Neighborhood based heuristics for a Two-level Hierarchical Location Problem with modular node capacities

Bernardetta Addis[*,⋆] Giuliana Carello[*] Alberto Ceselli [◇]

[*] *Dipartimento di Elettronica e Informazione, Politecnico di Milano*
Via Ponzio 34, 20133 Milano, Italia

[⋆] *Dipartimento di Informatica, Università degli Studi di Torino*
C.So Svizzera 185, 10149 Torino, Italia

[◇] *Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano*
via Bramante 65, 26013 Crema, Italia

Abstract

In many telecommunication network architectures a given set of client nodes must be served by different kinds of facility, which provide different services and have different capabilities. Such facilities must be located and dimensioned in the design phase.

We tackle a particular location problem in which two sets of facilities, mid level and high level, have to be located. Different devices can be installed in each mid level facility, providing different capacities at different costs. The assignment of clients to facilities and of facilities to higher level entities must be optimized, as well. We propose a heuristic approach, based on very large scale neighborhood search, to tackle the problem, in which both ad-hoc algorithms and general purpose solvers are applied to explore the search space. We report on experimental results using datasets of instances from the literature. These experiments show that the approach is promising and that Integer Linear Programming based neighborhoods are significantly effective.

Keywords: *local search, variable neighborhood search, very large scale neighborhood search, integer programming, location, telecommunications*

1 Introduction

Many telecommunication networks have a hierarchical structure, in which different sets of nodes play different roles. In such networks, nodes representing clients must be served by nodes representing facilities. As different kinds of tasks are required, different kinds of facilities are needed. Usually the set of clients is given, while facilities of different kinds must be located and dimensioned. Examples of such network structure can be found in IP networks, in which access nodes must be connected to edge nodes which, in turn, must be connected to the core backbone nodes, or in fiber-to-the-home networks, where clients must be connected to cabinet nodes which collect traffic and send it to central offices.

The optimal design of networks with the above structure can be seen as a Facility Location Problem in which two different sets of facilities are considered, mid level and high level facilities. In the star-star topology of the network each client must be assigned to exactly one mid level facility. Besides, each mid level facility must be assigned to exactly one high level facility. For each client a demand amount to be served is given and, for each facility, a capacity is given, which limits the amount of demands of clients assigned to it. Moreover, each mid level facility must be dimensioned, by installing different kinds of devices, capable of serving different amounts of demand at different costs.

A recent review on hierarchical facility location problems, covering papers since the mid-80s, can be found in [9]. Hierarchical facility location problems are classified according to features such as flow pattern and service availability; applications, models and approaches are described. In a classical generalization of the facility location problem, the so-called Multi-level Facility Location Problem, a set of clients is given together with k sets of facilities, where each set represents a different facility level. Each client must be assigned to a path of k facilities, and its demand must be routed through a facility of each level following a hierarchical order. For that problem, heuristic and exact approaches [10] as well as approximation properties [2] have been investigated. Another similar generalization of the facility location problem is the Two-level Simple Plant Location Problem proposed in [6]: each client must be assigned to one and only one facility of the mid level which, in turns, must be assigned to one and only one facility of the high level; facilities of both levels are uncapacitated.

The problem we considered shares some features also with the Two-echelon Single Source Capacitated Facility Location Problem described in [11], where each client must be assigned to exactly one mid level facility which, in turns, must be assigned exactly to one depot; indeed, depots act as high level facilities, which however are not capacitated. The most similar problem to the one proposed in this paper is tackled in [8]: the location of two different types of facilities, concentrators and routers, in a telecommunication network is considered; both concentrators and routers are capacitated. Each terminal in the network, which represents a client, has to be assigned to exactly one concentrator, which must in turn be assigned to one router. The location of both concentrators and routers has to be chosen. The problem is heuristically tackled by computing both lower and upper bounds. Modeling of telecommunications applications as Two-Level Facility Location Problems are also proposed in [5] and [12]: in [5] the IP network design problem is heuristically tackled; in [12] a hierarchical continuous location problem, where a set of concentrators and one central equipment must be located, is tackled with a column generation approach.

The problem of simultaneously locating and dimensioning capacitated facilities in a star-star network is considered and exactly tackled in [1]. Such a problem is denoted as the *Two-level Hierarchical Capacitated Facility Location Problem* (TLHCFLP). The TLHCFLP is *NP*-hard, as it generalizes the classical Facility Location Problem. An exact optimization algorithm is proposed which exploits a hybrid formulation and dynamic column generation within a branch-and-bound framework. This algorithm can solve instances with up to 200 clients and 50 candidate location sites in less than two hours.

However, as real life applications such as fiber-to-the-home network design, may require to solve problems with up to thousands of clients, fast and efficient heuristics are needed. In this paper we propose a heuristic approach to the TLHCFLP which combines Integer Linear Programming models, local search, variable neighborhood search [15] and very large scale neighborhood search [16], using both ad-hoc algorithms and general purpose solvers to explore the search space.

In Section 2 we discuss an ILP model for TLHCFLP, in Section 3 we describe our algorithms and in Section 4 we report some computational results. Some brief conclusion is drawn is Section 5.

2 Problem description and formulation

In the TLHCFLP a set of client nodes $\mathcal{I}$ is given. Each client node $i \in \mathcal{I}$ has a demand a_i and it must be connected to a mid level facility, which in turn must be connected to a high level facility. Both kinds of facility must be located: candidate site for mid level and high level facilities are given and represented by sets $\mathcal{J}$ and $\mathcal{K}$, respectively. Placing a mid level facility in a site $j \in \mathcal{J}$ and a high level facility in a site $k \in \mathcal{K}$ implies installation costs c_j and g_k, respectively. Assigning client i to a mid level facility located in $j \in \mathcal{J}$ and a mid level facility located in j to a high level facility located in $k \in \mathcal{K}$ implies connection costs d_{ij} and l_{jk}, respectively.

Each mid level facility must be dimensioned by equipping it with a device chosen in a set $\mathcal{T} = \{1 \dots T\}$. For each device $t \in \mathcal{T}$ the capacity b_t and the setup cost f_t are given; the b_t coefficient represents also the demand to be served by a high level facility to which the mid level facility equipped with device t is assigned. To keep the problem close to real life applications we suppose that each device t provides half

capacity with respect to device $t+1$; moreover, according to economy of scale, device costs are assumed to be sub-linear with respect to the provided capacity. All high level facilities provide the same capacity B.

To model the TLHCFLP problem four kind of variables are needed to represent the decisions which must be taken: whether to open or not a high level facility in each $k \in \mathcal{K}$ (binary variables z_k), whether to open or not a mid level facility equipped with device $t \in \mathcal{T}$ in each candidate site $j \in \mathcal{J}$ (binary variables y_{jt}), whether to assign or not a mid level facility opened in $j \in \mathcal{J}$ and equipped with device $t \in \mathcal{T}$ to a high level facility in $k \in \mathcal{K}$ (binary variables w_{jtk}), whether to assign or not a client $i \in \mathcal{I}$ to a mid level facility located in $j \in \mathcal{J}$ (binary variables x_{ij}).

The TLHCFLP can be modelled as follows:

$$\min \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} d_{ij}x_{ij} + \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}(c_j + f_t)y_{jt} + \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}} l_{jk}w_{jtk} + \sum_{k\in\mathcal{K}} g_k z_k \tag{1}$$

$$\sum_{j\in\mathcal{J}} x_{ij} \geq 1, \forall\ i \in \mathcal{I} \tag{2}$$

$$\sum_{i\in\mathcal{I}} a_i x_{ij} \leq \sum_{t\in\mathcal{T}} b_t y_{jt}, \forall\ j \in \mathcal{J} \tag{3}$$

$$\sum_{t\in\mathcal{T}} y_{jt} \leq 1, \quad \forall j \in \mathcal{J} \tag{4}$$

$$\sum_{k\in\mathcal{K}} w_{jtk} \geq y_{jt}, \forall\ j \in \mathcal{J}, \forall\ t \in \mathcal{T} \tag{5}$$

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} b_t w_{jtk} \leq B z_k, \forall\ k \in \mathcal{K} \tag{6}$$

$$x_{ij}, w_{jtk}, y_{jt}, z_k \in \{0,1\} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \tag{7}$$

The objective function (1) aims at minimizing the sum of installation, setup and assignment cost. Constraints (2) force each client to be assigned to at least one mid level facility, while constraints (5) force each open mid level facility to be assigned to a high level one. Inequalities (3) guarantee that each mid level facility capacity, which is provided by equipping it with a suitable device, is sufficient to serve the demand of all the assigned clients, while inequalities (4) guarantee that each mid level facility is equipped with at most one device. Finally, inequalities (6) guarantee that each high level facility has enough capacity to serve the demand of all the assigned mid level facilities. Integrality conditions (7) complete the model.

The model has a polynomial number of variables and constraints, and is therefore suitable to be tackled by general purpose ILP solvers. As discussed in detail in [1], this approach does not allow to solve to proven optimality a large number of instances, whose features are similar to that of practical applications. Ad-hoc exact algorithms perform substantially better, but still allow to solve only instances whose size is far from that of practical applications. Therefore, in the next section we propose heuristic algorithms which aim to be as fast as possible, still producing near-optimal solutions.

3 Algorithms

Our heuristic algorithms are based on two main phases: a Descent Phase, which provides intensification performing a variable neighborhood search, and a Kick Phase, which provides diversification in an iterated local search fashion. The Descent Phase exploits ad-hoc algorithms to explore both classical neighborhoods and very large scale ones, while the Kick Phase explores very large scale neighborhoods using ILP techniques. These two phases are performed in sequence and the whole sequence is repeated until no improving solution is found or a maximum number of iterations is reached.

3.1 Descent Phase

We developed 5 different local search neighborhoods for the Descent Phase. Two are generated applying basic swap moves, while three of them are very large scale neighborhoods, which generalize the neighborhoods proposed by Ahuja et al. in [4] for a capacitated single source facility location problem.

Single Exchange Neighborhoods We developed two single exchange neighborhoods. The first one, single client exchange neighborhood (SCE), considers all the pairs of clients assigned to two different mid level facilities. Their assignments are swapped provided that the residual capacity on each of the considered mid level facilities is sufficient to receive the new client, once the one currently assigned has relinquished. The second one, single facility exchange neighborhood (SFE), considers all the pairs of open mid level facilities assigned to two different high level facilities and swaps their assignments, provided that the residual capacity on each of the high level facilities is sufficient to receive the new mid level facility, once the currently assigned one has relinquished.

Improvement graph based neighborhoods Three neighborhoods have been developed which are based on the improvement graph and on very large scale neighborhoods proposed in [4]. In such neighborhood a sequence of moves is considered instead than a single swap. The possible moves are represented by arcs of an improvement graph, the arc cost representing the increasing or decreasing in the objective function if the move is applied. The improving sequences of moves are represented by negative cost cycles. To guarantee the feasibility of the sequence of moves, at most one move involving each mid level or high level facility can be applied. As the minimum cost set disjoint cycle is a difficult problem, improving neighbors are heuristically found.

In the client cycle (CC) neighborhood, improvement graph nodes are associated to clients or to mid level facilities. An arc between two client nodes i and j, assigned to h_i and h_j in the current solution, respectively, represents the possibility of assigning client i to h_j, while j is relinquish. The arc exists if the move is feasible, and its cost is the difference between new cost ($d_{ih_j} + f_\tau$, where τ is the device needed by h_j if i is assigned and j has relinquished) and the current cost ($d_{j,h_j} + f_t$ where t is the current device cost). An arc between a client node i and a facility node k represents the possibility of assigning i to k. The arc exists if the assignment does not exceed the facility and device capacity, and it may exist also between a client and a close facility. In this case, it represents also the possibility of opening the facility: its cost takes into account both the client assignment and device cost and the assignment cost of mid level facility to a high level one. The considered high level facility may be open or not: thus the improvement graph represents the possibility of opening both mid level and high level facilities. A root node is added. An arc between the root node and a client one represents the possibility of moving the client without assigning another client to its current facility: its negative cost is equal to the current assignment cost of the client. An arc between a facility node and the root represents the possibility of opening the facility and its cost is equal to the opening cost, while high level facility assignment and device cost are taken into account by the arc between a client and the facility.

In the client cycle and mid level facility closing (CCF) neighborhood the possibility of opening a mid level facility (and therefore a high level facility) is not considered. The facility nodes are associated only to already opened facilities. However, the possibility of changing the facility device is considered. On this improvement graph negative cost cycles are sought such that a mid level facility is involved in at most one move at a time. Moves which are not allowed in the same sequence in the former neighborhood can be applied simultaneously in CCF neighborhood.

In the mid level facility cycle (MFC) neighborhood clients are not considered. The improvement graph nodes are associated to mid level facility and high level facility. An arc between two nodes representing mid level facilities i and j is associated to the possibility of assigning i to the high level facility to which j is assigned in the current solution, providing that j has relinquished. An arc between a mid level facility node and a high level facility node represents the possibility of changing the assignment of the mid level facility. The arc cost takes into account the increasing or decreasing of the assignment costs. Besides, opening and closing of high level facilities are represented in the neighborhood. A root node is added.

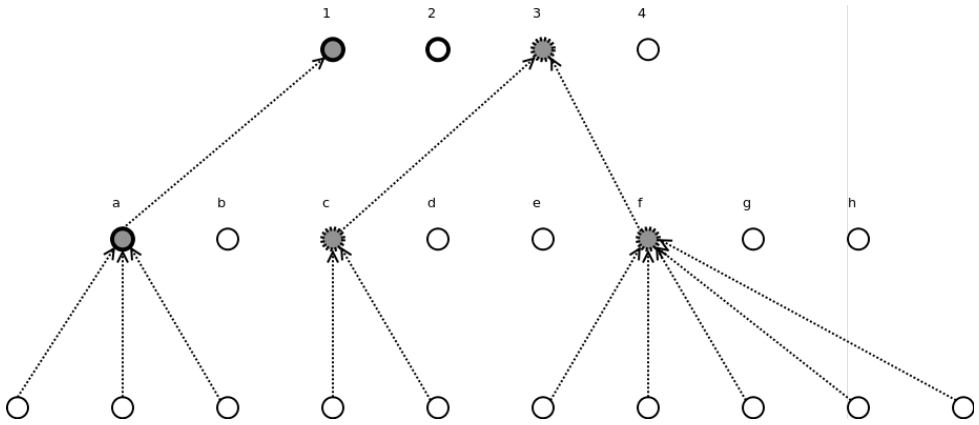

Figure 1: RANDOM very large scale neighborhood search.

An arc between the root and a mid level node represents the mid level facility relinquishing the current assignment, and its negative cost takes into account the assignment cost. An arc between a high level facility and the root takes into account the opening cost if the facility is not open in the current solution. In MFC neighborhood mid level facilities play the role that clients play in CCF, and high level facilities play the role that mid level facilities play in CCF.

Variable neighborhood search framework The above describe neighborhoods are combined to provide the Descent Phase of the heuristic. Each of the neighborhood is explored in a steepest descent fashion until the local minimum is reached. Then, the heuristic procedure starts to investigate another neighborhood. After preliminary computational experiments, we decided to explore the neighborhoods in the following order: CC, CCF, MCF, SCE, SFE. When SFE local minimum is reached, the procedure starts again from CC. If all the neighborhoods are explored and no improvement is found, the kick is applied.

3.2 Kick Phase

When no improvement is obtained using the above local search operators, we try to explore two very large scale neighborhoods using ILP based techniques.

Both are based on the idea of reducing model (1)–(7), so that the remaining problem can be effectively optimized by general purpose ILP solvers. Both neighborhoods are defined from a starting TLHCFLP solution $(\bar{x}, \bar{y}, \bar{w}, \bar{z})$.

RANDOM. In the first neighborhood we consider model (1)–(7), randomly fixing some high level location variables; when one of these variables represents an open facility in site k, we fix also the mid level location variables corresponding to mid level facilities assigned to k. Formally, we consider in turn each variable z_k, and we fix $z_k = \bar{z}_k$ with a probability α which is a parameter of the algorithm. Let J_k be the set of mid level facilities assigned to high level facility k in the starting solution, that is $J_k = \{j | \sum_{t \in \mathcal{T}} \bar{w}_{jtk} = 1\}$. Whenever a variable z_k is fixed, we fix also $y_{jt} = 1$ for each $j \in J_k$. For instance, a solution for a TLHCFLP instance with $N = 10$, $M = 8$ and $K = 4$ is depicted in Figure 1. Gray nodes represent sites where facilities are built, and arrows represent assignments of clients to mid level facilities and of mid level facilities to high level facilities. Let us assume that z variables corresponding to sites 1 and 2 are fixed by our random procedure (nodes with bold border in the figure): the y variable corresponding to site a is fixed as well, since site a contains a facility assigned to 1; any other location or assignment variable is left free, and the remaining problem is optimized.

LOCAL BRANCHING. The second neighborhood is inspired by local branching methods [3]. We introduce new constraints in the model, forcing a limit on the Hamming distance between the starting and any feasible solution. In particular, we add to model (1)–(7) the following constraints:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}|x_{ij}-\bar{x}_{ij}|\leq\beta$$

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}|y_{jt}-\bar{y}_{jt}|\leq\gamma$$

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}}|w_{jtk}-\bar{w}_{jtk}|\leq\delta$$

$$\sum_{k\in\mathcal{K}}|z_k-\bar{z}_k|\leq\phi$$

where β, γ, δ and ϕ are parameters of the algorithm. These constraints can be linearized with standard techniques; as reported in Section 4, we experimentally observed that general purpose solvers can effectively exploit the resulting model to produce solutions improving $(\bar{x},\bar{y},\bar{w},\bar{z})$.

4 Computational results

We implemented our heuristics in C++, using CPLEX 11.00 [7], with default parameter settings, but a time limit of two hours, to solve ILP subproblems. CPLEX relies on a state-of-the-art branch-and-cut method, which includes general purpose cut generation and primal heuristics. Our experiments ran on a Centrino Core2 3 GHz workstation equipped with 2GB of RAM.

In order to test our heuristics we considered three sets of instances. Dataset 1 consists of 71 instances drawn from [13] and adapted to TLHCFLP as described in [1]; in this dataset the number of clients range from 50 to 200 and the number of facilities from 10 to 50: it aims at testing our method on instances with a wide range of features but no particular structure. Dataset 2 includes 24 instances drawn from [14] and [4] and adapted to TLHCFLP in [1], in which the number of clients is 50 and the number of facilities range from 16 to 50. These instances have on the average a low ratio between the overall client demand and the high level facility capacities, and represent a stress-test for the existing exact algorithms. Finally, Dataset 3 consists of 12 harder large size instances; these are still drawn from [14] and [4], and adapted to TLHCFLP as described in [1]. Dual bounds are computed for all these instances using the exact methods described in [1]; for most instances in Dataset 1 and 2 these correspond to the optimal solution value, while we could not check their quality for instances in Dataset 3.

In a preliminary set of experiments two settings showed to be particularly appealing. The first one (Rbi) consists in using iteratively Descent Phase and RANDOM Kick Phase, setting $\alpha = 0.5$ and solving each ILP subproblem to optimality (that is, searching for the best improving move in the RANDOM neighborhood); the second one (LBfi) consists in using iteratively Descent Phase and LOCAL BRANCHING Kick Phase, setting $\alpha = \delta = +\infty$, $\beta = \phi = 2$, and stopping the optimization of each ILP subproblem as soon as an improving solution is found (that is, searching for a first improving move in the LOCAL BRANCHING neighborhood). At most 10 Descent - Kick iterations are performed in each test.

Tables 1, 2 and 3 report the comparison of Rbi and LBfi heuristics respectively on Dataset 1, Dataset 2 and Dataset 3. The first block of each table reports the name of the instance and the number of clients, candidate mid level location sites and candidate high level location sites. A block follows for each heuristic, reporting the percentage gap with respect to the best known dual bound, the number of Descent - Kick iterations performed and the CPU time spent. The last line of each table reports average values over instances in the dataset; the average over instances in Dataset 3 does not include instances capa10000 capa12000 and capa14000.

Table 1: Comparison of Kicks - Dataset 1

				Rbi			LBfi		
inst.	N	M	K	gap	# kicks	cpu time (s)	gap	# kicks	cpu time (s)
p10	50	10	10	0.00%	1	0.57	0.79%	7	135.45
p11	50	10	10	0.00%	3	4.76	0.00%	4	2.35
p12	50	10	10	24.91%	0	0.05	0.00%	3	1.91
p13	50	20	20	1.06%	4	8.14	0.01%	4	9.80
p14	50	20	20	5.07%	1	18.79	0.00%	3	5.71
p15	50	20	20	0.60%	3	9.16	0.00%	5	8.28
p16	50	20	20	0.72%	1	2.90	0.00%	4	20.33
p17	50	20	20	0.01%	2	8.93	0.01%	4	9.79
p18	50	20	20	0.00%	6	8.63	0.00%	3	5.69
p19	50	20	20	0.80%	3	2.48	0.00%	5	8.26
p1	50	10	10	0.01%	1	152.33	0.01%	4	60.76
p20	50	20	20	4.89%	0	0.13	0.00%	4	20.38
p21	50	20	20	3.38%	3	4.49	0.00%	6	9.90
p22	50	20	20	3.72%	3	4.42	0.00%	6	7.22
p23	50	20	20	0.38%	5	7.57	0.70%	6	10.54
p24	50	20	20	0.00%	1	14.40	0.00%	9	19.35
p25	150	30	30	8.63%	1	7.56	0.00%	5	47.75
p26	150	30	30	0.00%	1	9.03	0.00%	2	12.58
p27	150	30	30	3.23%	1	24.48	0.01%	4	42.15
p28	150	30	30	0.00%	2	35.68	0.00%	7	130.81
p29	150	30	30	8.64%	2	4.45	0.01%	5	30.72
p2	50	10	10	1.61%	3	1.09	0.52%	10	18.80
p30	150	30	30	18.12%	1	3.94	0.00%	4	33.91
p31	150	30	30	0.75%	4	32.37	0.01%	8	45.09
p32	150	30	30	22.06%	2	12.37	0.00%	4	15.90
p33	150	30	30	1.41%	1	23.10	0.00%	5	50.36
p34	150	30	30	0.00%	3	17.31	0.00%	2	13.58
p35	150	30	30	2.61%	2	7.15	0.01%	4	47.29
p36	150	30	30	0.00%	1	32.71	0.00%	7	155.48
p37	150	30	30	0.00%	1	12.22	0.00%	7	65.91
p38	150	30	30	0.00%	1	8.58	0.00%	6	39.09
p39	150	30	30	0.00%	4	33.35	0.00%	6	47.66
p3	50	10	10	2.47%	2	0.72	0.01%	9	595.71
p40	150	30	30	32.51%	1	22.48	0.00%	3	20.31
p41	90	10	10	0.00%	1	4.40	0.00%	4	6.17
p42	80	20	20	17.20%	0	3.88	0.45%	9	96.64
p43	70	30	30	0.00%	1	20.78	2.70%	6	106.22
p44	90	10	10	0.19%	1	1.43	0.00%	3	6.65
p45	80	20	20	31.91%	0	0.68	0.00%	7	33.12
p46	70	30	30	0.00%	1	75.99	1.02%	6	58.65
p47	90	10	10	1.17%	1	0.34	0.01%	3	7.97
p48	80	20	20	15.53%	2	1406.62	0.01%	4	7221.20
p49	70	30	30	0.19%	1	30.14	0.00%	9	59.78
p4	50	10	10	3.92%	2	1.15	0.39%	10	6013.87
p50	100	10	10	14.88%	1	1.72	0.04%	10	11.44
p51	100	20	20	0.00%	1	12.63	0.01%	10	30.38
p52	100	10	10	0.00%	3	0.92	0.00%	6	6.71
p53	100	20	20	0.00%	1	3.48	0.00%	4	8.81
p54	100	10	10	0.49%	1	0.94	0.00%	3	1.71
p55	100	20	20	0.01%	2	2.23	0.01%	5	13.53
p56	200	30	30	1.01%	3	1991.31	1.03%	10	549.81
p57	200	30	30	5.44%	1	2301.60	1.33%	10	211.85
p58	200	30	30	0.64%	4	14499.00	0.82%	10	3038.70
p59	200	30	30	0.80%	3	988.99	1.09%	10	350.90
p5	50	10	10	6.42%	0	0.04	0.00%	4	9.49
p60	200	30	30	0.21%	2	7230.52	0.67%	10	88.62
p61	200	30	30	2.45%	3	500.85	0.43%	10	93.54
p62	200	30	30	18.22%	0	4.86	1.85%	10	194.80
p63	200	30	30	0.53%	2	7505.30	2.42%	10	163.62
p64	200	30	30	11.39%	1	426.22	1.50%	9	225.93
p65	200	30	30	0.01%	1	869.77	1.46%	10	275.40
p66	200	30	30	0.91%	1	14404.90	1.03%	10	2489.65
p67	200	30	30	1.77%	1	578.06	0.01%	10	409.75
p68	200	30	30	10.07%	0	6718.81	0.67%	10	91.39
p69	200	30	30	1.54%	3	4331.82	0.43%	10	103.32
p6	50	10	10	2.01%	2	0.70	0.00%	2	1.15
p70	200	30	30	0.69%	3	7742.91	1.80%	10	207.87
p71	200	30	30	0.01%	2	7544.48	2.42%	10	169.46
p7	50	10	10	2.93%	2	1.82	0.00%	6	5.89
p8	50	10	10	0.00%	2	4.34	0.00%	7	7.15
p9	50	10	10	9.44%	1	0.29	0.00%	2	1.34
Overall				4.36%	1.77	1155.73	0.36%	6.39	349.46

Table 2: Comparison of Kicks - Dataset 2

				Rbi			LBfi		
inst.	N	M	K	gap	# kicks	cpu time (s)	gap	# kicks	cpu time (s)
cap101	50	25	25	0.00%	1	1.64	0.52%	8	7.94
cap102	50	25	25	35.11%	0	2.28	0.12%	5	7.97
cap103	50	25	25	12.96%	0	1.27	0.00%	7	8.62
cap104	50	25	25	0.00%	1	1.89	0.00%	5	6.87
cap121	50	50	50	0.01%	1	22.75	0.01%	2	33.76
cap122	50	50	50	0.01%	1	292.70	7.63%	2	461.04
cap123	50	50	50	0.01%	1	412.36	0.96%	5	98.85
cap124	50	50	50	10.09%	1	540.40	0.01%	10	43.57
cap131	50	50	50	1.36%	1	16.20	0.00%	10	31.73
cap132	50	50	50	0.00%	1	5.67	0.00%	7	23.67
cap133	50	50	50	0.00%	1	7.41	0.00%	9	17.90
cap134	50	50	50	26.28%	0	59.37	0.00%	3	17.08
cap51	50	16	16	1.50%	1	0.95	0.00%	5	4.13
cap61	50	16	16	0.00%	1	1.61	0.00%	9	5.08
cap62	50	16	16	5.21%	2	0.93	0.01%	10	3.71
cap63	50	16	16	0.00%	3	2.49	0.00%	10	4.68
cap64	50	16	16	0.00%	1	0.68	0.00%	9	2.13
cap71	50	16	16	5.91%	0	0.94	0.00%	5	2.98
cap72	50	16	16	4.94%	0	0.59	0.00%	5	2.91
cap73	50	16	16	0.00%	1	1.20	0.00%	4	2.86
cap74	50	16	16	0.00%	1	1.36	0.36%	1	2.48
cap91	50	25	25	1.66%	2	1.08	0.30%	9	20.78
cap92	50	25	25	0.00%	1	1.68	0.00%	10	5.22
cap93	50	25	25	0.00%	1	1.25	0.00%	9	5.30
cap94	50	25	25	13.98%	1	1.61	0.94%	10	9.24
Overall				4.76%	0.96	55.21	0.43%	6.76	33.22

Table 3: Comparison of Kicks - Dataset 3

				Rbi			LBfi		
inst.	N	M	K	gap	# kicks	cpu time (s)	gap	# kicks	cpu time (s)
capa10000	1000	100	100	513.84%	0	383.40	513.84%	1	4836.85
capa12000	1000	100	100	-	0	83.48	757.65%	0	162.07
capa14000	1000	100	100	-	0	145.04	901.47%	0	191.04
capa8000	1000	100	100	5.01%	2	21703.40	7.73%	10	5087.22
capb5000	1000	100	100	5.63%	2	7768.38	19.26%	10	5113.36
capb6000	1000	100	100	47.36%	0	7234.76	47.36%	10	25.58
capb7000	1000	100	100	61.49%	1	14467.50	61.61%	10	15.96
capb8000	1000	100	100	5.48%	3	21826.10	25.87%	10	14.72
capc5000	1000	100	100	58.41%	0	7231.85	58.41%	10	18.48
capc5750	1000	100	100	49.57%	1	14477.60	49.96%	10	21.58
capc6500	1000	100	100	49.07%	0	7242.53	49.07%	10	21.40
capc7250	1000	100	100	11.38%	4	23401.70	68.20%	10	24.11
Overall				32.60%	1.44	13928.20	43.05%	10.00	1149.16

First, by looking at the Average values at the bottom of each table, we observed that LBfi outperformed Rbi in both Dataset 1 and Dataset 2, both in terms of accuracy and CPU time. Rbi provided on the average better quality solutions on Dataset 3, at the expense of much higher CPU time.

Second, we observed that LBfi was able to reach an optimal solution in a large set of instances. We also observed that, when optimality is not reached, LBfi often hit the limit on the maximum number of Descent-Kick iterations; therefore, we conjecture that by raising such a limit, the quality of these solutions could be further improved.

Third, we compared the results of LBfi with that of the exact algorithms described in [1]; for instance, in Dataset 1 we observed that when the exact algorithm is able to solve an instance within a time limit of 2 hours, LBfi reaches solutions which are on the average 0.16% worse than optimum, and is about three times faster; on the remaining instances the solutions of LBfi are 0.73% away from that of the exact algorithm, and the average CPU time is less than one tenth.

5 Conclusions

In this paper we proposed effective heuristics for a Two-level facility location problem arising in telecommunications network design, namely the Two-level Hierarchical Capacitated Facility Location Problem. We designed and experimentally tested two algorithms; both model and explore very large scale neighborhoods using ILP based formulations and techniques. In particular, one of them is able to reach optimal solutions for a large set of instances, effectively tackles large size instances, and provides on the average very tight primal bounds in a fraction of the CPU time spent by exact algorithms.

Acknowledgments. The authors are grateful to Maria Paola Scaparra for kindly providing the source code used in [4].

References

[1] B.Addis, G. Carello, A.Ceselli. Exactly solving a Two-level Hierarchical Location Problem with modular node capacities. *Optimization Online*, Report ID 2010-02-2552, 2010.

[2] K. Aardal, F.A. Chudak, and D.B Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, 72:161–167, 1999.

[3] M. Fischetti, A. Lodi. Local branching *Mathematical Programming*, 98(1):23–47, 2003.

[4] R.K. Ahuja, J.B. Orlin, S. Pallottino, M.P. Scaparra, and M.G. Scutellà. A multi-exchange heuristic for the single source capacitated facility location problem. *Management Science*, 50(6):749:760, 2004.

[5] S. Chamberland. An Efficient Algorithm for Designing Reliable IP Networks with an Access/Edge/Core Hierarchical Structure. *Networks 2008*, conference presentation, Budapest, 2008.

[6] P. Chardaire, J.-L. Lutton, and A. Sutter. Upper and lower bounds for the two-level simple plant location problem. *Annals of Operations Research*, 86:117–140, 1999.

[7] ILOG CPLEX 11.0 Users Manual. ILOG Inc, 2007

[8] A.A.V.. Ignacio, V.J.M.F. Filho, and R.D. Galvao. Lower and upper bounds for a two-level hierarchical location problem in computer networks. *Computers and Operations Research*, 35:1982–1998, 2008.

[9] G. Sahin and H. Süral. A review of hierarchical facility location models. *Computers and Operations Research*, 34:2310–2331, 2007.

[10] D. Tcha and B. Lee. A branch-and-bound algorithm for the multi-level uncapacitated location problem. *European Journal of Operations Research*, 18:35–43, 1984.

[11] S. Tragantalerngsak, J. Holt, and M. Ronnqvist. An exact method for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 123:473–489, 2000.

[12] M. Trampont, C. Destr, and A. Faye. Solving a hierarchical network design problem with two stabilized column generation approaches. *INOC 2009 - International Network Optimization Conference*, Pisa, 2009.

[13] K. Holmberg, M. Ronnqvist, D. Yuan. An exact algorithm for the capacitated facility location problem with single sourcing. *European Journal of Operational Research*, 113:544-559, 1999.

[14] J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of Operational Research Society*, 41:1069-1072, 1990.

[15] P. Hansen, N. Mladenovich. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449-467, 2001.

[16] R.K. Ahuja, Ö. Ergun, J.B. Orlin, A.P. Punnen, A survey of very large-scale neighborhood search techniques, *Discrete Applied Mathematics*, 123:75–102, 2002.