Approximation Algorithms for Facility Location

Jens Vygen

University of Bonn

Outline

Facility Location: Applications

- manufacturing plants
- storage facilities, depots
- warehouses, retail stores
- libraries, fire stations, hospitals
- servers in the internet
- base stations for wireless services
- buffers distributing signals on a chip
- ...

Goal: Optimum service for clients at minimum cost

Common features of facility location problems

- Two sets: clients and potential facilities
  - Each client must be served.
  - A potential facility can be opened or not.
  - Clients can only be served by open facilities.
- Two cost components: facility cost and service cost.
  - Opening a facility involves a certain cost.
  - Serving a client from a facility involves a certain cost.
- The total cost is to be minimized.

Illustration: facility location instance

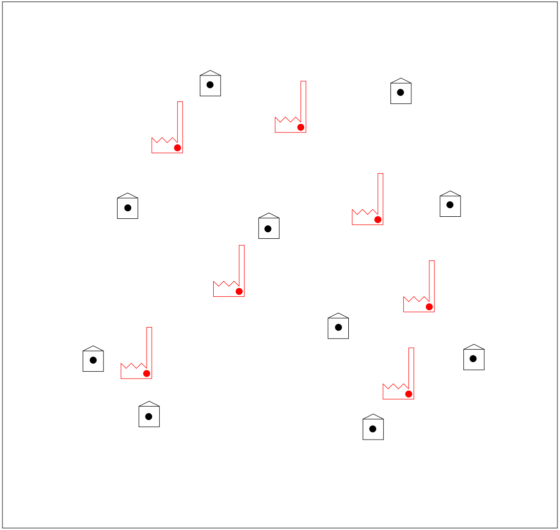


Illustration: choosing a set of open facilities

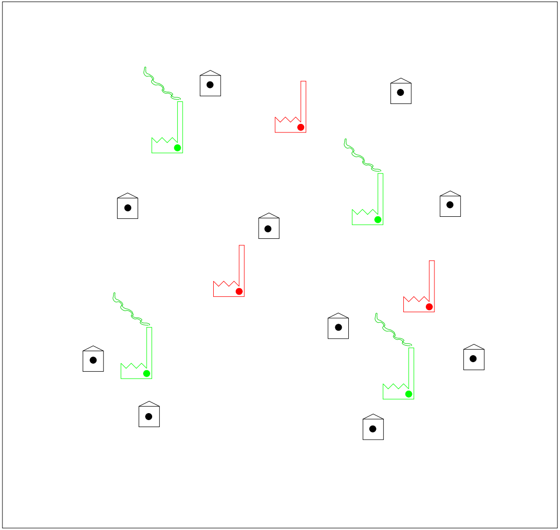
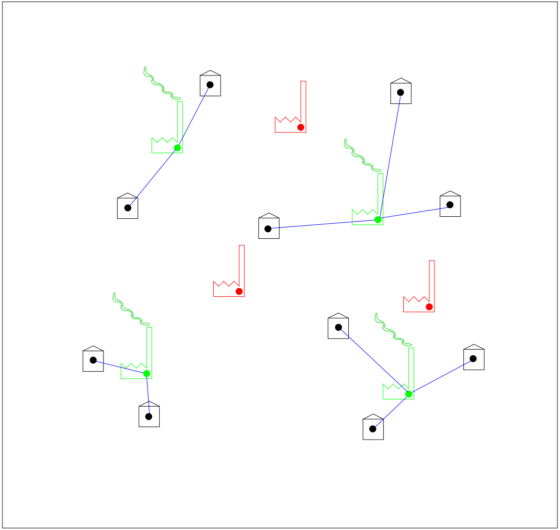


Illustration: connect clients to open facilities



But there are many variants

- Can a client's demand be satisfied by more than one facility?
- Are there constraints on the total demand, or total service cost, that a facility can handle?
- Do the service costs satisfy the triangle inequality?
- Are there finitely or infinitely many potential facilities?
- Do the facility costs depend on the total demand served?
- Is it allowed to serve only a subset of clients, and pay for those that are not served?
- Is there a bound on the number of facilities that we can open?
- Does the total service cost of a facility depend on the sum of the distances to its clients, or the length of a shortest tour, or the length of an optimal Steiner tree?
- Are we interested in the sum of all service costs, or rather in the maximum service cost?
- Do we need to serve facilities by second-stage facilities (etc.)?

Example 1: Fermat-Weber Problem

The most prominent example for continuous facility location

Locating a single facility in $\mathbb{R}^n$: Given $a_1, \ldots, a_m \in \mathbb{R}^n$ and weights $w_1, \ldots, w_m \in \mathbb{R}_+$, find $p \in \mathbb{R}^n$ minimizing

$$\sum_{i=1}^{m} w_i ||p - a_i||.$$

- For ℓ_1-norm solvable in linear time (Blum et al. 1973)
- ℓ_2-norm, $n = 2$, $m = 3$: Simple geometric solution (Fermat, Torricelli, Cavalieri, Simpson, Heinen)
- For ℓ_2-norm: construction by ruler and compasses impossible (Bajaj 1988)
- Approximate solution for ℓ_2-norm: Weiszfeld's algorithm (Weiszfeld 1937, Kuhn 1973, Vardi and Zhang 2001, Rautenbach et al. 2004)

Example 2: Uncapacitated Facility Location (UFL)

The most prominent example for discrete facility location

Instance:

- a finite set $\mathcal{D}$ of clients;
- a finite set $\mathcal{F}$ of potential facilities;
- a fixed cost $f_i \in \mathbb{R}_+$ for opening each facility $i \in \mathcal{F}$;
- a service cost $c_{ij} \in \mathbb{R}_+$ for each $i \in \mathcal{F}$ and $j \in \mathcal{D}$.

We look for:

- a subset S of facilities (called *open*) and
- an assignment $\sigma : \mathcal{D} \to S$ of clients to open facilities,
- such that the sum of facility costs and service costs

$$\sum_{i \in S} f_i + \sum_{j \in \mathcal{D}} c_{\sigma(j)j}$$

is minimum.

More examples discussed later

- Capacitated Facility Location
- Universal Facility Location
- Facility Location and Network Design with Service Capacities

These are more general and more realistic in many applications.

Approximation Algorithms: Definition

Let f be a function assigning a real number to each instance.
An f-**approximation algorithm** is an algorithm for which a polynomial p exists such that for each instance I:

- the algorithm terminates after at most $p(size(I))$ steps,
- the algorithm computes a feasible solution, and
- the cost of this solution is at most $f(I)$ times the optimum cost of instance I.

f is called the **approximation ratio** or **performance guarantee**.
If f is a constant, we have a **(constant-factor) approximation algorithm**.

Uncapacitated Facility Location is as hard as Set Covering

Set Covering: Given a finite set U, a family $\mathcal{S}$ of subsets of U with $\bigcup_{S \in \mathcal{S}} S = U$, and weights $w : \mathcal{S} \to \mathbb{R}_+$, find a set $\mathcal{R} \subseteq \mathcal{S}$ with $\bigcup_{R \in \mathcal{R}} R = U$ with minimum total weight $\sum_{R \in \mathcal{R}} w(R)$.

- No $o(\log |U|)$-approximation algorithm exists unless $P = NP$. (Raz, Safra 1997)
- Greedy algorithm has performance ratio $1 + \ln |U|$. (Chvátal 1979)
- Set Covering is a special case of Uncapacitated Facility Location: define $\mathcal{D} := U$, $\mathcal{F} := \mathcal{S}$, $f_S = w(S)$ for $S \in \mathcal{S}$, $c_{Sj} := 0$ for $j \in S \in \mathcal{S}$ and $c_{Sj} := \infty$ for $j \in U \setminus S$.
- Conversely, the greedy algorithm for Set Covering can be applied to Uncapacitated Facility Location: Set $U := \mathcal{D}$, $\mathcal{S} = \mathcal{F} \times 2^{\mathcal{D}}$, and $w(i, D) := f_i + \sum_{j \in D} c_{ij}$. (Hochbaum 1982)

A natural assumption: metric service costs

Therefore we assume henceforth **metric service costs**:

$$c_{ij} \geq 0$$

and

$$c_{ij} + c_{i'j} + c_{i'j'} \geq c_{ij'}$$

for all $i, i' \in \mathcal{F}$ and $j, j' \in \mathcal{D}$.

Equivalently, we assume c to be a (semi)metric on $\mathcal{D} \cup \mathcal{F}$.

Motivation:

- The general problem is as hard as SET COVERING.
- In many practical problems service costs are proportional to geometric distances, or to travel times, and hence are metric.

But: Greedy algorithm has performance guarantee $\Omega(\log n / \log\log n)$ even for metric instances. (Jain et al. 2003)

Integer Linear Programming Formulation

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{D}} c_{ij}x_{ij} \\
\text{subject to} & \\
& \begin{array}{rcll}
x_{ij} & \leq & y_i & (i\in\mathcal{F}, j\in\mathcal{D}) \\
\displaystyle\sum_{i\in\mathcal{F}} x_{ij} & = & 1 & (j\in\mathcal{D}) \\
x_{ij} & \in & \{0,1\} & (i\in\mathcal{F}, j\in\mathcal{D}) \\
y_i & \in & \{0,1\} & (i\in\mathcal{F})
\end{array}
\end{array}
$$

(Balinski 1965)

Linear Programming Relaxation

$$\text{minimize} \quad \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij}$$

subject to

$$\begin{array}{rcll} x_{ij} & \leq & y_i & (i \in \mathcal{F}, j \in \mathcal{D}) \\ \sum_{i \in \mathcal{F}} x_{ij} & = & 1 & (j \in \mathcal{D}) \\ x_{ij} & \geq & 0 & (i \in \mathcal{F}, j \in \mathcal{D}) \\ y_i & \geq & 0 & (i \in \mathcal{F}) \end{array}$$

The Dual LP

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{j \in \mathcal{D}} v_j \\ \text{subject to} & \end{array}$$

$$\begin{array}{rcll} v_j - w_{ij} & \leq & c_{ij} & (i \in \mathcal{F}, j \in \mathcal{D}) \\ \displaystyle\sum_{j \in \mathcal{D}} w_{ij} & \leq & f_i & (i \in \mathcal{F}) \\ w_{ij} & \geq & 0 & (i \in \mathcal{F}, j \in \mathcal{D}) \end{array}$$

First Approximation Algorithm: LP Rounding

- Compute an optimum solutions (x^*, y^*) and (v^*, w^*) to the primal and dual LP.
- By complementary slackness, $x^*_{ij} > 0$ implies $v^*_j - w^*_{ij} = c_{ij}$, and thus $c_{ij} \leq v^*_j$.
- Let G be the bipartite graph with vertex set $\mathcal{F} \cup \mathcal{D}$ containing an edge $\{i, j\}$ iff $x^*_{ij} > 0$.
- Assign clients to clusters iteratively as follows.
  - In iteration k, let j_k be a client $j \in \mathcal{D}$ not assigned yet and with v^*_j smallest.
  - Create a new cluster containing j_k and those vertices of G that have distance 2 from j_k and are not assigned yet.
  - Continue until all clients are assigned to clusters.
- For each cluster k we choose a neighbour i_k of j_k with f_{i_k} minimum, open i_k, and assign all clients in this cluster to i_k.

Analysis of the LP Rounding Approximation Algorithm

- The service cost for client j in cluster k is at most

$$c_{i_k j} \;\leq\; c_{ij} + c_{ij_k} + c_{i_k j_k} \;\leq\; v_j^* + 2v_{j_k}^* \;\leq\; 3v_j^*,$$

  where i is a common neighbour of j and j_k.
- The facility cost f_{i_k} can be bounded by

$$f_{i_k} \;\leq\; \sum_{i \in \mathcal{F}} x_{ij_k}^* f_i \;=\; \sum_{i \in \mathcal{F}:\{i,j_k\} \in E(G)} x_{ij_k}^* f_i \;\leq\; \sum_{i \in \mathcal{F}:\{i,j_k\} \in E(G)} y_i^* f_i.$$

  As j_k and $j_{k'}$ cannot have a common neighbour for $k \neq k'$, the total facility cost is at most $\sum_{i \in \mathcal{F}} y_i^* f_i$.
- The total cost is at most

$$3 \sum_{j \in \mathcal{D}} v_j^* + \sum_{i \in \mathcal{F}} y_i^* f_i,$$

  which is at most four times the LP value. Hence we get:

Theorem

This is a 4-approximation algorithm for metric UFL.

(Shmoys, Tardos and Aardal 1997)

Better approximation ratios for metric UFL

technique	ratio	RT	authors	year
LP-Rounding	3.16	−	Shmoys, Tardos, Aardal	1997
LP-Rounding+Greedy	2.41	−	Guha, Khuller	1998
LP-Rounding	1.74	−	Chudak	1998
Local Search	5.01	○	Korupolu, Plaxton, Rajaraman	1998
Primal-Dual	3.00	+	Jain, Vazirani	1999
Primal-Dual+Greedy	1.86	+	Charikar, Guha	1999
LP-Rounding+Primal-Dual+Greedy	1.73	−	Charikar, Guha	1999
Local Search	2.42	○	Arya et al.	2001
Primal-Dual	1.61	+	Jain, Mahdian, Saberi	2002
LP-Rounding	1.59	−	Sviridenko	2002
Primal-Dual+Greedy	1.52	+	Mahdian, Ye, Zhang	2002

RT : running time; − : slow; ○ : medium; + : fast

Primal-Dual Algorithm by Jain, Mahdian and Saberi (2002)

Start with $U := \mathcal{D}$ and time $t = 0$. Increase t, maintaining $v_j = t$ for all $j \in U$. Consider the following events:

- $v_j = c_{ij}$, where $j \in U$ and i is not open. Then start to increase w_{ij} at the same rate, in order to maintain $v_j - w_{ij} = c_{ij}$.
- $\sum_{j \in \mathcal{D}} w_{ij} = f_i$. Then open i. For all $j \in \mathcal{D}$ with $w_{ij} > 0$: freeze v_j and set $w_{i'j} := \max\{0, c_{ij} - c_{i'j}\}$ for all $i' \in \mathcal{F}$, and remove j from U.
- $v_j = c_{ij}$, where $j \in U$ and i is open. Then freeze v_j and set $w_{i'j} := \max\{0, c_{ij} - c_{i'j}\}$ for all $i' \in \mathcal{F}$, and remove j from U.

Improvement by Mahdian, Ye and Zhang (2002)

- Multiply all facility costs by 1.504.
- Apply the Jain-Mahdian-Saberi algorithm.
- Now consider the original facility costs.
- Apply greedy augmentation (Charikar, Guha 1999):
  Let g_i be the service cost saving induced by adding facility i.
  Iteratively pick an element $i \in \mathcal{F}$ maximizing $\frac{g_i}{f_i}$ as long as this ratio is greater than 1.

Theorem

This is a 1.52-approximation algorithm for metric UFL.

Lower bound on approximation ratios

Theorem

There is no 1.463-factor approximation algorithm for metric UFL unless $P = NP$.

(Sviridenko [unpublished], based on Guha and Khuller [1999] and Feige [1998])

Local Search as a general heuristic

Basic Framework:

- Define a neighbourhood graph on the feasible solutions.
- Start with any feasible solution x.
- If there is a neighbour y of x that is (significantly) better, set $x := y$ and iterate.

Features:

- Quite successful for many practical (hard) problems
- Many variants of local search heuristics
- Typically no guarantees of running time and performance ratio.

Local Search in Combinatorial Optimization

Example: TSP

- Even simple 2-opt typically yields good solutions. Variants (chained Lin-Kernighan) with empirically less than 1% error
- Worst-case running time of k-opt is exponential for all k.
- Performance ratio $\Omega(n^{\frac{1}{2k}})$.

(Applegate et al. 2003, Chandra, Karloff, Tovey 1999)

Example: Facility Location

- Probably the first nontrivial problem where local search led to constant-factor approximation algorithms.
  (Korupolo, Plaxton and Rajamaran 2000, Arya et al. 2004)
- But: for metric UFL worse in theory (maybe also in practice)
- The only known technique to obtain a constant-factor approximation for Capacitated Facility Location.

Capacitated Facility Location (CFL)

Instance:

- finite sets $\mathcal{D}$ (clients) and $\mathcal{F}$ (potential facilities);
- metric service costs $c_{ij} \in \mathbb{R}_+$ for $i \in \mathcal{F}$ and $j \in \mathcal{D}$;
- an opening cost $f_i \in \mathbb{R}_+$ for each facility $i \in \mathcal{F}$;
- a capacity $u_i \in \mathbb{R}_+$ for each facility $i \in \mathcal{F}$;
- a demand d_j for each client $j \in \mathcal{D}$.

We look for:

- a subset S of facilities (called *open*) and
- an assignment $x : S \times \mathcal{D} \to \mathbb{R}_+$ with $\sum_{i \in S} x_{ij} = d_j$ for $j \in \mathcal{D}$ and $\sum_{j \in \mathcal{D}} x_{ij} \leq u_i$ for $i \in S$
- such that the sum of facility costs and service costs

$$\sum_{i \in S} \left(f_i + \sum_{j \in \mathcal{D}} c_{ij} x_{ij} \right)$$

is minimum.

Splittable or Unsplittable Demands

Assume that facilities with given capacities are open.
Task: assign the clients to these facilities, respecting capacity constraints.

- Splittable (or uniform) demand:
  Hitchcock transportation problem.
- Unsplittable non-uniform demand:
  Generalizes bin packing.

Consequence: CFL with unsplittable demands has no approximation algorithm. It is strongly NP-hard to distinguish between instances with optimum cost 0 and ∞.

Hence consider splittable demands only.

Universal Facility Location (UniFL)

Instance:

- finite sets $\mathcal{D}$ (clients) and $\mathcal{F}$ (potential facilities);
- metric service costs, i.e. a metric c on $\mathcal{D} \cup \mathcal{F}$;
- a demand $d_j \geq 0$ for each $j \in \mathcal{D}$;
- for each $i \in \mathcal{F}$ a cost function $f_i : \mathbb{R}_+ \to \mathbb{R}_+ \cup \{\infty\}$, left-continuous and non-decreasing.

We look for:

- a function $x : \mathcal{F} \times \mathcal{D} \to \mathbb{R}_+$ with $\sum_{i \in \mathcal{F}} x_{ij} = d_j$ for all $j \in \mathcal{D}$

(a *feasible solution*), such that $c(x) := c_F(x) + c_S(x)$ is minimum, where

$$c_F(x) := \sum_{i \in \mathcal{F}} f_i\left(\sum_{j \in \mathcal{D}} x_{ij}\right) \qquad \text{and} \qquad c_S(x) := \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij}.$$

UniFL: Facility cost function given by an oracle

$f_i(z)$: cost to install capacity z at facility i.

Given by an oracle that, for each $i \in \mathcal{F}$, $u, c \in \mathbb{R}_+$ and $t \in \mathbb{R}$, computes $f_i(u)$ and

$$\max\{\delta \in \mathbb{R} : u + \delta \geq 0,\ f_i(u+\delta) - f_i(u) + c|\delta| \leq t\}.$$

Proposition

There always exists an optimum solution.

(Mahdian and Pál 2003)

UniFL: important special cases

- Uncapacitated Facility Location:
  $d_j = 1$ ($j \in \mathcal{D}$), and $f_i(0) = 0$ and $f_i(z) = t_i$ for some $t_i \in \mathbb{R}_+$ and all $z > 0$ ($i \in \mathcal{F}$).
- Capacitated Facility Location:
  $f_i(0) = 0$, $f_i(z) = t_i$ for $0 < z \le u_i$ and $f_i(z) = \infty$ for $z > u_i$, where $u_i, t_i \in \mathbb{R}_+$ ($i \in \mathcal{F}$).
- Soft-Capacitated Facility Location:
  $d_j = 1$ ($j \in \mathcal{D}$), and $f_i(z) = \lceil \frac{z}{u_i} \rceil t_i$ for some $u_i \in \mathbb{N}$, $t_i \in \mathbb{R}_+$ and all $z \ge 0$ ($i \in \mathcal{F}$).

Simple local search operations

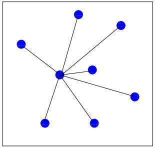
- ADD: open a facility (CFL); add capacity to a facility (UniFL).
- DROP: close a facility (CFL).
- SWAP: open one facility, close another one (CFL).

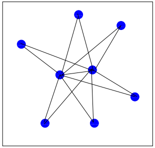
Even for CFL with non-uniform demands, these operations do not suffice:
When closing one facility, it may be necessary to open many other ones (and re-assign the demand along the edges of a star).

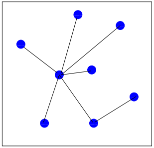
Previous approximation algorithms for CFL and UniFL

Kuehn, Hamburger 1963	add,drop,swap	CFL	—	
Korupolu, Plaxton, Rajamaran 1998	add,drop,swap	CFL	8.001	uniform capacities
Chudak, Williamson 1999	add,drop,swap	CFL	5.829	uniform capacities
Pál, Tardos, Wexler 2001	add,star	CFL	8.532	
Mahdian, Pál 2003	add,star	UniFL	7.873	
Zhang, Chen, Ye 2004	add,double-star	CFL	5.829	
Garg, Khandekar, Pandit 2005	add,double-star	UniFL	5.829	not polynomial!
Vygen 2005	add,comet	UniFL	6.702	

All based on local search.


star


double star


comet

Add Operation for UniFL

Let $t \in \mathcal{D}$ and $\delta > 0$. Replace current solution x by an optimum solution y of the transportation problem

$$\min\left\{ c_S(y) \;\middle|\; \begin{array}{l} y : \mathcal{F} \times \mathcal{D} \to \mathbb{R}_+,\ \sum_{i \in \mathcal{F}} y_{ij} = d_j \,(j \in \mathcal{D}), \\ \sum_{j \in \mathcal{D}} y_{ij} \le \sum_{j \in \mathcal{D}} x_{ij} \,(i \in \mathcal{F} \setminus \{t\}),\ \sum_{j \in \mathcal{D}} y_{tj} \le \sum_{j \in \mathcal{D}} x_{tj} + \delta \end{array} \right\}.$$

We denote by

$$c^x(t, \delta) := c_S(y) - c_S(x) + f_t\left(\sum_{j \in \mathcal{D}} x_{tj} + \delta\right) - f_t\left(\sum_{j \in \mathcal{D}} x_{tj}\right)$$

the estimated cost (which is at least $c(y) - c(x)$).

How to find a profitable ADD operation

Lemma

Let $\epsilon > 0$ and $t \in \mathcal{F}$. Let x be a feasible solution. Then there is an algorithm with running time $O(|V|^3 \log |V| \epsilon^{-1})$ that

- *finds a $\delta \in \mathbb{R}_+$ with $c^x(t, \delta) \leq -\epsilon c(x)$*
- *or decides that no $\delta \in \mathbb{R}_+$ exists for which $c^x(t, \delta) \leq -2\epsilon c(x)$.*

(Mahdian, Pál 2003)

Pivot Operation

Let x be a feasible solution. Let A be a graph with $V(A) = \mathcal{F}$ and

$$\delta \in \Delta_A^x := \left\{ \delta \in \mathbb{R}^{\mathcal{F}} \,\middle|\, \sum_{j \in \mathcal{D}} x_{ij} + \delta_i \geq 0 \text{ for all } i \in \mathcal{F}, \sum_{i \in \mathcal{F}} \delta_i = 0 \right\}.$$

Then we consider the operation $\textsc{Pivot}(A, \delta)$, which means:

- Compute a minimum-cost (w.r.t. c) uncapacitated δ-flow in (A, c).
- W.l.o.g., the edges carrying flow form a forest.
- Scan these edges in topological order, reassigning clients according to flow values.
- This increases the cost of the solution by at most the cost of the flow plus

$$\sum_{i \in \mathcal{F}} f_i\left(\sum_{j \in \mathcal{D}} x_{ij} + \delta_i\right) - f_i\left(\sum_{j \in \mathcal{D}} x_{ij}\right).$$

How to find a profitable PIVOT operation

But: how to choose δ?

- δ cannot be chosen almost optimally for the complete graph (unless $P = NP$).
- We show how to choose δ almost optimally if A is a forest.

Restrict attention to Pivot on arborescences

Let A be an arborescence with $V(A) = \mathcal{F}$. Let x be a feasible solution.
For $\delta \in \Delta_A^x$ define

$$c_{A,i}^x(\delta) := f_i\left(\sum_{j\in\mathcal{D}} x_{ij} + \delta_i\right) - f_i\left(\sum_{j\in\mathcal{D}} x_{ij}\right) + \left|\sum_{j\in A_i^+} \delta_j\right| c_{ip(i)}$$

for $i \in \mathcal{F}$ and

$$c^x(A,\delta) := \sum_{i\in\mathcal{F}} c_{A,i}^x(\delta).$$

Here A_i^+ denotes the set of vertices reachable from i in A, and $p(i)$ is the predecessor of i.

How to find a profitable Pivot for an arborescence

Lemma

Let $\epsilon > 0$. There is an algorithm with running time $O(|\mathcal{F}|^4\epsilon^{-3})$ that

- *finds a $\delta \in \Delta_A^x$ with $c^x(A, \delta) \leq -\epsilon c(x)$*
- *or decides that no $\delta \in \Delta_A^x$ exists for which $c^x(A, \delta) \leq -2\epsilon c(x)$.*

(Vygen 2005)

Bounding the cost of a local optimum

Let $0 < \epsilon < 1$. Let x, x^* be feasible solutions to a given instance.

Lemma

If $c^x(t, \delta) \geq -\frac{\epsilon}{|\mathcal{F}|} c(x)$ for all $t \in \mathcal{F}$ and $\delta \in \mathbb{R}_+$, then

$$c_S(x) \leq c_F(x^*) + c_S(x^*) + \epsilon c(x).$$

(Pál, Tardos and Wexler 2001)

Lemma

If $c^x(A, \delta) \geq -\frac{\epsilon}{|\mathcal{F}|} c(x)$ for all stars and comets A and $\delta \in \Delta^x_A$, then

$$c_F(x) \leq 4c_F(x^*) + 2c_S(x^*) + 2c_S(x) + \epsilon c(x).$$

(Vygen 2005)

The total cost of a local optimum

These two lemmata imply:

Theorem

If $c^x(t,\delta) > -\frac{\epsilon}{8|\mathcal{F}|}c(x)$ *for* $t \in \mathcal{F}$ *and* $\delta \in \mathbb{R}_+$ *and* $c^x(A,\delta) > -\frac{\epsilon}{8|\mathcal{F}|}c(x)$ *for all stars and comets* A *and* $\delta \in \Delta_A^x$*, then*

$$c(x) \leq (1+\epsilon)(7c_F(x^*) + 5c_S(x^*)).$$

By scaling facility costs by $\frac{\sqrt{41}-5}{2}$ we get a polynomial-time $(\frac{\sqrt{41}+7}{2}+\epsilon)$-approximation algorithm for UniFL.

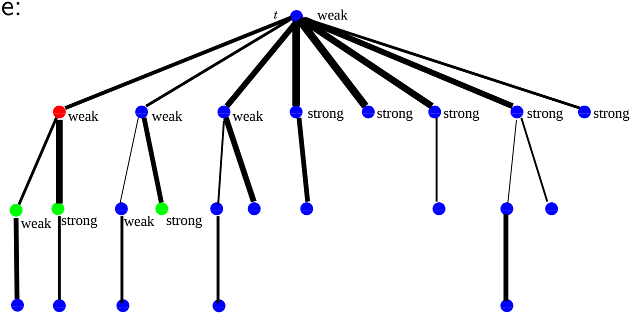
How to bound the facility cost

Let x be the current solution and x^* be an optimum solution.
Let $b(i) := \sum_{j \in \mathcal{D}} (x_{ij} - x^*_{ij})$ $(i \in \mathcal{F})$.
Let y be an optimum transshipment from $S := \{i \in \mathcal{F} : b(i) > 0\}$ to $T := \{i \in \mathcal{F} : b(i) < 0\}$.
W.l.o.g., the edges where y is positive form a forest F.
The cost of y is at most $c_S(x^*) + c_S(x)$.

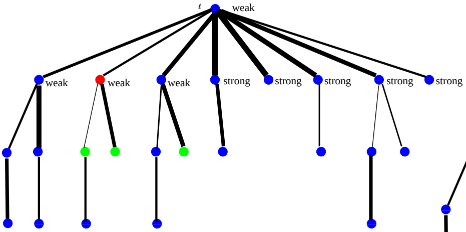
Using F and y, we will define a set of pivot operations on stars and comets, whose total estimated cost is at most
$4c_F(x^*) - c_F(x) + 2c_S(x^*) + 2c_S(x)$.
An operation (A, δ) **closes** $s \in S$ if $\delta_s = -b(s) < 0$, and it **opens** $t \in T$ if $0 < \delta_t \leq -b(t)$.
Over all operations to be defined, we will close each $s \in S$ once, open each $t \in T$ at most four times, and use an estimated routing cost at most twice the cost of y.

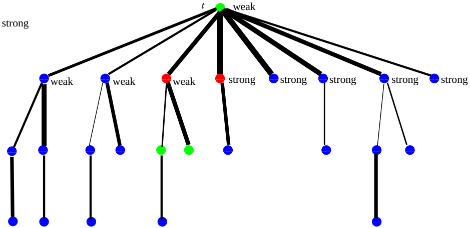
How to define the operations (1)

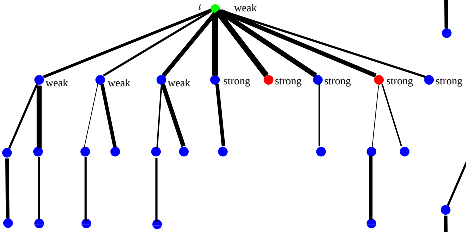
Orient F as a set of arborescences rooted at elements of T.
Call a vertex **weak** if there is more flow on downward than on upward incident arcs, otherwise **strong**. Let $t \in T$.
Open t up to twice if t is strong and up to three times if t is weak.
For each child s of t: Close s once, and open each child of s at most once (if weak) or twice (if strong).
Example:

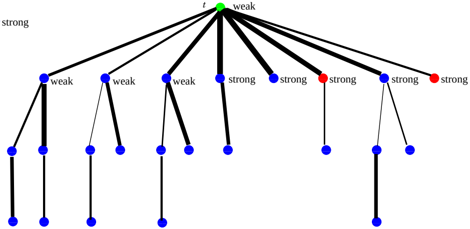


How to define the operations (2)

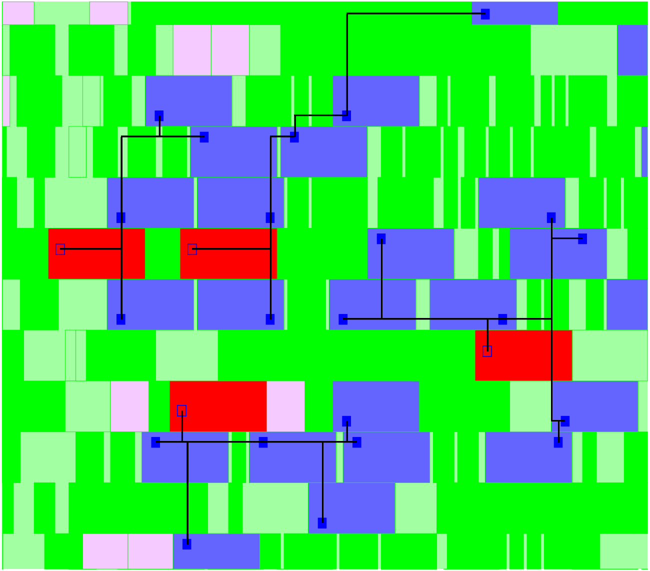








VLSI Design: Distributing a signal to several terminals



blue: terminals red: facilities

Problem Statement

Instance:

- metric space (V, c),
- finite set $\mathcal{D} \subseteq V$ (terminals/clients),
- demands $d : \mathcal{D} \to \mathbb{R}_+$,
- facility opening cost $f \in \mathbb{R}_+$,
- capacity $u \in \mathbb{R}_+$.

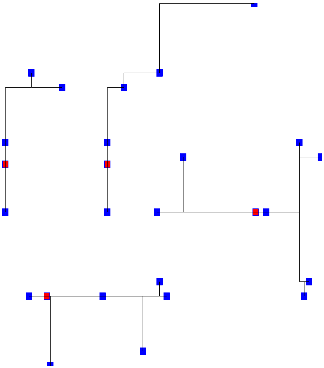
Find a partition $\mathcal{D} = D_1 \dot\cup \cdots \dot\cup D_k$ and Steiner trees T_i for D_i $(i = 1, \ldots, k)$ with

$$c(E(T_i)) + d(D_i) \leq u$$

for $i = 1, \ldots, k$ such that

$$\sum_{i=1}^{k} c(E(T_i)) + kf$$

is minimum.



Complexity Results

(All the following results are by Maßberg and Vygen 2005)

Proposition

- *There is no $(1.5 - \epsilon)$-approximation algorithm (for any $\epsilon > 0$) unless $P = NP$.*
- *There is no $(2 - \epsilon)$-approximation algorithm (for any $\epsilon > 0$) for any class of metrics where the Steiner tree problem cannot be solved exactly in polynomial time.*
- *There is a 2-approximation algorithm for geometric instances (similar to Arora's approximation scheme for the TSP). However, this is not practically efficient.*

Lower bound: spanning forests

Let F_1 be a minimum spanning tree for $(\mathcal{D}, c)$.
Let $e_1, \ldots, e_{n-1}$ be the edges of F_1 so that $c(e_1) \geq \ldots \geq c(e_{n-1})$.
Set $F_k := F_{k-1} \setminus \{e_{k-1}\}$ for $k = 2, \ldots, n$.

Lemma

F_k is a minimum weight spanning forest in $(\mathcal{D}, c)$ with exactly k components.

Proof.

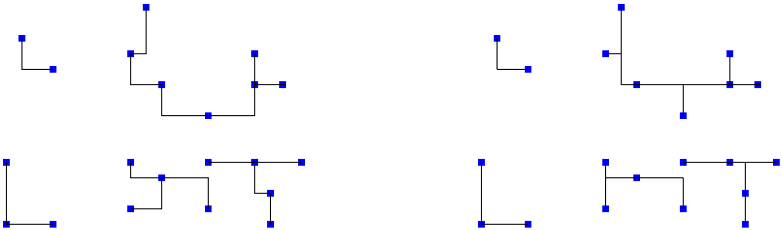
By induction on k. Trivial for $k = 1$. Let $k > 1$.
Let F^* be a minimum weight k-spanning forest.
Let $e \in F_{k-1}$ such that $F^* \cup \{e\}$ is a forest. Then

$$c(F_k) + c(e_{k-1}) = c(F_{k-1}) \leq c(F^*) + c(e) \leq c(F^*) + c(e_{k-1}).$$

$\square$

Lower bound: Steiner forests

A k-Steiner forest is a forest F with $\mathcal{D} \subseteq V(F)$ and exactly k components.



Lemma

$\frac{1}{\alpha}c(F_k)$ is a lower bound for the cost of a minimum weight k-Steiner forest, where α is the Steiner ratio.

Lower bound: number of facilities

Let t' be the smallest integer such that

$$\frac{1}{\alpha}c(F_{t'}) + d(\mathcal{D}) \leq t' \cdot u$$

Lemma

t' is a lower bound for the number of facilities of any solution.

Let t'' be an integer in $\{t', \ldots, n\}$ minimizing

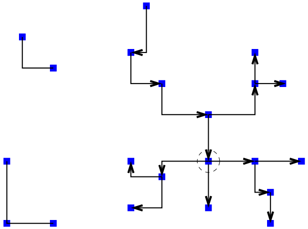
$$\frac{1}{\alpha}c(F_{t''}) + t'' \cdot f.$$

Theorem

$\frac{1}{\alpha}c(F_{t''}) + t'' \cdot f$ is a lower bound for the cost of an optimal solution.

Algorithm A

1. Compute a minimum spanning tree on $(\mathcal{D}, c)$.
2. Compute t'' and spanning forest $F_{t''}$ as above.
3. Split up overloaded components by a bin packing approach.



It can be guaranteed that for each new component at least $\frac{u}{2}$ of load will be removed from the initial forest.

Analysis of Algorithm A

Recall: $\frac{1}{\alpha}c(F_{t''}) + t'' \cdot f$ is a lower bound for the optimum.

We set $L_r := \frac{1}{\alpha}c(F_{t''})$ and $L_f := t'' \cdot f$.

Observe: $L_r + d(\mathcal{D}) \leq \frac{u}{f}L_f$.

The cost of the final solution is at most

$$c(F_{t''}) + t''f + \frac{2}{u}\Big(c(F_{t''}) + d(\mathcal{D})\Big)f$$

$$= \alpha L_r + L_f + \frac{2f}{u}\left(\alpha L_r + d(\mathcal{D})\right)$$
$$\leq \alpha L_r + L_f + 2\alpha L_f$$

Theorem

Algorithm A is a $(2\alpha + 1)$*-approximation algorithm.*

Algorithm B

Define metric c' by $c'(v, w) := \min\{c(v, w), \frac{uf}{u+2f}\}$.

1. Compute a Steiner tree F for $\mathcal{D}$ in (V, c') with some β-approximation algorithm.
2. Remove all edges e of F with $c(e) \geq \frac{uf}{u+2f}$.
3. Split up overloaded components of the remaining forest as in algorithm A.

Theorem

Algorithm B has perfomance ratio 3β*.*

Using the Robins-Zelikovsky Steiner tree approximation algorithm we get a 4.648-approximation algorithm.

With a more careful analysis of the Robins-Zelikovsky algorithm we can get a 4.099-approximation algorithm in $O(n^{2^{10000}})$ time.

Algorithm C

Define metric c'' by $c''(v,w) := \min\{c(v,w), \frac{uf}{u+f}\}$

1. Compute a tour F for $\mathcal{D}$ in (V, c'') with some γ-approximation algorithm.
2. Remove the longest edge of F.
3. Remove all edges e of F with $c(e) \geq \frac{uf}{u+f}$.
4. Split up overloaded components of the remaining forest as in algorithm A.

Theorem

Algorithm C has perfomance ratio 3γ.

Using Christofides' TSP approximation algorithm we get a 4.5-approximation algorithm in $O(n^3)$ time.

Comparison of the three approximation algorithms

- Algorithm A computes a minimum spanning tree.
- Algorithm B calls the Robins-Zelikovsky algorithm.
- Algorithm C calls Christofides' algorithm.
- Then each algorithm deletes expensive edges and splits up overloaded components.

algorithm	metric	perf.guar.	runtime
A	$(\mathbb{R}^2, \ell_1)$	4	$O(n \log n)$
A	general	5	$O(n^2)$
B	general	4.099	$O(n^{2^{10000}})$
C	general	4.5	$O(n^3)$

Experimental Results

Algorithm A on six real-world instances:

	inst1	inst2	inst3	inst4	inst5	inst6
# terminals	3675	17140	45606	54831	109224	119461
MST length	13.72	60.35	134.24	183.37	260.36	314.48
t'	117	638	1475	2051	3116	3998
L_r	8.21	31.68	63.73	102.80	135.32	181.45
$L_r + L_f$	23.07	112.70	251.06	363.28	531.05	689.19
# facilities	161	947	2171	2922	4156	5525
service cost	12.08	54.23	101.57	159.93	234.34	279.93
total cost	32.52	174.50	377.29	531.03	762.15	981.61
gap (factor)	1.41	1.55	1.59	1.46	1.44	1.42

Reduction of power consumption

Algorithm A on four chips, compared to the previously used heuristic:

chip	Jens	Katrin	Bert	Alex
technology	180nm	130nm	130nm	130nm
# clocktrees	1	3	69	195
total # sinks	3805	137265	40298	189341
largest instance	375	119461	16260	35305
power (W, old)	0.100	0.329	0.306	2.097
power (W, new)	0.088	0.287	0.283	1.946
difference	−11.1%	−12.8%	−7.5%	−7.2%

Some Open Problems

- Close the gap between 1.46 and 1.52 for the approximability of Uncapacitated Facility Location.
- Find better lower bounds than 1.46 for capacitated problems (such as CFL).
- Is Universal Facility Location really harder than CFL?
- Improve the approximation ratio for the problem with service capacities (in $(\mathbb{R}^2, \ell_1)$, with a practically efficient algorithm).
- In some real-world instances, there exists an interval graph on the terminals, and we have to partition this graph into cliques. Is there an approximation algorithm for the resulting problem?
- What other interesting problems combining facility location with network design, or routing, can be approximated?
- What about multi-stage extensions?

Further Reading

- J. Vygen. Approximation Algorithms for Facility Location Problems (lecture notes, with complete proofs and references). Can be downloaded at `http://www.or.uni-bonn.de/~vygen`
- B. Korte, J. Vygen. Combinatorial Optimization: Theory and Algorithms (Chapter 22). Springer, Berlin, third edition 2006. Also available in Japanese!
- J. Maßberg, J. Vygen. Approximation Algorithms for Network Design and Facility Location with Service Capacities. Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2005); LNCS 3624, Springer, Berlin 2005, pp. 158–169