Variable Neighborhood Search for a Prize Collecting Capacity Constrained Connected Facility Location Problem

Markus Leitner*
Carinthia University of Applied Sciences
School of Telematics / Network Engineering
Klagenfurt, Austria
markus.leitner@fh-kaernten.at

Günther R. Raidl
Vienna University of Technology
Institute of Computer Graphics and Algorithms
Vienna, Austria
raidl@ads.tuwien.ac.at

Abstract

We present a Variable Neighborhood Search approach for a network design problem occurring in real world when the bandwidth of an existing network shall be enhanced. Using two different neighborhood structures we show that a carefully designed combination of a metaheuristic and an exact method based on integer linear programming is able to improve solution quality compared to using heuristic methods only.

1. Introduction

We consider a connected facility location problem where only clients which are reasonable from an economic point of view need to be served (prize collecting variant) with capacity constraints on possible facilities (PCConFL). PCConFL arises when increasing the available bandwidth of an existing network by installing additional fibre optic routes. Instead of connecting clients directly (*fibre to the home*), facilities which act as mediation points between the newly installed and original network are introduced and connected. Each facility will serve several clients that are physically near to it up to some maximum capacity.

More formally, we are given a connected undirected graph $G = \langle V, E, c\rangle$ with node set V, edge set E and edge costs $c(u,v) \in \mathbb{N}$, $\forall (u,v) \in E$. Each edge $(u,v) \in E$ represents a possibly new fibre optic route while its costs $c(u,v)$ correspond to the expenses of installing it. Furthermore, we are given clients C, possible facilities $F \subseteq V$ and a root node $r \in V$. The latter models the whole existing fibre optic infrastructure to which new facilities must be connected. Clients $k \in C$ have demands $d_k \in \mathbb{N}$, prizes $p_k \in \mathbb{N}$ and a given set of potential facilities $F_k \subseteq F$ they may be assigned to. While each client may be assigned to at most one facility, each facility can serve clients up to a maximum total demand $d_{\max}$.

A solution to PCConFL $S = \langle F_{\mathrm{S}}, C_{\mathrm{S}}, G_{\mathrm{T}}\rangle$ consists of a set of facilities $F_{\mathrm{S}} \subseteq F$ to be opened, a set of customers $C_{\mathrm{S}} \subseteq C$ that will be served by F_{S} and a Steiner tree $G_{\mathrm{T}} = \langle V_{\mathrm{S}}, T_{\mathrm{S}}\rangle$ with node set $V_{\mathrm{S}} \subseteq V$ and edge set $T_{\mathrm{S}} \subseteq E$ connecting all facilities in F_{S} with the root node r. The profit of a solution S is given by the sum of prizes of all served clients minus the cost of the Steiner tree connecting opened facilities , i.e. $o(S) = \sum_{k \in C_{\mathrm{S}}} p_k - \sum_{(u,v) \in T_{\mathrm{S}}} c(u,v)$, and the objective is to maximize this profit. The root node r is treated as a facility that may not be removed and is therefore not considered explicitly in the following.

2. Related Work

To the best of our knowledge PCConFL has not been considered so far. However, several authors studied the Connected Facility Location Problem (ConFL). In ConFL facilities do not have capacity constraints, all clients need to be served, opening costs for facilities and costs for assigning clients to them are given. Karger and Minkoff [4] motivated the problem by the so-called *maybecast* and presented the first constant factor approximation. Swamy and Kumar [9] gave primal-dual approximation algorithms with factor 8.55 for ConFL and a 4.55 approximation for the *rent-or-buy* problem in which all opening costs are zero and each node is a possible facility (i.e. $F = V$). Hasan et al. [3] improved the result of Swamy and Kumar by proposing a 8.29 approximation algorithm for ConFL based on LP-rounding and a factor 7 approximation when all opening costs are equal. Recently Ljubić [6] presented a hybrid Variable Neighborhood Search approach utilizing a simple swap neighborhood as well as a branch-and-cut approach to solve ConFL exactly.

*This work is supported by the Austrian Research Promotion Agency (FFG) under grant 811378.

Algorithm 1: Facility Swap (Facilities F_{S})

```
$\hat{F} = F_{\mathrm{S}}$
forall $f \in F$ do
    $F' = F_{\mathrm{S}} \Delta \{f\}$
    derive Steiner tree $G_{\mathrm{T}}$ for $F'$
    derive customer assignment for $F', G_{\mathrm{T}}$
    if $o(F') > o(\hat{F})$ then
        $\hat{F} = F'$
$F_{\mathrm{S}} = \hat{F}$
```

3. The Heuristic

To solve PCConFL we consider its three mutually strongly dependent subproblems, which are (1) selecting facilities, (2) connecting them by a Steiner tree and (3) assigning clients to selected facilities.

Our heuristic optimizes the selection of facilities by Variable Neighborhood Search (VNS) using the neighborhood structures described in Section 3.1 while the determination of the Steiner tree (see Section 3.2) as well as the assignment of customers to facilities (see Section 3.3) is carried out by second and third level heuristics which are called for each candidate set of facilities. For the VNS, it is therefore sufficient to represent a solution by its set of facilities F_{S}.

3.1. Choosing Facilities

3.1.1. Facility Swap Neighborhood

The Facility Swap Neighborhood (FSN) of a solution consists of all solutions that differ from an initial solution F_{S} by exactly one open facility; i.e. either one facility may be removed or added. Algorithm 1 shows how it is searched. Therefore, FSN consists of $|F|$ neighboring solutions.

3.1.2. Facility Exchange Neighborhood

Instead of simply swapping a single facility as done by FSN, the Facility Exchange Neighborhood (FEN) replaces one facility by another one, see Algorithm 2. Here, each candidate solution has $|F_{\mathrm{S}}| \cdot |F \setminus F_{\mathrm{S}}| = O(|F|^2)$ neighbors.

3.2. Connecting Facilities

We use the Minimum Spanning Tree (MST) heuristic [7] for the Steiner Tree Problem to connect opened facilities $f \in F_{\mathrm{S}}$ by a Steiner tree G_{T}. Initially the MST heuristic computes a MST G'_{T} in the distance network which is a complete graph with node set F_{S} and edge costs $d(u, v)$ corresponding to the costs of the shortest path between u and v in G, $\forall u, v \in F_{\mathrm{S}}$. Afterwards, G_{T} is initialized by the subgraph of G induced by all edges part of any shortest path corresponding to an edge in G'_{T}. If G_{T} is not yet a tree, the MST heuristic reduces it to a tree and finally recursively removes non-facility nodes $v \notin F_{\mathrm{S}}$ of degree one.

Algorithm 2: Facility Exchange (Facilities F_{S})

```
$\hat{F} = F_{\mathrm{S}}$
forall $f \in F \setminus F_{\mathrm{S}}$ do
    $F' = F_{\mathrm{S}} \cup \{f\}$
    forall $g \in F_{\mathrm{S}}$ do
        $F' = F' \setminus \{g\}$
        derive Steiner tree $G_{\mathrm{T}}$ for $F'$
        derive customer assignment for $F', G_{\mathrm{T}}$
        if $o(F') > o(\hat{F})$ then
            $\hat{F} = F'$
        $F' = F' \cup \{g\}$
$F_{\mathrm{S}} = \hat{F}$
```

3.3. Assigning Clients to Facilities

Considering each facility $f \in F_{\mathrm{S}}$ as a knapsack with capacity $d_{\max}$ and each customer $k \in C$ as an item with weight d_k and profit p_k it is easy to see that assigning clients to a given set of open facilities $F_{\mathrm{S}} \in F$ is a Multiple Knapsack Problem with Assignment Restrictions (MKPAR) [5] and identical capacities (MKPAR-I).

MKPAR has been introduced by Dawande et al. [1] who considered the special case when the prizes of all items are equal to their costs. They showed that the problem is strongly NP hard even for the case of identical capacities and when the bipartite graph modelling the assignment restrictions is sparse. They described a greedy 1/3 approximation algorithm and two 1/2 approximation algorithms, one based on successively solving the single knapsack problem and a second one based on LP rounding.

In the following we present a heuristic as well as an exact method to assign clients to facilities. Furthermore, a mixed strategy utilizing both methods is explained in Section 3.4.

3.3.1. Heuristic Assignment

Algorithm 3 finds a heuristic solution to MKPAR-I by successively solving a series of single knapsack problems (KP). For each facility $f \in F_{\mathrm{S}}$ the corresponding KP is approached by a greedy algorithm which considers assignable customers $k \in C_f = \{i \in C \mid f \in F_i\}$ by their relative efficiency $\frac{p_k}{d_k}$ in decreasing order. Each client k is assigned to f if it has not been previously assigned to another facility and the resulting total demand of f does not exceed $d_{\max}$.

Since the set of clients C_f assignable to a facility f as well as efficiency values for clients can be precomputed, we assume that the entries of each set C_f is ordered by efficiency values in decreasing order.

As non-profitable facilities will be removed from a solution later on, the order in which to consider facilities is of

Algorithm 3: Assign (Facilities F_{S}, Tree G_{T})

```
root $G_{\mathrm{T}}$ at $r$
$s(f) = |\{g \in F_{\mathrm{S}} \mid g$ is successor of $f\}|, \forall f \in F_{\mathrm{S}}$
for $f \in F_{\mathrm{S}}$ in decreasing order w.r.t. $s(f)$ do
    for $k \in C_f$ in decreasing order w.r.t. $\frac{p_k}{d_k}$ do
        assign $k$ to $f$ if valid
```

great relevance. Let G_{T} be the Steiner tree rooted at r connecting the facilities in F_{S}. Algorithm 3 considers facilities by their number of succeeding facilities in G_{T} in decreasing order. A facility g is a successor of facility f iff the path from r to g includes f. This ordering prioritizes facilities that are "inside" the tree and typically closer to the root.

Rooting the tree and determining the number of successors for all facilities is implemented by a single depth first search. Therefore, Algorithm 3 has a computational complexity of $O(|V| + |F_{\mathrm{S}}| \log |F_{\mathrm{S}}| + |F_{\mathrm{S}}||C|)$.

3.3.2. Exact Assignment

To compute an optimal client assignment for a given set of facilities F_{S}, we solve the following integer linear programming (ILP) formulation from [1, 5] using a general purpose ILP solver.

$$\max \quad \sum_{k \in C} \sum_{f \in (F_k \cap F_{\mathrm{S}})} p_k x_{k,f} \tag{1}$$

$$\text{s.t.} \quad \sum_{f \in F_k \cap F_{\mathrm{S}}} x_{k,f} \leq 1 \qquad \forall k \in C \tag{2}$$

$$\sum_{k \in C | f \in F_k} d_k x_{k,f} \leq d_{\max} \qquad \forall f \in F_{\mathrm{S}} \tag{3}$$

$$x_{k,f} \in \{0, 1\} \qquad \forall k \in C, \forall f \in F_{\mathrm{S}} \tag{4}$$

Binary variables $x_{k,f}$ indicate whether client k is assigned to facility f ($x_{k,f} = 1$) or not ($x_{k,f} = 0$). Inequalities (2) ensure that each client is assigned to at most one facility, while inequalities (3) force the total demands resulting for each facility to be less than $d_{\max}$.

3.4. Combining Exact and Heuristic Assignment

As calculating an exact client assignment requires considerably more time than performing the heuristic client assignment, we combine both methods presented in Section 3.3 when searching the neighborhoods by using the heuristic assignment method as a relaxation of the exact approach. When searching a neighborhood N (i.e. FSN or FEN) we apply exact client assignment only to the most promising candidate solutions which are those that improve the best so far found solution within N with respect to the heuristic client assignment. As for each candidate set of facilities the objective value of the heuristic assignment is a lower bound to the one of the exact client assignment, we further use the heuristic solution as starting solution whenever calculating an exact client assignment. As a secondary effect this ensures that even if the ILP solver is terminated due to a time limit its solution is still never worse than the heuristic one.

3.5. Initial Solution

To create a feasible starting solution to PCConFL we declare all possible facilities as opened (i.e. $F_{\mathrm{S}} = F$), compute the corresponding Steiner Tree and assign clients to them using the heuristic assignment method. Afterwards, we remove all facilities with no assigned clients from F_{S} and remove non-profitable facilities using strong pruning as described in [8].

3.6. Variable Neighborhood Search

We use the general VNS framework [2] and FSN and FEN within an embedded variable neighborhood descent in this order. To analyze the impact of our methods for client assignment we compare variant VNS1 using the heuristic assignment method with variant VNS2 using the mixed strategy. Always applying the exact client assignment method turned out to be not competitive due to substantially longer running times.

3.6.1. Shaking

To escape from a local optimum we apply shaking based on FSN but swap $s = 1, \ldots, \lfloor \frac{|F|}{10} \rfloor$ facilities instead of only one. Afterwards, we perform strong pruning [8] to remove non-improving subtrees.

4. Test Instances

We combined instances for the PCSTP from Resende[1] with instances for one-dimensional Bin Packing[2] (BP). We selected the first node of a PCSTP instance as root and randomly chose $|F|$ other nodes as possible facilities. Each item in the BP instance is used as a client with a demand equal to the costs given, and the maximum demand for a single facility $d_{\max}$ is equal to the specified bin capacity.

Original edge costs from the PCSTP instance are multiplied by the estimated average number of clients that may be served by a single facility ($\frac{d_{\max}}{\overline{d}}$), and the prizes for clients are randomly chosen from the interval $[0.25 \cdot \overline{p}, 0.5 \cdot \overline{p}]$, with $\overline{p}$ being the average client prize in the PCSTP instance. Each client can be assigned to $|F_k| \in \{f_{\min}, \ldots, f_{\max}\}$ randomly chosen facilities.

[1]http://www.research.att.com/~mgcr/data/index.html

[2]http://www.wiwi.uni-jena.de/Entscheidung/binpp/index.htm

Table 1. Computational Results

				VNS1		VNS2	
$\|F\|$	$\|C\|$	$f_{\min}$	$f_{\max}$	$\overline{o(S)}$	std dev	$\overline{o(S)}$	std dev
				1235.6	0.52	**1308.0**	9.24
50	100	20	40	1537.0	0.00	**1594.1**	8.96
				1144.1	1.45	**1197.1**	10.02
				1854.0	0.00	**1921.0**	0.00
50	200	20	40	1789.0	0.00	**1901.0**	0.00
				1457.0	0.00	**1560.6**	1.90
				1605.1	7.30	**1621.7**	13.49
100	100	20	50	1757.6	5.39	**1773.3**	6.80
				1447.8	7.10	**1459.7**	7.37
				2852.7	2.59	**2932.0**	6.82
100	200	20	50	3378.1	3.57	**3417.7**	11.05
				2675.6	2.98	**2722.5**	10.4
				1510.3	11.98	**1544.2**	19.04
200	100	20	60	1776.0	5.73	**1781.6**	8.83
				1439.6	11.64	**1452.6**	18.86
				3016.2	13.97	3015.3	15.2
200	200	20	60	**3589.3**	17.22	3577.2	25.65
				2543.7	22.63	**2557.0**	9.59

5. Computational Results

We tested our algorithms on instances with $|F| \in \{50, 100\}$ possible facilities, $|V| = 500$ nodes and $|E| = 1000$ edges as well as on instances with $|F| = 200$, $|V| = 1000$ and $|E| = 2000$.

Table 1 reports instance characteristics as well as average solution values and corresponding standard deviations for three instances of each type. Standard deviations have been computed using 10 runs for each instance and we used a time limit of 300s for instances with $|F| \in \{50, 100\}$ and 450s in case $|F| = 200$ as termination criterion. All tests have been performed on a single core of a Intel Xeon 5150 with 2.66GHz and 8GB RAM; ILOG CPLEX 11.0 has been used to solve the ILPs in VNS2. Since proving the optimality of an exact client assignment in VNS2 frequently needs considerably more time than finding the optimum we enforce a time limit of 2s for solving each ILP.

Results indicate that VNS2 is able to improve solution quality for all but two instances. However, the average improvement of VNS2 over VNS1 tends to get smaller with increasing number of possible facilities. One reason for this behavior is the larger number of possibilities for selecting facilities leading to more iterations required in order to find a high quality solution, so that the higher speed of VNS1 is particularly advantageous. Additionally, with an increasing number of possible facilities, solving the ILP in VNS2 will need disproportionately more time or will be aborted due to the time limit with a larger remaining gap. Nevertheless, VNS2 is clearly able to compensate its higher computational effort for almost all instances of our instances.

6. Conclusions and Future Work

This article introduced a prize collecting variant of the connected facility location problem with capacity constraints. Based on a characterization of the problem by its subproblems, a general variable neighborhood search utilizing two different types of neighborhood structures and second and third level procedures for completing solutions by connecting chosen facilities and assigning clients has been presented. Results show that the variant combining the greedy heuristic with the exact ILP-based method for the assignment of clients has clear advantages over the pure heuristic approach.

In future work we want to investigate a self-adaptive strategy with respect to the time limit for solving the exact part and apply similar ideas to other metaheuristics. Additionally, it seems important to develop an exact method for the problem in order to compare the results of our approach with optimal ones. Finally, we want to accomplish a more detailed computational study as well as test our approach on real-world instances.

References

[1] M. Dawande, J. Kalagnanam, P. Keskinocak, F. Salman, and R. Ravi. Approximation algorithms for the multiple knapsack problem with assignment restrictions. *Journal of Combinatorial Optimization*, 4(2):171–186, 2000.

[2] P. Hansen and N. Mladenovic. An introduction to variable neighborhood search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-heuristics, Advances and trends in local search paradigms for optimization*, pages 433–458. Kluwer Academic Publishers, 1999.

[3] M. K. Hasan, H. Jung, and K.-Y. Chwa. Approximation algorithms for connected facility location problems. *Journal of Combinatorial Optimization*, 2008. accepted.

[4] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 613–623, Washington DC, 2000. IEEE Computer Society.

[5] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Heidelberg, 2004.

[6] I. Ljubić. A hybrid VNS for connected facility location. In T. Bartz-Beielstein et al., editors, *Hybrid Metaheuristics, 4th International Workshop, HM 2007*, volume 4771 of *LNCS*, pages 157–169. Springer, 2007.

[7] K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters*, 27(3):125–128, 1988.

[8] M. Minkoff. The prize collecting Steiner tree problem. Master's thesis, Massachusetts Institute of Technology, 2000.

[9] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.