	15
Class 4	0.36	1.26	55.40	11	0.17	0.97	56.40	14
Class 5	0.78	1.92	86.02	3	0.23	0.90	109.16	12
Class 6	0.86	1.98	340.87	3	0.30	0.86	392.77	9

In Table 2, we report the summary of the results for the first set of experiments. For each class of problems, the average optimality gap and the maximum optimality gap as a percentage, the average heuristic duration in seconds, and the number of optimum solutions obtained are reported. In this experiment, we vary the scatter search parameters with respect to the problem size as follows: $h_{\max} = 5 + n$, $b_1 = 2 + p/2$, $b_2 = 2 + p/4$, and the number of restarts is 5, i.e. each instance is solved five times, each starting with a different seed solution. We pick the best of five runs for each instance as the heuristic solution.

In all of these problem classes, the average gap between the optimum solution and the SS heuristic without improvement is less than 0.9%. Even the maximum optimality gap among these problems is less than 2%, and we find the optimal result in 59 of the 120 cases. With SSLS, in which the improvement step is used in scatter search, the average optimality gap is less than 0.3%, the maximum gap is less than 1% and, we find the optimum solution in 90 of the 120 problems. Since scatter search is a population based search heuristic, in small cases the duration of the heuristic is comparable to the CPLEX duration. However, for large cases (problem classes 5 and 6), the heuristic durations outperform the CPLEX durations.

Note that in this experiment, in order to find the best parameter set, we experimented with different scatter search parameter sets, including fixed parameter values as low as $h_{\max} = 10$, $b_1 = 2$ and $b_2 = 2$. Fixed parameter values, although they perform well for small cases, cannot guarantee a good performance in terms of solution quality for larger cases. Therefore, we set the parameters as a function of the problem size, i.e. the parameters K, n, m, and p. We experimented with different combinations of these input parameters and obtained better results when scatter search parameters are a function of the number of transshipment points available (n) and required (p) as given in the calculation of $h_{\max}$, b_1 and b_2 above.

5.2. Experiment set 2

We observe that, for the relatively large instances considered in experiment set 1, the average duration of the scatter search is better than the average duration for obtaining optimum solutions with CPLEX. Therefore, it is of further interest to see how the heuristic performs for larger instances when compared to the CPLEX solutions in terms of solution quality and speed. For this purpose, we generate 160 large instances with tight capacities, i.e. 16 data sets with 10 problem instances each. The parameters, except the capacities of the plants and the transshipment points, in each instance of a data set are generated randomly as in experiment set 1. The capacity of each plant, except the last one, is generated according to $U[0.8 * \text{avCap}, 1.2 * \text{avCap}]$ where $\text{avCap} = (\sum_{i \in \mathcal{J}} D_i)/K$. The last plant's capacity is adjusted so that the total demand is equal to the total capacity of the plants. On the other hand, the capacity of each transshipment point is generated according to $U[0.8 * \text{avWcap}, 1.2 * \text{avWcap}]$ where $\text{avWcap} = (\sum_{i \in \mathcal{J}} D_i)/p$. We use the SSLS heuristic with 5 as the number of restarts as before, and the search parameter values, which vary as the problem size changes, $h_{\max} = 10 + n/2$, $b_1 = 2 + 2 * p$ and $b_2 = 2 + p/2$ are determined after some fine-tuning.

The results are summarized in Table 3 where we define a data set with its associated input parameter values including the number of customers (m), the number of potential transshipment points (n), the number of facilities to be opened (p) and the number of plants (K). Note that the problem instances we consider are larger than the problem instances previously used for similar problems in the literature. The largest instances employed for test purposes by Klose [25] and [26] include up to $m = 500$ customers, $n = 50$ potential transshipment points, and $K = 10$ plants. In [27], one of

Table 3
Optimality gaps (%) and runtimes (s) of instances for Experiment 2 (10 instances solved for each set)

Data set	m	n	p	K	Ave. gap	Max. gap	No. opt	Avg. time	Avg. Cplex time	Instances with Max. Cplex time		
										Cplex	SSLS	Gap
1	300	100	10	5	0.72	1.60	3	693.54	1283.72	2033.61	1102.79	0.99
2	300	100	10	10	0.68	1.62	4	748.15	1262.99	3093.81	707.74	0.00
3	300	100	20	5	0.91	2.38	3	900.29	1383.00	1670.72	679.51	0.00
4	300	100	20	10	1.14	2.86	2	954.73	1591.88	1913.57	784.38	1.16
5	300	200	10	5	1.15	2.56	4	1214.93	5243.91	18 981.80	1038.67	2.52
6	300	200	10	10	0.95	2.31	4	1196.26	4636.52	7799.15	1035.12	0.00
7	300	200	20	5	0.94	2.16	4	954.63	4218.85	6129.78	643.56	0.00
8	300	200	20	10	1.22	2.48	2	1064.56	5380.50	9813.64	895.64	0.48
9	500	100	10	5	0.73	1.86	3	994.70	3393.05	4780.08	508.59	0.74
10	500	100	10	10	0.77	1.66	2	1114.54	4136.34	11 439.50	1726.62	0.60
11	500	100	20	5	0.75	1.51	4	1126.66	4148.52	5846.72	983.77	1.40
12	500	100	20	10	0.75	1.87	3	1377.45	5333.24	8477.78	982.95	0.86
13	500	200	10	5	1.48	2.71	3	1357.11	10 784.95	24 947.80	1043.85	0.00
14	500	200	10	10	1.50	2.86	3	1411.08	13 222.01	29 712.60	1789.00	2.59
15	500	200	20	5	1.52	2.76	3	1493.83	8331.50	10 442.17	1746.07	0.49
16	500	200	20	10	1.68	2.96	2	1558.82	9217.75	14 348.37	2043.65	2.17

the studies most relevant to our own work, the maximum problem size has the parameter values $m = 60$, $n = 40$, $p = 8$, and $K = 20$.

In Table 3, we report the average optimality gap, the maximum optimality gap, the number of optimum solutions obtained with SSLS, and the average durations of SSLS and CPLEX for 10 instances in each data set. In the last three columns of Table 3, we report the maximum duration to achieve the optimum solution for all instances of a data set along with the corresponding SSLS duration and the optimality gap for that particular instance. In all 160 instances, the average duration of the hybrid scatter search heuristic is less than the average duration of the CPLEX solution. Especially, in larger instances, the discrepancy between the average durations is larger; see, for example, data sets 13, 14, 15, 16. Some of these instances can be very time consuming to solve by CPLEX. For example, there is an instance in data set 13 which is solved in 24947.8 s with CPLEX whereas it takes 1043.85 s with the SSLS heuristic to solve it to optimality. Moreover, in terms of solution quality, the SSLS heuristic is effective in solving large instances. The largest average optimality gap we obtain over all of the data sets is less than 1.7%. The average of all of the optimality gaps in 160 instances is 1.03%.

6. Conclusions

In this paper, we developed an efficient hybrid scatter search framework that incorporates path relinking for locating capacitated transhipment points. We also obtained promising computational results that show the effectiveness of the procedure. Scatter search performance depends on the selection of the initial search parameters as well as the implementation of each step. We observed that seemingly insignificant changes in scatter search parameter values can contribute to the solution quality and decrease the overall duration of the heuristic. We also observed that if premature convergence is handled, the scatter search framework can provide improved solutions even after many non-improving iterations. This is mainly due to its inherent diversification characteristic. In addition, due to the inherent randomness of the procedure, a multi-start approach appears very effective.

The model and the solution procedure presented in this paper can be extended in several ways. Our ongoing research includes two such generalizations and further improvements of the solution procedure. The first one involves developing a scatter search framework for problems with single-sourcing of customers from warehouses, and the second one considers location and assignment decisions in capacitated multi-commodity location problems in similar settings.

Acknowledgements

The authors thank the anonymous referees for their helpful comments and suggestions.

References

[1] Hitchcock FL. The distribution of a product from several sources to numerous localities. Journal of Mathematical Physics 1941;20:224–30.
[2] Laguna M., Martí R. editors. Scatter search: methodology and implementations in *C*. Boston, MA: Kluwer Academic Publishers; 2003.
[3] Rardin RL, Uzsoy R. Experimental evaluation of heuristic optimization. Journal of Heuristics 2001;7:261–304.
[4] Kariv O, Hakimi L. An algorithmic approach to network location problems: part ii: the *p*-medians. SIAM Journal of Applied Mathematics 1979;37(3):539–60.
[5] Daskin MS. Network and discrete location: models, applications and algorithms. New York, NY: Wiley; 1995.
[6] Correa ES, Steiner MTA, Freitas AA, Carnieri C. A genetic algorithm for solving a capacitated *p*-median problem. Numerical Algorithms 2004;35(2–4):373–88.
[7] Lorena LAN, Senne ELF. A column generation approach to capacitated *p*-median problems. Computers and Operations Research 2004;31(6): 863–76.
[8] Horn M. Analysis and computational schemes for *p*-median heuristics. Environment and Planning A 1996;28(9):1699–708.
[9] Rosing KE, Hodgson MJ. Heuristic concentration for the *p*-median: an example demonstrating how and why it works. Computers and Operations Research 2002;29(10):1317–30.
[10] Alp O, Erkut E, Drezner Z. An efficient genetic algorithm for the *p*-median problem. Annals of Operations Research 2003;122(1–4):21–42.
[11] Lim A, Xu Z. A fixed-length subset genetic algorithm for the *p*-median problem. Lecture Notes in Computer Science 2003;2724:1596–7.
[12] Estivill-Castro V, Torres-Velazquez R. Hybrid genetic algorithm for solving the *p*-median problem. Lecture Notes in Artificial Intelligence 1999;1585:18–25.
[13] Rolland E, Schilling DA, Current JR. An efficient tabu search procedure for the *p*-median problem. European Journal of Operational Research 1997;96(2):329–42.
[14] Levanova TV, Loresh MA. Algorithms of ant system and simulated annealing for the *p*-median problem. Automation and Remote Control 2004;65(3):431–8.
[15] Garcia-Lopez F, Melian-Batista B, Moreno-Perez JA. Parallelization of the scatter search for the *p*-median problem. Parallel Computing 2003;29(5):575–89.
[16] Díaz JA, Fernández E. Hybrid scatter search and path relinking for the capacitated *p*-median problem. European Journal of Operational Research 2006;169:570–85.
[17] Resende MGC, Werneck RF. A hybrid heuristic for the *p*-median problem. Journal of Heuristics 2004;10(1):59–88.
[18] Hribar M, Daskin MS. A dynamic programming heuristic for the *p*-median problem. European Journal of Operational Research 1997;101(3): 499–508.
[19] Geoffrion AM, Graves GW. Multicommodity distribution system design by benders decomposition. Management Science 1974;20(5):822–44.
[20] Hindi KS, Basta T. Computationally efficient solution of a multi-product, two-stage distribution-location problem. Journal of Operations Research Society 1994;45:1316–23.
[21] Pirkul H, Jayaraman V. A multi commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution. Computers and Operations Research 1998;25:869–78.
[22] Jayaraman V, Pirkul H. Planning and coordination of production and distribution facilities for multiple commodities. European Journal of Operational Research 2001;133:394–408.
[23] Tragantalerngsak S, Holt J, Ronnqvist M. An exact method for the two-echelon, single source, capacitated facility location problem. European Journal of Operational Research 2000;123:473–89.
[24] Tragantalerngsak S, Holt J, Ronnqvist M. Lagrangian relaxation heuristic for two-echelon, single source, capacitated facility location problem. European Journal of Operational Research 1997;102:611–25.
[25] Klose A. An lp-based heuristic for two-stage capacitated facility location problems. Journal of the Operational Research Society 1999;50: 157–66.
[26] Klose A. A Lagrangian relax-and-cut approach for two-stage capacitated facility location problems. European Journal of Operational Research 2000;126:408–21.
[27] Marin A, Pelegrin B. A branch-and-bound algorithm for the transportation problem with location of *p* transshipment points. Computers and Operations Research 1997;24(7):659–78.
[28] Marin A, Pelegrin B. Applying lagrangian relaxation to the resolution of two-stage location problems. Annals of Operations Research 1999;86:179–98.
[29] Gao L, Robinson EP. A dual based optimization procedure for the two-echelon uncapacitated facility location problem. Naval Research Logistics 1992;39:191–212.
[30] Glover F. Heuristics for integer programming using surrogate constraints. Decision Sciences 1977;8:156–66.
[31] Martí RR, Laguna M, Glover F. Principles of scatter search. European Journal of Operational Research 2006;169:359–72.
[32] Glover F. Artificial evolution, lecture notes in computer science, vol. 1363. Berlin: Springer; 1998.
[33] Glover F. Tabu search for nonlinear and parametric optimization. Discrete and Applied Mathematics 1994;49:231–55.
[34] Glover F, Laguna M. Tabu search. Norwell, MA: Kluwer Academic Publishers; 1997.