A GRASP Algorithm for the Connected Facility Location Problem

Alessandro Tomazic* Ivana Ljubić†
Faculty of Business, Economics, and Statistics
University of Vienna
Brünnerstr. 72, 1210 Vienna, Austria
{alessandro.tomazic, ivana.ljubic}@univie.ac.at

Abstract

We apply a Greedy Randomized Adaptive Search Procedure (GRASP) to solve the Connected Facility Location Problem heuristically. Diversification property is assured by applying a randomized greedy algorithm to construct feasible solutions in a multi-start fashion. Intensification elements are guaranteed due to two facility-based local search techniques. The computational study is conducted on a parameterized set of randomly generated benchmark instances. The obtained results reflect the quality of the proposed approach with respect to both, the quality of solutions and the computational effort, by comparison with lower bounds obtained from a Branch-and-Cut framework.

1 Introduction

Given an undirected connected graph G with vertices V and the set of edges E, a nontrivial partition $P = (F, C)$ of V identifying *facilities* and *customers*, *edge costs* $c : E \mapsto \mathbb{Q}^+$ and a function $p : F \mapsto \mathbb{Q}^+$ that defines the *opening costs* for the facilities, the *Connected Facility Location Problem* is defined as finding a connected subtree $T = (V[T], E[T])$, such that $C \subset V[T]$ and $G[F \cap V[T]]$ is connected, that minimizes the objective function

$$\sum_{v \in V[T] \cap N_T(C)} p(v) \; + \; \sum_{e \in E[T]} c(e) \; .$$

With $N_T(C)$, we denote the neighboring nodes of C in T. In a solution T with an edge connecting a facility v with customer u, we say that v is *open* and u is *supplied* by v. *Potential suppliers* F_p are those nodes from F that have at least one neighbor in C, i.e. $F_p = N_G(C) \cap F$.

It follows immediately from the definition that in every feasible solution customer nodes must be leaves. Therefore, given a graph G, edges between two customers can be deleted, without loss of generality.

In the design of telecommunication networks the following problem can be modelled as ConFL: build a last-mile network by replacing outdated copper- by fiber-optic-connections, thereby placing multiplexers to switch between them. Connect then multiplexers to each other, so that connection- and installation-costs are minimized.

The ConFL Problem has been introduced in [4], and the best-known approximation ratio of 4.23 has been obtained recently by Eisenbrand et al. [2]. Ljubić [5] concentrated on the rooted version of the problem, in which a facility r is open at no costs.

In the next Section we show how to integrate a greedy heuristic and two local-search methods in a GRASP framework. Section 3 describes how the problem can be transformed into the Minimum Steiner Arborescence Problem (SA) in graphs. We also propose to solve the SA using a branch-and-cut algorithm (B&C) whose bounds are then used in in section 4 to measure the quality of our heuristic approach. Section 5 gives some conclusions and ideas for improving the algorithm.

2 The GRASP Algorithm

A GRASP [3] is a multi-start iterative approach for combinatorial optimization problems where each iteration consists of two phases: greedy construction and local improvement. The best overall solution is reported as the final one.

In every iteration of the GRASP algorithm, we use a randomized *Greedy* procedure to construct a feasible solution. As a search intensification mechanism, we iteratively apply *open-* and *close-facility* moves, followed by a shortest path Steiner tree heuristic to find locally optimal solutions. The whole process is iterated in a multi-start fashion until a prespecified number of iterations is reached.

Diversification property is assured by dynamically increasing the number of potential candidates to be inserted within the randomized greedy procedure.

*Supported by the Austrian Research Promotion Agency, FFG, within Bridge 2 programme (812973)
†Supported by the Hertha-Firnberg Fellowship of the Austrian Science Foundation (FWF)

978-0-7695-3297-4/08 $25.00 © 2008 IEEE
DOI 10.1109/SAINT.2008.64

2.1. The randomized Greedy construction

For the construction phase a deterministic *Greedy* heuristic presented below is modified. The algorithm incrementally opens potential suppliers and connects them into a tree T, until all customers are supplied.

Greedy Algorithm The potential supplying facilities F_p are sorted in increasing order according to the following criterion:

$$\frac{p(v)dist(v,T)}{deg_{C'}(v)+1}, \quad v \in F_p \tag{1}$$

where $dist(v,T)$ is the length of a shortest path from v to a vertex in T with respect to c. If $T = \emptyset$ or $v \in V[T]$, $dist(v,T) = \min_{u,v \in F} c(u,v)$. $deg_{C'}(v)$ is the number of unsupplied customers that can be served by v. Hereby, a higher priority is given to the vertices close to the partially constructed subtree T, incident to many unsupplied customers or with low opening costs.

Among all closed potential suppliers, let v be the best one according to (1). We open v and connect all the neighboring customers to it if not already supplied cheaper. Then we extend T by adding the shortest path from v to T. Afterwards facilities are reassigned where appropriate, and if some of the previously opened facilities gets closed, the facility network is redesigned by application of the *Shortest Path Heuristic*[7] described below.

The algorithm iteratively opens facilities in this manner until all the customers are supplied.

Shortest Path Steiner Tree Heuristic (SPH) For a set of open facilities we use a heuristic to redesign the facility network. The basic subproblem at this stage is the Steiner Tree Problem with the set of open facilities chosen as terminals and the remaining facilities as Steiner nodes. Therefore we apply the well known Shortest Path Heuristic that efficiently returns an approximation of the optimal connection between open facilities. Hereby the tree is iteratively extended by adding the shortest path to the nearest unconnected terminal node starting with an arbitrary one.

Denote with $G[F]$ the subgraph induced by F, and with $E[F]$ the corresponding set of edges. The computational complexity of the SPH is $O(|F|(|E[F]|\log|F|)$ (see [1] for the explanation of an efficient implementation using Dijkstra's shortest path algorithm and binary heaps).

Randomized Greedy In a multistart version with starting alternatives, we add a tolerance to the selection of facilities to open. Our candidate list of length k will simply consist of the first k vertices in F_p - sorted with respect to (1) - that are not opened yet. So in each step of the method we randomly choose one among those k facilities. Setting the parameter k to 1 corresponds to the non-randomized version of the algorithm. In our implementation the number of candidates is dynamically set within the GRASP procedure and applies for a complete *Greedy* run. We begin with $k = 1$ and linearly (rounded to integers) increase it up to $3/20$ of the number of vertices available after each *Greedy* insertion.

Since in every iteration the Dijkstra algorithm and the SPH are called, the total runtime complexity of *Greedy* is $O(|F|(|F||E[F]|\log|F| + |C|))$.

2.2. Local Search Techniques

To intensify the search in the local regions of a such constructed greedy solution, we explore *open-* and *close-facility* neighborhoods in two phases: we first search for the locally optimal solution with respect to the open-facility neighborhood, in the second phase we search for the improvement by sequentially applying *close-facility* moves. In the case of facility closure the *Shortest Path Heuristic* is reapplied.

Our computational studies have shown that further repetitions of the two phases would be rather time consuming than beneficial. Therefore we decided to apply them only once per GRASP iteration.

A set of references related to the neighborhood search techniques for the Steiner Tree Problem can be found in [1], for example.

Open-facility Moves A closed potential supplier $v \in F_p$ is opened to check, whether supplying customers results in cost reduction. Thereby we consider the opening costs of v, the costs for connecting customers and the savings of previously paid connection costs for each reconnected customer. If this reduces the objective value, we perform the operations. When reconnecting customers we may end up in finding an open facility u not supplying any customers. In this case u can be closed and its opening costs are taken into account when evaluating the opening of v. Computational complexity of a single move is $O(|F||E[F]|\log|F| + |C|)$. For the total exploration of the neighborhood we need $O(|F|(|F||E[F]|\log|F| + |C|))$ time.

Close-facility Moves Closing a facility v can only result in a feasible solution when all the customers can be supplied by other open facilities. Here we focus on the open ones and try to reconnect its customers such that it costs us less than we gain by closing v. The closure of a facility may also have impact on the structure of our facility network. Since v isn't indispensable because of its supplying function anymore, solving the corresponding Steiner Tree Problem would return an optimal facility network for the current supply situation. This is approximated by running the Shortest Path Steiner Tree Heuristic described previously. A single move in that case has a computational complexity of $O(|F|(|E[F]|\log|F| + |C|))$. For the total exploration of the neighborhood we need $O(|F|^2(|E[F]|\log|F|+|C|))$ time. By using priority queues we might improve this computational complexity. When exploring the neighborhoods, we consider *first improvement* strategy where facilities to be opened/closed are always processed in a randomized order

to prevent the algorithm from getting stuck at the same local optimum.

2.3. Integration in the GRASP-Framework

The *Greedy*-heuristic together with the local search techniques are embedded in a GRASP framework as described in Algorithm 1.

Algorithm 1 The GRASP-Framework

Input: $(G = (V, E), c, p)$
Output: A feasible ConFL solution (T, A)
1: **for** $iter = 0$ to $iter = |F_p|/5$ **do**
2: $(T, A) = Greedy(G, c, p)$
3: $OpenFacilityLocalSearch(G, c, p, T, A)$
4: $CloseFacilityLocalSearch(G, c, p, T, A)$
5: **end for**

3 Relation to the Minimum Steiner Arborescene Problem

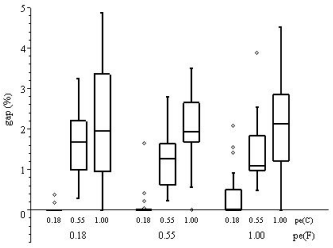
Given a directed connected graph $G_A = (V_A, A)$ with costs function on the arcs $w : A \mapsto \mathbb{Q}^+$, with a root $r \in V_A$ and a set of *terminals* $T_A \subset V_A$, the *Minimum Steiner Arborescence Problem* searches for a rooted subtree of G_A of minimum costs, such that there is a directed path from r to every $v \in T_A$.

Every ConFL problem on a graph $G = (V, E)$ can be transformed into a SA problem in the following way:

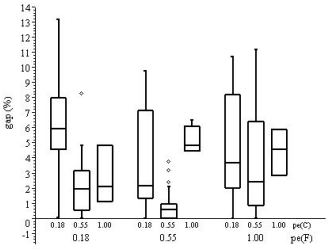
1. Introduce an artificial root r and for every potential supplier $v \in F_p$ its counterpart v'. Set $V_A = V \cup \{r\} \cup \{v' \mid v \in F_p\}$.
2. Connect all potential suppliers to r, i.e. set $A = \{(r, v) \mid v \in F_p\}$ and $w(r, v) = 0 \forall v \in F_p$.
3. Connect all facilities with each other: $A = A \cup \{(u, v), (v, u) \mid u, v \in F\}$ and $w(u, v) = w(v, u) = c(u, v)$, $u, v \in F$.
4. Split potential suppliers: $A = A \cup \{(v, v') \mid v \in F_p\}$. Set $w(v, v') = p(v)$, $v \in F_p$.
5. Connect facilities and clients only in one direction: $A = A \cup \{(v, v_c) \mid v \in F_p, v_c \in C\}$ and $w(v, v_c) = c(v, v_c)$.

In such obtained instance there are obviously no outgoing arcs from any customer node $v_c \in C$, and there are only outgoing arcs from the root r. To assure feasibility of a ConFL solution that is obtained by removing r and its adjacent arcs, and contracting splitted nodes again, we request that the outgoing degree of r must be equal to one.

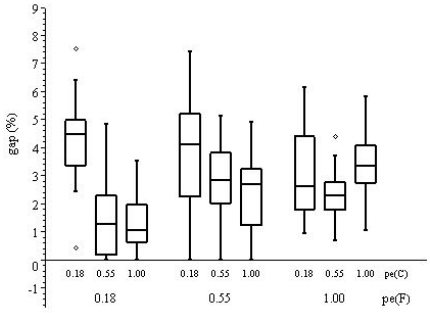
To solve the minimum SA problem on G_A to optimality, we use an adaptation of the branch-and-cut algorithm proposed in [6]. The bounds obtained using this algorithm are presented in the next Section.



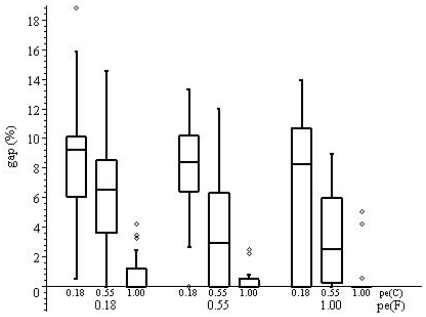
(a) SET1: $|F| = 20$, $|C| = 100$, $t_{avg} \approx 3s$



(b) SET1: $|F| = 100$, $|C| = 20$, $t_{avg} \approx 44s$



(c) SET1: $|F| = 50, |C| = 50, t_{avg} \approx 15s$



(d) SET3: $|F| = 50, |C| = 50, t_{avg} \approx 15s$

Figure 1. Percentage gaps between GRASP solutions and optimal values.

4 Computational Results

The procedure was tested on a set of randomly generated graphs with random integer weights[1]. Edges of the network

[1]All instances are available at `homepage.univie.ac.at/alessandro.tomazic`.

$G[F]$ are generated with probability $pe(F) \in \{0.1, 0.5, 1\}$, whereas connections between facilities and customers are established with probability $pe(C) \in \{0.18, 0.55, 1\}$. Lower and upper bounds for the edge weights were set to 50 and 100. We generated two sets, **SET1** ($|F| \times |C| \in \{100 \times 20, 20 \times 100, 50 \times 50\}$) and **SET2** ($100 \times 100$) of 81 and 27 instances, respectively, with facility opening costs randomly assigned to values between 150 and 200. For 27 additional 50×50 instances (**SET3**) we randomly assigned the facility opening costs to values between 800 and 1600.

After letting MAPLE generate a random facility network $G[F]$, we randomly link customers to the existing vertices using the parameters given above. This resulted in instances where almost always every facility was a potential supplier.

Our C++ implementation of the GRASP was tested on an Intel Core 2 Duo E4300 with 1.8 GHz, 3.25 GB RAM.

SET1 and SET3: Each single box in the diagrams of Figure 1 reflects the results of 10 GRASP runs on 3 random instances with the same construction properties (30 results). The computation time that our method needed did not exceed two minutes. The exact method completed the computations in 5 minutes on average for the instances of SET1 and SET3, but needed almost 30 minutes for three of them. Figure 1 depicts percentage gaps between GRASP solutions and optimal solutions (OPT) calculated as: $gap = (\mathrm{GRASP} - \mathrm{OPT})/\mathrm{OPT}[\%]$.

We observe that GRASP easily finds optimal solutions if $|F|$ is small when compared to $|C|$ and the connections between F and C are sparse (Figure 1(a)). With increasing density $pe(C)$, the performance gets worse, but median gaps are still within 2% of optimum.

For instances whose number of facilities is large when compared to the number of customers (Figure 1(c)) we observe that GRASP has difficulties to deal with, by providing solutions whose median gaps are between 1% and 6% of optimum.

Given the same graph topology, we also tested the influence of the cost structure to the GRASP performance, by comparing 50×50 instances of the SET1 and the SET3 (Figures 1(c), 1(d)). There is obviously no direct dependency between the quality of obtained solutions and the parameters $pe(F)$ and $pe(C)$ of the SET1 group. However, when average facility opening costs are by an order of magnitude higher than the average connection costs (SET3), GRASP solves graphs with complete bipartite structure between F and C ($pe(C) = 1$) very efficiently (median gap less than 1%), and had difficulties with sparse structures ($pe(C) \in 0.18, 0.55$).

SET2: For 27 instances of size 100×100, after running B&C for one hour, only one instance was solved to optimality, and for three of them not even a feasible solution was found. On the contrary, the GRASP results were all obtained within less than 5 minutes, and in less than 2 minutes on average.

Group		LB-gap		UB-gap		GRASP	B&C-gap
$pe(F)$	$pe(C)$	avg	med	avg	med	$t[s]$	avg
0.1	0.18	8.6	8.8	–	–	41.5	–
0.1	0.55	5.5	5.8	3.2	1.9	123.9	9.1
0.1	1.0	5.7	5.5	-2.2	-2.0	69.7	3.4
0.5	0.18	6.1	6.3	-0.6	-0.5	106.0	5.5
0.5	0.55	5.8	6.1	-0.4	0.2	27.9	5.4
0.5	1.0	5.8	5.9	0.0	-0.2	185.5	5.8
1.0	0.18	4.7	4.4	-1.7	-1.7	275.1	3.0
1.0	0.55	4.9	4.7	-1.1	-0.7	46.8	3.8
1.0	1.0	3.4	3.2	-2.0	-2.0	15.6	1.4

Table 1. GRASP vs. branch-and-cut results for SET2 instances (100×100).

In Table 4, for three instances within a group and for 10 runs per instance, we report the following values: LB-$gap = (\mathrm{GRASP} - \mathrm{LB})/\mathrm{LB}$, the average and median gaps between the GRASP solution and the B&C lower bound (LB); UB-$gap = (\mathrm{UB} - \mathrm{GRASP})/\mathrm{UB}$, the average and median gaps between the GRASP solution and the B&C upper bound (UB); average GRASP running time in seconds ($t[s]$) and the optimality gap of the B&C obtained as $(\mathrm{UB} - \mathrm{LB})/\mathrm{LB}$.

The most difficult instances for B&C are the sparse graphs, where GRASP even outperforms the upper bounds found by B&C (positive UB-gap values) after one hour.

We conclude that, on three sets of randomly generated instances with uniform topology and different cost structures, GRASP performs fast and provides stable results whose average gap to optimum varies between 0% and 10%. For both approaches, B&C and GRASP, the most difficult (easiest) instances appear to be those with sparse (dense) customer-facility topologies when $|F| = |C|$.

References

[1] M. P. de Aragão, C. C. Ribeiro, E. Uchoa, and R. Werneck. Hybrid local search for the Steiner problem in graphs. In *Extended Abstracts of MIC 2001*, pages 429–433, Porto, Portugal, 2001.

[2] F. Eisenbrand, F. Grandoni, T. Rothvoß, and G. Schäfer. Approximating connected facility location problems via random facility sampling and core detouring. In *Proceedings of SODA*, pages 1174–1183, 2008.

[3] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[4] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *FOCS*, pages 613–623, 2000.

[5] I. Ljubić. A hybrid VNS for connected facility location. In *Hybrid Metaheuristics 2007*, pages 157–169, 2007.

[6] I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Progamming, Series B*, 105(2-3):427–449, 2006.

[7] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica*, 6:573–577, 1990.