

# A multi-exchange heuristic for the single source capacitated facility location problem

R.K. Ahuja

*Department of Industrial & Systems Engineering, University of Florida, Gainesville, FL, USA*

*ahuja@ufl.edu*

J.B. Orlin

*Sloan School of Management, MIT, Cambridge, MA, USA, jorlin@mit.edu*

S. Pallottino

M.P. Scaparra

M.G. Scutellà

*Dipartimento di Informatica, Università di Pisa, Pisa, Italy, [pallottino,scaparra,scutella@di.unipi.it]*

## Abstract

We present a very large scale neighborhood (VLSN) search algorithm for the capacitated facility location problem with single-source constraints. The neighborhood structures are induced by customer multi-exchanges and by facility moves. We consider both traditional single-customer multi-exchanges, detected on a suitably defined *customer improvement graph*, and more innovative multi-customer multi-exchanges, detected on a *facility improvement graph* dynamically built through the use of a greedy scheme. Computational results for some benchmark instances are reported, which demonstrate the effectiveness of the approach for solving large-scale problems. A further test on real data involving an Italian factory is also presented.

**Keywords:** location problems, large scale optimization, neighborhood search, negative cycles

## 1 Introduction

The single source capacitated facility location problem, usually referred to as SSCFLP, is a well-known location problem which finds large applicability in many sectors, such as distribution systems planning or telecommunication networks design. The objective in this problem is to locate a number of facilities (e.g., plants, warehouses or concentrators), that have to serve at minimum cost a set of customers. The cost includes fixed charges for opening the facilities and transportation costs for satisfying customer demands. Each customer  $j$  has an associated demand,  $w_j$ , that must be served

by a single facility. Facilities can be located only at some prespecified sites, and there is a limit,  $s_i$ , to the total demand that a facility located at a site  $i$  can meet. The costs  $c_{ij}$  of supplying the demand of a customer  $j$  from a facility established at location  $i$ , as well as the fixed costs  $f_i$  for opening a facility at  $i$ , are known. Let

$$x_{ij} = \begin{cases} 1, & \text{if customer } j \text{ is assigned to a facility located at } i, \\ 0, & \text{otherwise;} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{if a facility is located at candidate site } i, \\ 0, & \text{otherwise.} \end{cases}$$

Then, the problem can be stated mathematically as follows:

$$\min \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \quad (1)$$

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} = 1 \quad \forall j \in J \quad (2)$$

$$\sum_{j \in J} w_j x_{ij} \leq s_i y_i \quad \forall i \in I \quad (3)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J \quad (4)$$

$$y_i \in \{0, 1\} \quad \forall i \in I \quad (5)$$

where  $I = \{1, \dots, n\}$  is the set of potential locations and  $J = \{1, \dots, m\}$  is the set of customers. The expression (1) establishes that the objective of the problem is the minimization of the fixed costs plus the shipment costs. Assignment constraints (2) ensure that each customer is allocated to exactly one center, while constraints (3) enforce the total demand of customers assigned to a facility not to exceed its maximum capacity.

SSCFLP belongs to the class of NP-hard problems. Most of the existing work for solving it has been developed using a Lagrangean heuristic framework. Several authors have considered relaxing the assignment constraints (2). Their methods differ in the way lower bounds and feasible solutions are generated. For instance, Barcelo and Casanova (1984) developed a two phase primal heuristic procedure. Pirkul (1987) separated the Lagrangean subproblem into  $n$  knapsack problems. From their solutions, he obtained a lower bound to the original problem with a set of open facilities, which is then used to build a feasible solution. Sridharan (1991), Hindi and Pieńkosz (1999), and Rönnqvist et al. (1999) computed lower bounds along the same line. However, Sridharan (1991) set up a single source transportation problem to find feasible solutions. Hindi and Pieńkosz (1999) employed a restricted neighborhood search. Rönnqvist et al. (1999) used a repeated matching algorithm.

Klincewicz and Luss (1986) devised a Lagrangean heuristic based upon relaxing the capacity constraints (3), and solved the resulting Lagrangean subproblem through the dual ascent heuristic of Erlenkotter (1978). Beasley (1993) proposed a Lagrangean heuristic based upon relaxing both constraints (2) and (3). Agar and Salhi (1998) introduced efficient modifications to Beasley approach. The only attempts to solve SSCFLP to optimality are due to Neebe and Rao (1983), who proposed a branch and bound scheme based on the setup of a partitioning problem, and to Holmberg et al. (1999), who incorporated the repeated matching Lagrangean heuristic described in Rönnqvist et al. (1999) into a branch and bound framework.

The objective of this work is to propose a very large scale neighborhood (VLSN) technique for SSCFLP. The method consists of the alternating use of customer exchanges and facility moves. Customer exchanges mainly affect the customer assignment among facilities already located, and are detected by searching for special paths or cycles on suitably defined improvement graphs. Facility moves aim at finding improving configurations of the open facilities. The neighborhood structures thus obtained are embedded in a local improvement algorithm. Restart mechanisms are also considered.

The rest of the paper is organized as follows. In Section 2, we set up some useful notations. Section 3 introduces some VLSN structures for SSCFLP. In Section 4, we describe the neighborhood induced by single-customer multi-exchanges. In Section 5, we address the more general multi-customer multi-exchange neighborhood. To the best of our knowledge, this kind of moves was never investigated before. Section 6 is dedicated to the description of the facility neighborhood. Section 7 addresses the main implementation issues, while Section 8 reports the empirical results for a large suite of test problems. Finally, Section 9 presents conclusions and suggestions.

## 2 Notation

Let  $S$  be a feasible solution to the problem (1)-(5). The set  $S$  can be represented as a partition of the customer set  $J$  into  $n$  subsets, i.e.,  $S = \{S_1, S_2, \dots, S_n\}$ , where each subset  $S_i$ ,  $i = 1, \dots, n$ , corresponds to a potential facility site and contains the set of customers assigned to that facility in  $S$ . Eventually,  $S_i$  can be empty if no facility is established at location  $i$ .

The cost of each subset  $S_i$ ,  $i = 1, \dots, n$ , is given by

$$C(S_i) = \begin{cases} \sum_{j \in S_i} c_{ij} + f_i, & \text{if } S_i \neq \emptyset, \\ 0, & \text{otherwise;} \end{cases} \quad (6)$$

and the cost of the whole partition  $S$  is

$$C(S) = \sum_{i=1}^n C(S_i). \quad (7)$$

A solution  $S$  is feasible if the following inequality holds for  $i = 1, \dots, n$ :

$$W(S_i) = \sum_{j \in S_i} w_j \leq s_i. \quad (8)$$

The *residual capacity*,  $r_i$ , of each subset  $S_i$ , i.e., the capacity still available at a facility located at  $i$ , is defined as:

$$r_i = s_i - W(S_i) \geq 0. \quad (9)$$

In the following, we will denote by  $I(S)$  the set of open facilities in the current solution  $S$ , and by  $F(j)$ ,  $j \in J$ , the facility serving customer  $j$ .

### 3 Very large scale neighborhood structures

The first VLSN structure we will consider for solving SSFLP is the customer neighborhood. Given a solution  $S$  to SSCFLP, the customer neighborhood of  $S$  is defined as the set of new solutions obtainable from  $S$  by exchanging customers among facilities in a cyclic manner. Such a neighborhood is constructed by moving among the facilities either single customers or subsets of customers conveniently chosen. More precisely, the multi-customer neighborhood is defined in terms of multi-customer cyclic or path exchanges as follows.

Let  $Q = \{i_1, i_2, \dots, i_q\}$  denote a sequence of indexes of  $q$  subsets of the partition  $S = \{S_1, S_2, \dots, S_n\}$ . A *multi-customer cyclic exchange* is a sequence  $U = \{U_{i_1}, U_{i_2}, \dots, U_{i_{q-1}}, U_{i_q}\}$  of  $q$  sets of customers such that:

- i)  $U_{i_r} \subseteq S_{i_r}$  and  $U_{i_r} \neq \emptyset$ , for  $r = 1, \dots, q$ ;
- ii) each subset  $U_{i_r}$  of the sequence belongs to a different subset  $S_i$ .

A *multi-customer path exchange* is a special multi-customer cyclic exchange where  $U_{i_q} = \emptyset$ . Any multi-customer cyclic or path exchange  $U$  uniquely defines a new solution  $S' = \{S'_1, \dots, S'_n\}$  in the neighborhood of  $S$ , obtained by shifting customer subsets along the sequence. Formally:

$$S'_{i_r} = \begin{cases} S_{i_r}, & \text{if } i_r \notin Q, \\ S_{i_r} \cup U_{i_{r-1}} \setminus U_{i_r}, & \text{if } i_r \in Q \text{ and } r = 2, \dots, q, \\ S_{i_r} \cup U_{i_q} \setminus U_{i_r}, & \text{if } i_r \in Q \text{ and } r = 1. \end{cases} \quad (10)$$

A multi-customer cyclic or path exchange  $U$  is *feasible* if the capacity constraints are satisfied for each partition set of the new solution  $S'$ , and *profitable* if the new solution  $S'$  is such that  $C(S') < C(S)$ . The set of all the new solutions obtainable from  $S$  through a feasible exchange defines the *multi-customer neighborhood* of  $S$ . Since the size of this neighborhood can be excessively large, our approach tries to reduce the number of possible clusters by limiting their size to a maximum of  $K$  elements, where  $K$  is an algorithm parameter. Furthermore, the neighborhood is searched only partially using a heuristic approach which detects special negative cost cycles in a suitably defined graph, called the *facility improvement graph*. The search is performed in such a way to generate only those clusters which are more likely to produce improving neighbor solutions. This task is accomplished in the dynamic generation of the facility improvement graph.

The *single-customer neighborhood* is a special case of the multi-customer neighborhood, obtained when only moves of single customers are allowed among facilities. The definition of *single-customer cyclic* or *path exchanges* can be easily derived from the multi-customer exchange definitions by imposing  $|U_{i_r}| = 1, i = 1, 2, \dots, q$  for cyclic exchanges, and  $|U_{i_r}| = 1, i = 1, 2, \dots, q - 1$  and  $|U_{i_q}| = 0$  for path exchanges. Also in this special case, the neighborhood is searched only partially, using a heuristic approach which detects special negative cost cycles in a suitably defined graph. However, due to the inherent simpler definition of the single-customer moves, the graph associated with these moves, called the *customer improvement graph*, is defined and explored in a different (and more classical) way. The customer improvement graph will be outlined in the following section, whereas the description of the more complex facility improvement graph will follow in Section 5.

The second VLSN structure we considered is the neighborhood induced by facility moves, that is the set of solutions obtainable by changing the state of one facility from close to open or vice versa, or by swapping used and unused sites. A facility move can be performed by leaving the customer assignment almost unchanged or, occasionally, may require a complete reallocation of all customers. In practice, we only searched a restricted facility neighborhood, in the sense that only moves involving promising facilities are attempted.

## 4 Single-customer multi-exchanges

Let us describe how to build and explore the single-customer neighborhood.

## 4.1 The Customer Improvement Graph

Given a solution  $S$ , the *customer improvement graph* relative to  $S$  is a directed graph  $G^c(S) = (N^c(S), A^c(S))$  defined as follows. The set of nodes  $N^c(S)$  contains a *regular node*,  $j$ , for each customer  $j \in J$ , a *pseudonode*,  $p_i$ , for each facility location  $i$ , and an *origin* node,  $o$ . The pseudonodes are added to  $N^c(S)$  to model exchanges along paths rather than cycles. The set of arcs  $A^c(S)$  contains an arc  $(j, k)$  for each pair of regular nodes  $j$  and  $k$  in  $N^c(S)$ , provided that customers  $j$  and  $k$  are assigned to two different facilities in  $S$ , i.e.,  $F(j) \neq F(k)$ , and that after the insertion of  $j$  in  $S_{F(k)}$  and the removal of  $k$  from it, the capacity of facility  $F(k)$  is not exceeded, i.e.,  $w_j - w_k \leq r_{F(k)}$ . An arc connecting two nodes  $j$  and  $k$ , in fact, signifies that customer  $j$  leaves facility  $F(j)$  and is assigned to facility  $F(k)$ , which in turn relinquishes customer  $k$ .

The cost  $\alpha_{jk}$  of the arc  $(j, k)$  reflects the cost variation of facility  $F(k)$  and it is therefore defined as

$$\alpha_{jk} = c_{F(k)j} - c_{F(k)k}. \quad (11)$$

$A^c(S)$  also contains an arc  $(j, p_i)$  from each regular node  $j$  to each pseudonode  $p_i$ , provided that customer  $j$  is not served by facility  $i$  in  $S$ , i.e.,  $F(j) \neq i$ , and that customer  $j$ 's demand does not exceed the residual capacity of facility  $i$ , i.e.,  $w_j \leq r_i$ . An arc  $(j, p_i)$  signifies that customer  $j$  is moved to facility  $i$  but no customer is relinquished from  $i$ , and its cost is simply

$$\alpha_{jp_i} = c_{ij}. \quad (12)$$

Finally, we have an arc  $(o, j)$  from the origin to each regular node, and an arc  $(p_i, o)$  from each pseudonode back to the origin. Each arc  $(o, j)$  has cost:

$$\alpha_{oj} = \begin{cases} -c_{F(j)j}, & \text{if } |S_{F(j)}| > 1, \\ -c_{F(j)j} - f_{F(j)}, & \text{if } |S_{F(j)}| = 1. \end{cases} \quad (13)$$

Each arc  $(p_i, o)$  has cost:

$$\alpha_{p_i, o} = \begin{cases} f_i, & \text{if } S_i = \emptyset, \\ 0, & \text{otherwise.} \end{cases} \quad (14)$$

## 4.2 Negative subset-disjoint cycles

Let us now define negative subset-disjoint cycles in the customer improvement graph (Thompson and Orlin 1989).

**Definition 4.1** A directed cycle  $(n_1, \dots, n_q)$  in the improvement graph  $G^c(S)$  associated with solution  $S$  is a negative subset-disjoint cycle if each one of its nodes, except the origin (if present), is associated with a different facility location and if the sum of the costs of its arcs is negative.

We can now state the equivalence between the problem of detecting a negative subset-disjoint cycle in  $G^c(S)$  and the problem of finding a feasible profitable multi-exchange for  $S$ .

**Property 4.1** There is a one-to-one correspondence between the set of feasible cyclic [path] exchanges relative to the current solution  $S$  and the set of subset-disjoint cycles in  $G^c(S)$ . Furthermore, each cyclic [path] exchange is a profitable exchange if and only if the corresponding subset-disjoint cycle is negative.

In particular, the fact that the customer improvement graph only includes feasible arcs with respect to the capacity constraints ensures that each exchange corresponding to a cycle in  $G^c(S)$  is a feasible exchange. Furthermore, the way we define the costs of the arcs in  $A^c(S)$  guarantees that the cost of each cycle in  $G^c(S)$  accurately reflects the cost variation of the solution due to the corresponding customer exchanges. Hence, if the cycle cost is negative, the associated exchange is profitable, and vice-versa. The node sequence  $(n_1, \dots, n_q)$  corresponds to a path exchange if the subset-disjoint cycle contains the origin and a pseudonode; otherwise, it corresponds to a cyclic exchange.

Figure 1 shows the customer improvement graph associated with the solution  $S$  of a simple location problem with 2 facilities and 3 customers. A single-customer cyclic exchange involving customers  $j_1$  and  $j_3$  (a) and its corresponding cycle in  $G^c(S)$  (b) are also illustrated.

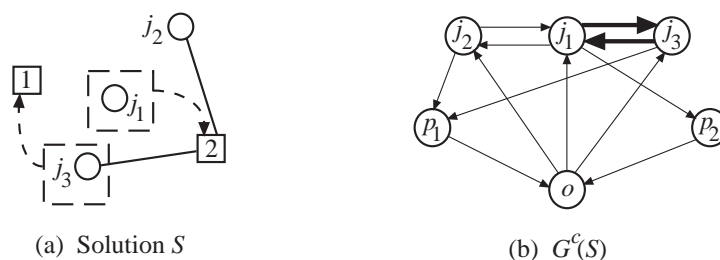


Figure 1: Customer Improvement Graph

### 4.3 Search for negative subset-disjoint cycles

As a consequence of Property 4.1, the problem of finding an improving move in the single-customer neighborhood can be reformulated as the problem of identifying negative-cost subset-disjoint cycles

in  $G^c(S)$ . This problem is known to be NP-complete (Thompson and Orlin 1989). However, the heuristic method proposed by Ahuja et al. (2001) for the capacitated minimum spanning tree problem is able to detect such cycles very effectively, missing some of them only rarely. Hence, we used the same heuristic in our solution approach. The algorithm is based on a modification of the label-correcting approach for the shortest path problem. The main changes are introduced to check for the path subset-disjointness. The reader is referred to (Ahuja et al. 2001, Ahuja et al. 2002) for further details concerning this algorithm and for a survey on the very large neighborhood techniques.

## 5 Multi-customer multi-exchanges

Previous scientific works dealing with multi-exchange techniques (see, for example, Thompson and Orlin 1989, Frangioni et al. 2000, Ahuja et al. 2001, Thompson and Psaraftis 1993), usually adopted a standard methodology for building the improvement graph, which is basically the one we have just described for the single-customer moves. Namely, each node of the graph corresponds to an element of the partition that can be moved, and each arc represents the transfer of the “tail” element toward the partition subset currently containing the “head” element. This scheme becomes impractical when modeling multi-customer multi-exchanges, due to the exponential number of clusters that should be considered for movement. We therefore resort to a different variant to limit the graph growth, which consists of constructing the improvement graph dynamically and uses a greedy scheme to select only promising customer subsets to be moved. As it will become more explicit in the following sections, the exact meaning of each arc is established and becomes available only during the search phase, since only at that stage the moving customer subset is chosen, on the basis of the information gathered along the path up to that arc.

### 5.1 The Facility Improvement Graph

The *facility improvement graph*  $G^f(S) = (N^f(S), A^f(S))$ , used to model multi-customer multi-exchanges, has a *regular node*  $i$  for each open facility, a *pseudonode*,  $p_h$ , for each facility location  $h \in I$ , and an *origin*,  $o$ .

The set of arcs  $A^f(S)$  can contain an arc  $(i, h)$  between each pair of regular nodes. Such an arc indicates that the facility established at location  $i$  relinquishes a set of up to  $K$  customers which are then assigned to the facility located at  $h$ . It also implies the subsequent removal of a subset of at most  $K$  customers from  $S_h$ . The selection approach for choosing a convenient subset of up to

$K$  customers to be transferred from  $S_i$  to  $S_h$ , denoted as  $U_{ih}$ , will be described in the next section. Such a subset may not exist, in which case  $(i, h)$  is not included in the arc set. If it is included, its costs  $\beta_{ih}$  is defined as

$$\beta_{ih} = \sum_{j \in U_{ih}} c_{hj} - \sum_{j \in U_{ih}} c_{ij}. \quad (15)$$

The arc set  $A^f(S)$  also contains an arc  $(i, p_h)$  from each regular node  $i$  to each pseudonode  $p_h$ , with  $i \neq h$ . Such an arc implies that no subset of customers is relinquished from facility  $h$ , and its cost  $\beta_{ip_h}$  is given by equation (15) as well. Additionally,  $G^f(S)$  includes a set of directed arcs  $(o, i)$  which connect the origin to every regular node, plus a set of directed arcs  $(p_h, o)$  from each pseudonode  $p_h$  back to the origin. Each arc  $(o, i)$  has null cost, unless the subsequent removal of a customer subset from  $S_i$  leaves facility  $i$  empty, in which case its cost is  $-f_i$ . Such a condition can be checked only at running time, when the customer subset leaving facility  $i$  to another facility becomes available. Finally, the cost of each arc  $(p_h, o)$  is null if facility  $h$  has already customers assigned to it; otherwise, it is equal to the fixed cost,  $f_h$ , for opening it.

The inclusion of the origin  $o$  and of the pseudonodes  $p_h$  in the facility improvement graph has the usual rationale of modeling path exchanges. The use of path exchanges is particularly relevant in the multi-customer neighborhood since it frequently leads to changes in the facility locations.

Figure 2 represents the facility improvement graph associated with the solution  $S$  of a problem with 9 customers and 4 facility sites. It also shows the path exchange corresponding to the cycle  $\{o, 3, 1, 2, p_4\}$ , which involves closing facility 3 and opening facility 4.

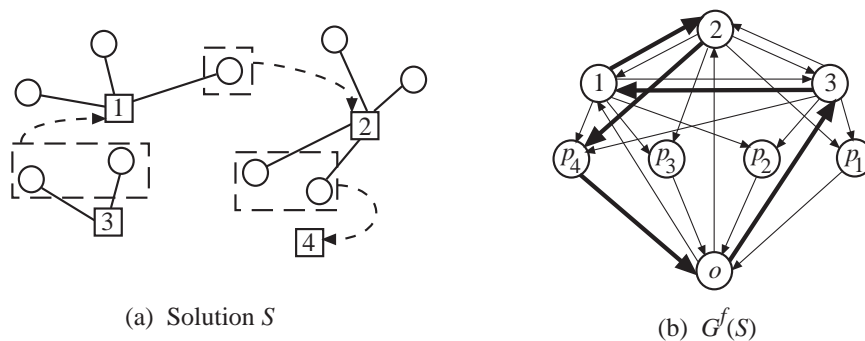


Figure 2: Facility Improvement Graph

## 5.2 Negative subset-disjoint cycles

**Definition 5.1** A sequence of nodes  $(n_1, \dots, n_q)$  in the facility improvement graph  $G^f(S)$ , associated with solution  $S$ , is a negative subset-disjoint cycle if each one of its nodes, except the origin (if

present), is associated with a different facility location, and if its overall cost is negative. The cost of a cycle is simply defined as the sum of the costs of its arcs.

The problem of finding some feasible profitable multi-customer multi-exchanges for  $S$  can then be translated into the problem of detecting negative subset-disjoint cycles in  $G^f(S)$ . The correspondence between cycles in  $G^f(S)$  and path or cyclic exchanges is based on the property that the neighborhood search algorithm not only finds subset-disjoint cycles, but also produces for each cycle a sequence of customer subsets which move among the facilities involved in it. Further, the method used for determining such customer subsets guarantees that their transfer along the sequence produces only feasible exchanges. Therefore:

**Property 5.1** *Each sequence of customer subsets uniquely identified by a subset disjoint cycle in  $G^f(S)$  corresponds to a feasible cyclic or path exchange with respect to  $S$ . Furthermore, if the subset-disjoint cycle is negative, the corresponding cyclic or path exchange is a profitable exchange.*

The customer subset sequence corresponds to a path exchange if the subset-disjoint cycle contains the origin and a pseudonode; otherwise, it corresponds to a cyclic exchange. It has to be noted that, in this case, the correspondence between cycles in  $G^f(S)$  and path or cyclic exchanges for  $S$  is not a one-to-one correspondence. In fact, each feasible subset-disjoint cycle in  $G^f(S)$  corresponds to a feasible exchange, but not all the possible feasible exchanges are mapped onto cycles in the facility improvement graph. The lack of this correspondence in both directions depends on our restriction to consider only a limited number of moving customer subsets.

### 5.3 Selection of moving customers

Let us introduce the definition of *candidate leaving subsets*.

**Definition 5.2** *Given a customer subset  $S_h$  and a cluster of customers  $U_e \notin S_h$ , a subset  $U_l$  of  $S_h$  is a candidate leaving subset w.r.t.  $S_h$  and  $U_e$  if it satisfies the following condition:  $W(U_l) \geq W(U_e) - r_h$ .*

In other words, a subset of customers  $U_l$  is a candidate leaving subset w.r.t.  $S_h$  and an entering subset  $U_e$  if the capacity limit of facility  $h$  is not violated after the insertion of  $U_e$  in  $S_h$  and the removal of  $U_l$  from it. Assume that, during the search for a subset-disjoint cycle, node  $h$  is reached from node  $i$ , and that we know the subset of customers  $U_{ih}$  entering  $S_h$  from  $S_i$ , as depicted in Figure 3. Our customer selection policy consists in determining, for each node  $l$  reachable from

node  $h$ , a subset of customer,  $U_{hl} \in S_h$ , to be reassigned from facility  $h$  to facility  $l$  in such a way to minimize the total cost for the reassignment.  $U_{hl}$  must be a candidate leaving subset w.r.t.  $S_h$  and  $U_{ih}$ , and can contain at most  $K$  customers.

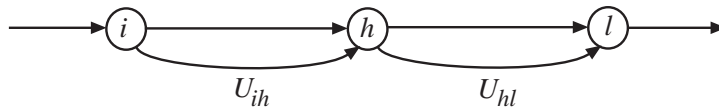


Figure 3: Subset Selection

Such a selection problem,  $SP$ , can be mathematically stated as follows, where constraint (18) guarantees that the selected customer subset is a candidate leaving subset with respect to the entering subset  $U_{ih}$  and  $S_h$ . The objective is to minimize the reassignment costs.

$$\min \sum_{j \in S_h} (c_{lj} - c_{hj})x_j \quad (16)$$

$$\text{s.t.} \quad \sum_{j \in S_h} x_j \leq K \quad (17)$$

$$\sum_{j \in S_h} w_j x_j \geq W(U_{ih}) - r_h \quad (18)$$

$$x_j \in \{0, 1\} \quad \forall j \in S_h \quad (19)$$

A solution to the problem  $SP$  uniquely determines a subset  $U_{hl}$  by simply setting  $U_{hl} = \{j \in S_h | x_j = 1\}$ .

There are two situations in which the selection of the leaving subset  $U_{hl}$  requires the solution of a modified version of problem  $SP$ . The first occurs when  $i = o$ , since in this case there is no subset moving from the origin into node  $h$ . This implies that the capacity constraint (18) is always satisfied, and  $SP$  can be easily solved by sorting the elements of  $S_h$  by non-decreasing costs  $(c_{lj} - c_{hj})$ , and selecting the first in the order which have negative costs, up to a maximum number  $K$ . The second situation occurs when node  $l$  already belongs to the path from the origin to node  $h$ , implying that a subset-disjoint cycle has been detected. This cycle identifies a cyclic exchange along the sequence of facilities  $l = i_1, i_2, \dots, i_q = h$ . When selecting the last customer subset  $U_{i_q i_1}$  of the exchange, we must guarantee not only that its removal from  $S_h = S_{i_q}$  restores the capacity constraint for facility  $h$ , according to (18), but also that its insertion in  $S_l = S_{i_1}$  does not violate the capacity limit of facility  $l$ . Hence we must add to problem  $SP$  the additional constraint:

$$\sum_{j \in S_{i_q}} w_j x_j \leq r_{i_1} + W(U_{i_1 i_2}). \quad (20)$$

Problem  $SP$  has the structure of a knapsack problem in which an upper bound is imposed on the number of items that can be selected. Interesting approximation algorithms for knapsack problems with cardinality constraints have been proposed by Caprara et al. (2000). In our experimental investigation, however, we simply implemented a greedy algorithm.

## 5.4 Search for negative subset-disjoint cycles

The algorithm which explores the facility improvement graph is built along the general lines of a label-correcting approach. However, the peculiar dynamic nature of  $G^f(S)$  invalidates the shortest path optimality conditions on which label-correcting algorithms are based. Namely, we cannot state that if  $P$  is an optimal path from a node  $s$  to a node  $t$  in  $G^f(S)$ , and  $i$  is one of its intermediate nodes, then the subpath of  $P$  from  $s$  to  $i$  is an optimal path from  $s$  to  $i$ . Due to the lack of this property, the search for a shortest path or cycle in such a graph should maintain multiple labels for each node, each one corresponding to a different subpath through which the given node can be reached. Our heuristic approach aims at finding a negative subset-disjoint cycle, not necessarily the best. Hence, we chose to maintain, for each node  $i$ , only one cost label and, in order to decide when to update it, we check the Bellman conditions as in standard shortest path algorithms (Ahuja et al. 2001). Still, there are some cases in which we do not update a node label even though the Bellman conditions are violated. One such case occurs when arcs emanating from the current node have already been examined during the search, and a change in the subpath leading to it would cause a cascade of subsequent changes in the cost of arcs which have already been considered. This issue will be further clarified in the description of the algorithm given below.

The search algorithm maintains a *cost label*  $d(i)$  and a *predecessor*  $pred(i)$  for each node  $i$ , as in standard shortest path frameworks, plus a label  $U(i)$  to store the subset of customers which moves into  $i$  from its predecessor along the path, and a label  $min\_w(i)$  which indicates the minimum demand of the leaving customer subsets associated with the arcs  $(i, h)$  emanating from  $i$ . Formally,  $min\_w(i) = \min\{W(U_{ih}) \mid (i, h) \in FS(i)\}$ , where  $FS(i)$  denotes the forward star of node  $i$ . Labels  $d(i)$ ,  $pred(i)$  and  $min\_w(i)$  are initially set to  $\infty$ , while  $U(i)$  is the empty set.

The search starts at the origin  $o$ . At each iteration a node  $i$  is extracted from the set  $Q$  of candidate nodes still to be examined, and the following operations are executed until either  $Q$  is empty or we find a negative subset-disjoint cycle.

The path from the origin to  $i$  is checked for disjointness. If it is not disjoint, node  $i$  is skipped and another node is extracted from  $Q$ . Otherwise, all the arcs emanating from  $i$  are generated and examined. For each arc  $(i, h) \in FS(i)$ , two cases are possible.

1. *The path from  $o$  to  $h$  is subset-disjoint.* Then,

- (a) the subset of customers  $U_{ih}$  is found by solving problem (16)-(19);
- (b) the cost  $\beta_{ih}$  of arc  $(i, h)$  is computed, using the subset  $U_{ih}$  determined at step (1a);
- (c) if the Bellman condition for  $(i, h)$  is satisfied, i.e.,  $d(h) \leq d(i) + \beta_{ih}$ , the arc is skipped. Otherwise, one of the following three cases can occur:

- i.  *$h$  is a regular node and its forward star has not been examined yet.* In this case,  $h$  is inserted in  $Q$  and its labels are updated, i.e.,  $pred(h) := i$ ,  $d(h) := d(i) + \beta_{ih}$ ,  $U(h) := U_{ih}$ . Also, if  $W(U_{ih}) < min\_w(i)$ , then  $min\_w(i) := W(U_{ih})$ .
- ii.  *$h$  is a regular node and its forward star has already been examined.* This implies that the customer subsets  $U_{hl}$  associated with the arcs  $(h, l)$  emanating from  $h$  have already been determined. If the labels of  $h$  are changed, those subsets might no longer be candidate leaving subsets with respect to  $S_h$  and the new entering subset  $U_{ih}$ . This would lead to the need for recomputing some subsets  $U_{hl}$ , which in turn might cause some other capacity constraints to be violated. To avoid this problem, the labels of  $h$  are updated as in the previous step only if all the subsets  $U_{hl}$  are still candidate leaving subsets after the update, i.e., if  $W(U_{ih}) \leq r_h + min\_w(h)$ .
- iii.  *$h$  is a pseudonode.* By adding arc  $(h, o)$  to the path up to  $h$ , a subset-disjoint cycle is obtained which includes a pseudonode and the origin. If the cost of the cycle is negative, a profitable path exchange has been detected and the algorithm terminates.

2. *The path from  $o$  to  $h$  is not subset-disjoint.* There are two possible cases.

- (a)  *$h$  is a regular node.* Then a cycle has been found. We execute steps (1a)-(1b) taking into account that the computation of  $U_{ih}$  at step (1a) requires the solution of problem  $SP$  with the additional constraint (20). If  $d(i) + \beta_{ih} - d(h) < 0$ , a profitable cyclic exchange has been found and the algorithm terminates.
- (b)  *$h$  is a pseudonode.* This means that  $h = p_l$  for some facility location  $l$ , and node  $l$  has already been encountered along the path from  $o$  to  $p_l$ . In this case, arc  $(i, h)$  is disregarded since it would produce the same cycle which is generated when considering the arc from  $i$  to the regular node  $l$ .

## 6 The Facility Neighborhood Structure

The facility neighborhood is defined by three types of moves aiming, respectively, at opening a new facility, closing an existing facility, and transferring a facility to a different location. A facility move is legal only if it maintains the feasibility and is profitable if it results in a decrease of the total cost. The evaluation of a move profitability requires reassigning customers to the new set of open facilities, which is itself NP-hard. In order to limit the computational effort, we only considered a restricted facility neighborhood, induced by estimates of the cost savings.

### 6.1 The opening moves

The *opening move* attempts to establish a new facility at an unused location. The restricted neighborhood search scheme for this move can be described as follows.

For each unused location  $i$ , we identify the set  $U_i = \{j \in J | c_{ij} - c_{F(j)j} < 0\}$ . Let  $z_i = \sum_{j \in U_i} (c_{ij} - c_{F(j)j}) + f_i$ . We then sort the unused facility locations by non-decreasing order of the saving estimate  $z_i$ . Locations with positive saving estimates are excluded from further consideration. The remaining locations are examined in turn as indicated by the ordering, and the profit of the corresponding neighbors is evaluated, until a profitable move is detected.

The saving associated with the opening of a facility at location  $i \in I \setminus I(S)$  is evaluated by first considering a partial customer reassignment, which only involves the customers in  $U_i$ . In particular,

1. if  $W(U_i) \leq s_i$ ,  $z_i$  is the effective cost saving for opening a facility at site  $i$  and allocating all the customers in  $U_i$  to it. A feasible and profitable move has then been detected;
2. if  $W(U_i) > s_i$ , we try to find a subset of  $U_i$  so that its assignment to  $i$  maintains feasibility while decreasing the overall cost of the solution. This problem can be formulated as a knapsack problem (KP) in the following way. Let  $x_j$  be a binary variable which takes value 1 if customer  $j$  is selected, 0 otherwise.

$$\min \quad \sum_{j \in U_i} (c_{ij} - c_{F(j)j})x_j + f_i \quad (21)$$

$$\text{s.t.} \quad \sum_{j \in U_i} w_j x_j \leq s_i \quad (22)$$

$$x_j \in \{0, 1\} \quad \forall j \in U_i \quad (23)$$

An approximation to the optimal solution of KP can be easily obtained by first redirecting to  $i$  the customers in  $U_i$  which produce the largest saving per unit demand volume. Let

$U_i^* = \{j \in U_i | x_j = 1\}$  be such an approximate solution, and  $z_i^*$  its objective function value. If  $z_i^* < 0$ , the opening of a facility at  $i$  and the allocation of the customers in  $U_i^*$  to it results in a feasible and profitable move.

If the partial reassignment operation fails to detect an improving solution, we try a complete reassignment of the customers to the new set of open facilities  $\bar{I} = I(S) \cup \{i\}$  by solving the following Generalized Assignment Problem (GAP). Let  $x_{ij}$  be a binary variable which takes value 1 if customer  $j$  is assigned to facility  $i \in \bar{I}$ , 0 otherwise.

$$\min \quad \sum_{i \in \bar{I}} \sum_{j \in J} c_{ij} x_{ij} \quad (24)$$

$$\text{s.t.} \quad \sum_{i \in \bar{I}} x_{ij} = 1 \quad \forall j \in J \quad (25)$$

$$\sum_{j \in J} w_j x_{ij} \leq s_i \quad \forall i \in \bar{I} \quad (26)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in \bar{I}, j \in J \quad (27)$$

In our local search algorithm, an approximate solution to GAP is obtained by the greedy algorithm proposed by Romeijn and Romero Morales (2000). The approach is based on the definition of a weight function which is used to measure the desirability of assigning each customer to each facility. The greedy algorithm then schedules the customers according to a decreasing order of desirability. Let  $S'$  be the solution so obtained, and  $C(S')$  be its cost. The set of constraints (26) guarantees that  $S'$  is a feasible solution. Hence, if  $C(S') < C(S)$ , a feasible profitable move has been detected.

## 6.2 The closing moves

The *closing move* attempts to close an existing facility and reallocates its customers among the remaining open facilities. As for the opening move, we only explore a restricted neighborhood. We associate with each location  $i$  in use, a saving estimate  $z_i = -f_i + \sum_{j \in S_i} (c_{\bar{i}j} - c_{ij})$ , where  $\bar{i}$  denotes the closest facility to customer  $j$  among the facilities in  $\bar{I} = I(S) \setminus \{i\}$ . Used facility locations with positive  $z_i$  or such that  $W(S_i) > \sum_{h \in I(S) \setminus \{i\}} r(S_h)$  are not considered. The moves associated with the remaining locations are evaluated in non-decreasing order of the saving estimates  $z_i$  until a profitable move is detected. When evaluating the cost for closing a facility at  $i$ , a first attempt at detecting a profitable move is made by reassigning only the customers currently assigned to  $i$ . This requires solving a Reduced Generalized Assignment Problem (RGAP) where the capacities  $s_i$  are replaced by the residual capacities  $r_i$ , for each  $i \in \bar{I}$ , and involving only the customers in  $S_i$ . RGAP

is solved through the same greedy algorithm used for GAP (Romeijn and Romero Morales 2000). If RGAP does not yield an improving solution, the reassignment of the whole set of customers to the new set of open facilities  $\bar{I}$  is tried by solving GAP as for the opening move.

### 6.3 The transferring moves

In some cases, closing and opening moves might not generate cost-decreasing solutions if considered separately. However, improving solutions might derive from their combined utilization, i.e., from *transferring moves*. A transferring move displaces a facility from its current location and reopens it at a different site.

A first attempt is simply made by moving an existing facility to a different location, and re-assigning the whole cluster of customers associated with the moving facility to the new location. Checking for feasibility and profitability of this kind of move is pretty inexpensive and hence the move can be attempted for each pair  $(i, h)$  of used-unused sites.

A more complex transferring move is also considered which completely redistributes customers to the new set of open facilities. The main issue concerning this move is how to conveniently select promising pairs of facility locations for which to attempt the move. The restricted neighborhood search in this case operates as follows: we first select the facility to be closed on the basis of the estimates of the customer redistribution costs, as described for the closing move in Section 6.2. Once the closing facility  $i$  has been chosen, we choose for the facility opening the location  $h$  which could capture the greatest amount of customers currently assigned to  $i$ , provided that  $\sum_{i \in \bar{I}} s_i \geq \sum_{j \in J} w_j$ , where  $\bar{I} = I(S) \setminus \{i\} \cup \{h\}$ . A customer  $j \in S_i$  is considered captured by site  $h$  if  $c_{hj} < c_{ij}$ . Once again the problem of allocating customers to the set of open facilities  $\bar{I}$  requires solving GAP.

## 7 The Neighborhood Search Algorithm

The facility and customer neighborhood structures are made operational within a neighborhood search algorithm. The general scheme is very simple and, in this context, we did not make any specific effort to extend and refine it through the use of more sophisticated guidance processes, such as tabu search. The basic algorithm simply proceeds from an initial feasible solution by a sequence of local changes which are obtained through facility or customer moves, and the process is iterated until a local optimum solution is found.

This general framework leaves large scope of operations, thanks to the amplitude of possible ways of alternating facility and customer moves. A reasonable choice, whose efficiency has been

corroborated by the computational results, works as follows. The simple transferring move, which is quite inexpensive and moves the facilities rather quickly to better locations, is executed first. The customer assignment improvement is then committed to the search for profitable multi-customer exchanges on the facility improvement graph. When no more profitable multi-customer exchanges can be detected, the algorithm tries the facility moves in the following order: opening, closing, and transferring. As soon as a facility move produces a cost-decreasing solution, the search for profitable multi-customer exchanges is restarted to improve the assignment of customers to the new set of open facilities. Only when all the facility moves fail, the finer single-customer exchange is executed to perfect the solution.

In order to circumscribe the tendency of the local improvement algorithm to get trapped at local optima, we have implemented a *multistart* mechanism by obtaining multiple initial feasible solutions through a Lagrangean relaxation/subgradient optimization procedure. Such a procedure is analogous to the one described in (Holmberg et al. 1999), Section 2. Namely, we apply Lagrangean relaxation to the assignment constraints (2), and obtain a problem which separates into  $n$  knapsack problems, one for each facility  $i$ . The solutions of the knapsack problems provide reduced costs associated with each facility location, values for the variables  $y_i$  and  $x_{ij}$ , and a lower bound to the problem. The dual problem is solved with subgradient optimization, and primal feasible solutions are obtained by appropriately reassigning customers which are either not assigned or assigned to multiple facilities. The reader is referred to (Holmberg et al. 1999) for further details.

In our computational investigation, we used the Lagrangean heuristic to generate 500 feasible solutions, and select 30 of them as restart solutions. The selected solutions are the ones which have the lowest cost and also guarantee diversity with respect to the set of open facilities.

## 7.1 Some implementation issues

There are some issues related to the actual implementation of the algorithmic paradigm which can substantially impact the effectiveness of the overall methodology. Some of them are related to the algorithm which explores the facility improvement graph, as described in Section 5.4. First, the rule used for choosing the next node to be extracted from  $Q$  typically conditions the kind of cycles or paths that are explored. In our experimental investigation, we considered three different implementations of  $Q$ , corresponding to a *breadth-first search (bfs)*, a *depth-first search (dfs)*, and a *best search (bs)* of the graph. Second, the cycle detection heuristic is applied by selecting a regular node as the origin from which to start the search. However, many profitable multi-customer

exchanges might be missed by considering only one node as the origin node. We thus apply the cycle search algorithm multiple times with different nodes as origin.

The last remark concerns some parameter options of the Lagrangean heuristic that generates the initial solutions. In accordance with the parameter setting used in (Holmberg et al. 1999), we initially set to 1 the parameter  $\lambda$  used in the computation of the step size of the subgradient method, and halved its value after  $p = 5$  consecutive iterations with no improvement of the lower bound. However, an in-depth analysis of the initial solutions thus obtained showed that this parameter combination produces solutions that are structurally quite homogeneous, thus thwarting the benefits of the restart mechanism, especially for medium sized problems. Subsequent fine-tuning of the Lagrangean heuristic parameters showed that by halving  $\lambda$  less frequently, greater diversity of the initial solutions in terms of the set of open facilities could be achieved. Hence, we report in the following the results obtained by fixing the parameter  $p$  to a bigger value, namely  $p = 10$ . The results for the initial value  $p = 5$  are also available at <http://www.di.unipi.it/~scaparra/research.html>.

## 8 Computational study

The proposed multi-exchange heuristics have been tested on two varieties of benchmark instances available in the literature, and on a real-life instance provided by an Italian cookie factory. Our analysis of the results will mainly concentrate on the most difficult problems within the aforementioned benchmark sets, since the main goal of our approach is to provide a solution tool for large-scale problems, which cannot be solved by commercial optimization software packages. Results for some of the small instances are briefly reported for completeness. Whenever it is possible, we validate the quality of our solutions through direct comparison with the optimal solutions found through the commercial code CPLEX (version 7.0). Both the CPLEX application and the VLSN heuristics were coded in C++, compiled with the GNU g++ compiler and run on a PC with a Athlon/1200Mhz processor and 512 Mb RAM, under the RedHat Linux 7.1 operating system.

The first group of benchmark instances comprises four sets of problems, which were randomly generated by Holmberg et al. (1999). The Lagrangean heuristic with repeated matching proposed in Holmberg et al. (1999), and denoted in the following as LH, is one of the most efficient heuristics for solving SSCFLP. In particular, in a previous work (Holt et al. 1999), the authors proved its superiority over Pirkul's heuristic. LH was therefore chosen as an additional term of comparison for evaluating the performance of our VLSN algorithms on this problem group.

The results for these problem sets are summarized in Tables 1 and 2. For the sake of brevity, we report the complete results only for set 4, which contains the most difficult instances (Table 2), whereas for the first three sets we only show the average statistics, that is the average deviation from the optimal solutions

obtained by CPLEX, and the average time for the whole set (Table 1). The complete tables for these sets are available at the website. Table 2 shows for each problem of the fourth set, the objective function values obtained by CPLEX, by LH and by the three versions of our VLSN algorithm, followed by the % deviations of the objective function values of the four heuristics from the best-known solutions. The last three columns give the CPU-times in seconds required by the VLSN heuristics to perform the 30 restarts.

Table 1: Average Statistics for Holmberg’s problems

Pr. Set	$n$	$m$	prob. tot	solved prob.		% Deviation				Time (sec.)		
				<i>LH</i>	<i>VLSN</i>	<i>LH</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>
p1-p24	10-20	50	24	23	24	0.001	0.000	0.000	0.000	0.54	0.54	0.53
p25-p40	30	150	16	12	10	0.027	0.135	0.132	0.073	12.78	12.66	12.53
p41-p55	10-30	70-100	15	11	10	0.065	0.024	0.028	0.011	1.61	1.62	1.61
p56-p71	30	200	16	8	9	0.189	0.021	0.019	0.027	16.33	15.97	15.85
p1-p71	10-30	50-200	71	54	53	0.063	0.049	0.039	0.028	7.07	6.97	6.91

Table 2: Computational results for problems p56-p71.

Pr. No.	Opt. Sol.	Objective Value				% Deviation				Time (sec.)		
		<i>LH</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>	<i>LH</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>
p56	21103	21163	21118	21120	21118	0.284	0.071	0.081	0.071	15.94	15.77	15.94
p57	26039	26167	26085	26075	26085	0.492	0.177	0.138	0.177	27.47	26.32	27.09
p58	37239	37792	37239	37240	37284	1.485	0.000	0.003	0.121	48.22	46.60	45.97
p59	27282	27300	27282	27282	27282	0.066	0.000	0.000	0.000	28.71	29.97	29.90
p60	20534	20534	20534	20534	20534	0.000	0.000	0.000	0.000	3.20	3.27	3.22
p61	24454	24454	24454	24454	24454	0.000	0.000	0.000	0.000	6.20	6.27	6.27
p62	32643	32643	32651	32648	32651	0.000	0.025	0.015	0.025	28.92	28.68	27.15
p63	25105	25105	25108	25108	25108	0.000	0.012	0.012	0.012	13.81	15.22	14.78
p64	20530	20530	20530	20530	20530	0.000	0.000	0.000	0.000	0.18	0.18	0.18
p65	24445	24445	24445	24445	24445	0.000	0.000	0.000	0.000	0.18	0.18	0.18
p66	31415	31504	31420	31429	31420	0.283	0.016	0.045	0.016	8.40	8.14	8.16
p67	24848	24876	24849	24849	24849	0.113	0.004	0.004	0.004	13.83	10.34	10.15
p68	20538	20538	20538	20538	20538	0.000	0.000	0.000	0.000	3.33	3.32	3.32
p69	24532	24532	24532	24532	24532	0.000	0.000	0.000	0.000	6.16	6.15	6.50
p70	32321	32393	32332	32323	32323	0.223	0.034	0.006	0.006	27.60	27.04	25.95
p71	25540	25562	25540	25540	25540	0.086	0.000	0.000	0.000	29.12	28.10	28.78

Before proceeding to the analysis of the results, a discussion of how the parameter  $K$  (the maximum number of customers that can be transferred simultaneously in the multi-exchange phase) was chosen for the experimentations is in order. Some preliminary tests, performed to study the impact of this parameter, showed that quite often the best results are obtained when  $K$  ranges from 4% to 9% of the total number of customers. An example of this type of trend is given in Figure 4 for the fourth set of problems. The graph shows how the average percentage deviation of the heuristic objective function values from the best-known solutions varies as  $K$  increases. This is best explained as follows. Up to a certain threshold, there

are benefits in moving additional customers. Beyond this, however, either the objective function value levels off or deteriorates, since the movement of a large number of customers tends to destroy the optimal customer assignments already identified by previous moves. In any case, the heuristics show a robust behavior independent of the value of  $K$ . For more detailed information about the impact of parameter  $K$ , the interested reader is referred to the website.

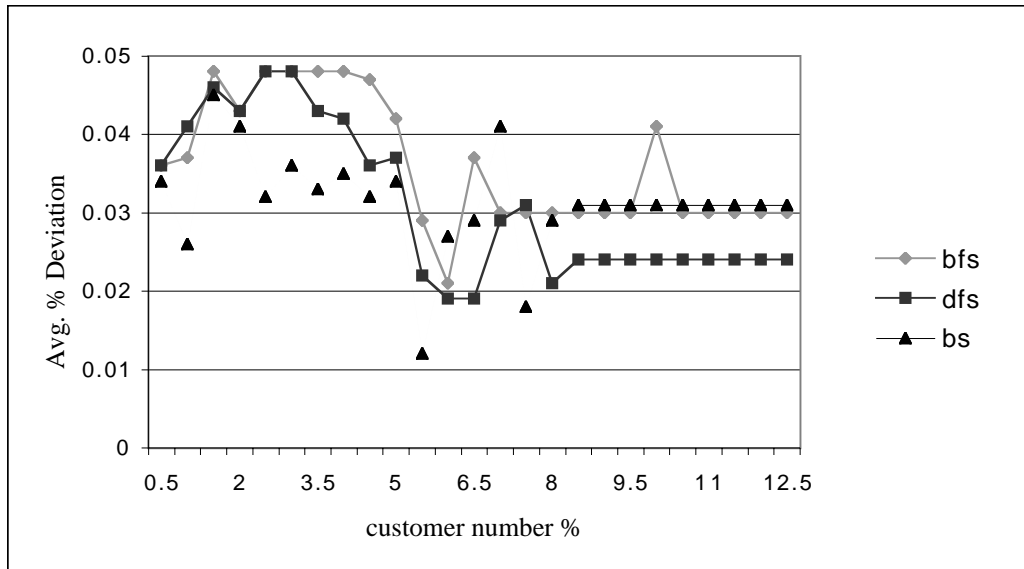


Figure 4: Average percentage deviation as a function of  $K$  for problem set 4.

Based on these observations, the results reported in Tables 1 and 2 are the ones produced by the best  $K$  within the identified range for each set. Namely,  $K$  is fixed to 3 for set 1, 6 for set 2, 7 for set 3, and 12 for set 4.

Table 1 shows that every VLSN implementation solves all the instances of the first set to optimality. On the other hand, the VLSN heuristics seem to perform worse than LH, in terms of average percentage deviation, on the second set of problems. This is mainly due to the subset of problems p29-p32, which seem to be characterized by an intrinsic difficulty; CPLEX, in fact, solved them in a significant amount of time (ranging from 130 to 620 seconds). The VLSN heuristics outperform LH on the third and fourth sets. In particular, the *bfs* version solves to optimality at the first restart the most difficult problem (p58), for which neither LH nor the exact method in (Holmberg et al. 1999) could find the optimal solution. Moreover, the VLSN heuristics, when implemented with the Lagrangean parameter  $p$  equal to 5, produce even better solutions on problem set 4 (see the website). Indeed, every VLSN heuristic with  $p = 5$  yields an average percentage deviation of only 0.012%, as opposed to the 0.189% average percentage deviation of LH. Furthermore, the *dfs* implementation solves to optimality also the difficult problem p57.

As far as the CPU-time is concerned, direct comparisons between LH and the VLSN heuristics are not possible since the two codes were run on two different machines. However, the running time does not seem to be the critical issue for our approach. Problem p58, for example, required less than one minute to be

solved to optimality by the *bfs* version of the VLSN algorithm, and the other implementations found very good approximations to the optimal solution in about the same amount of time (see Table 2). A well fine-tuned CPLEX application needed around two and a half hours to solve the same problem. In the average, the instances of the most difficult set were solved in less than 20 seconds by different implementations of the VLSN technique. Further, the running times reported in the tables refer to the 30 restarts, but, as shown in Table 3, much less restarts than the maximum allowed were needed for most of the instances. In many cases, especially for the instances of the first and second problem set, the best solutions were found in the first restart.

Table 3: Number of problems solved in each restart interval.

No. of Restarts	p1-p24			p25-p40			p41-p55			p56-p71			All		
	<i>bfs</i>	<i>dfs</i>	<i>bs</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>	<i>bfs</i>	<i>dfs</i>	<i>bs</i>
-	1	1	1	1	1	1	1	1	1	2	2	2	5	5	5
[1,5]	19	20	20	12	11	10	9	11	9	11	10	11	51	52	50
(5,10]	1	2	0	1	2	1	1	1	0	0	1	0	3	6	1
(10,20]	3	1	3	2	1	3	2	0	2	3	1	2	10	3	10
(20,30]	0	0	0	0	1	1	2	2	3	0	2	1	2	5	5

The second group of benchmark instances was taken from the Beasley OR-Library (1990). These instances were originally generated for the capacitated facility location problem without single-source constraints and some of them are not feasible when this additional requirement is imposed. We restricted our experimentation only to the feasible problems. Among them, the 24 medium-sized instances were solved in negligible computational time by both CPLEX and by our VLSN heuristics. We therefore only provide the results for the 12 large scale problems, namely those with 100 candidate locations and 1,000 customers. Beasley’s problems were previously used by Hindi and Pieńkosz (1999) as test cases for SSCFLP. Hence, we compare the performance of our approach to that of the Lagrangean heuristic proposed by those authors, denoted in the following by HP. Since it was not possible to compute optimal solutions for these very large problems using CPLEX, we used the optimal solutions to the problems without single-source constraint as lower bounds for estimating the optimality gap. In Table 4, we report the results obtained by HP and by the *bfs* implementation of our heuristic. For this large problem set, we only performed 5 restarts. The value of  $K$  was fixed to 60, that is 6% of the number of customers, in accordance with the choice made for the fourth set of Holmberg’s problems. The results show that our heuristic outperforms HP for nearly all problems with an average deviation of only 0.079% as opposed to 0.766% of HP. The other VLSN implementations perform slightly worse than *bfs* but still better than HP (see the website for additional details).

A further test has been performed using real data from an Italian cookie factory. In the supply chain of this factory, capacitated transit points have to be located at a subset of 23 prespecified sites in order to

Table 4: Computational results for problems capa1-capc4.

Pr. No.	Lower Bound	Objective Value		% Deviation		Time (sec.)
		<i>HP</i>	<i>bfs</i>	<i>HP</i>	<i>bfs</i>	<i>bfs</i>
capa1	19240822.45	19247166.00	19242536.31	0.033	0.009	214.29
capa2	18438046.54	18594257.87	18439832.00	0.847	0.010	108.08
capa3	17765201.95	17907533.52	17765201.95	0.801	0.000	60.10
capa4	17160439.01	17160612.23	17160809.04	0.001	0.002	12.19
capb1	13656379.58	13657994.99	13658413.01	0.012	0.015	145.54
capb2	13361927.45	13720471.92	13365455.33	2.683	0.026	85.51
capb3	13198556.43	13374673.25	13239067.70	1.334	0.307	50.05
capb4	13082516.50	13097933.52	13085407.10	0.118	0.022	19.02
capc1	11646596.97	11710752.16	11685611.34	0.551	0.335	116.85
capc2	11570340.29	11827419.33	11570437.68	2.222	0.001	38.93
capc3	11518743.74	11582710.22	11540012.26	0.555	0.185	28.60
capc4	11505767.39	11509395.65	11509395.65	0.032	0.032	16.13
Avg.				0.766	0.079	74.64

Problems size:  $n = 100$ ,  $m = 1000$ 

serve 104 clients at minimum cost. Additionally, each customer must be served by a single transit point. The instance has been solved both by CPLEX and by our VLSN algorithms. CPLEX was terminated after the generation of 300,000 nodes or a 2-hour run time. The best approximated solution thus found had a 0.07% duality gap and a daily cost of 6,570,877 Italian Liras (3,393.57 Euro). By comparison, our heuristics computed very good approximations in less than 50 seconds and showed a robust behavior independently of the value  $K$ . Specifically, for  $K = 6$  (i.e., 6%), the *bfs* version found a solution with a cost having a percent deviation of 0.48 from the cost of the CPLEX solution. The *dfs* version found a solution having a percent deviation of 0.66. Expanded results are available at the website.

## 9 Conclusions

We proposed some VLSN algorithms for the capacitated facility location problem with single-source constraints. Computational results for some benchmark instances and for a real case demonstrated the effectiveness of the approach for solving large-scale problems. In particular, the definition of multi-customer multi-exchanges, detected on a dynamic improvement graph, showed to be very effective in solving this challenging problem.

## Acknowledgments

The research of the first author was supported in part by the National Science Foundation Grants DMI-9900087 and DMI-0085682; the research of the second author was supported in part by the National Science

Foundation Grants DMI-9820998 and DMI-0217123; the research of the last three authors by MIUR, grant CTB 02.00500.ST97, and Ricerche Nazionali (40%), Cofinanziato Ateneo 2001.

## References

- Agar, M., S. Salhi. 1998. Lagrangean heuristics applied to a variety of large capacitated plant location problems. *Journal of the Operational Research Society* **49** 1072–1084.
- Ahuja, R.K., Ö. Ergun, J.B. Orlin, A.P. Punnen. 2002. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics* **123** 75–102.
- Ahuja, R.K., J. B. Orlin, D. Sharma. 2001. Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem. *Mathematical Programming* **91** 71–97.
- Barcelo J., J. Casanova. 1984. A heuristic Lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research* **15** 212–226.
- Beasley, J.E. 1993. Lagrangian heuristics for location problems. *European Journal of Operational Research* **65** 383–399
- Beasley, J.E. 1990. OR-Library: distributing test problems by electronic mail. *Journal of Operational Research Society* **41** 1069–1072 (also <http://www.ms.ic.ac.uk/info.html>).
- Caprara, A., H. Kellerer, U. Pfersich, D. Pisinger. 2000. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research* **123** 333–345.
- Erlenkotter, D. 1978. A dual-based procedure for uncapacitated facility location. *Operations Research* **26** 992–1009.
- Frangioni, A., E. Necciari, M. G. Scutellà. 2000. A multi-exchange neighborhood for minimum makespan machine scheduling problems. Tr 00-17, University of Pisa. To appear in *Journal of Combinatorial Optimization*.
- Hindi, H., K. Pieńkosz. 1999. Efficient solution of large scale, single-source, capacitated plant location problems. *Journal of the Operational Research Society* **50** 268–274.
- Holmberg, K., M. Ronnqvist, D. Yuan. 1999. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research* **113** 544–559.
- Holt, J., M. Ronnqvist, S. Tragantalerngsak. 1999. A repeated matching heuristic for the single-source capacitated facility location problem. *European Journal of Operational Research* **116** 51–68.
- Klincewicz, J., H. Luss. 1986. A Lagrangean relaxation heuristic for capacitated facility location with single-source constraints. *Journal of the Operational Research Society* **37** 495–500.

- Neebe, A., M. Rao. 1983. An algorithm for the fixed-charge assigning users to sources problem. *Journal of the Operational Research Society* **34** 1107–1113.
- Pirkul, H. 1987. Efficient algorithm for the capacitated concentrator location problem. *Computers & Operations Research* **14** 197–208.
- Romeijn, H., D. Romero Morales. 2000. A class of greedy algorithms for the generalized assignment problem. *Discrete Applied Mathematics* **103** 209–235.
- Sridharan, R. 1991. A lagrangian heuristic for the capacitated plant location problem with single source constraints. *European Journal of Operational Research* **66** 305–312.
- Thompson, P., J. B. Orlin. 1989. Theory of cyclic transfers. Unpublished manuscript. Operations Research Center, MIT.
- Thompson, P., H. Psaraftis. 1993. Cyclic transfer algorithms for multi-vehicle routing and scheduling problems. *Operations Research* **41** 935–946.