Dynamic Facility Location : Minimizing Sum of Radii

Nicolas K. Blanchard, supervised by Nicolas Schabanel, LIAFA

August 21, 2015

General Context

In the past few years the increased study of social networks of different sorts has given rise to large databases with not just static but also dynamic information. The interactions of individuals in a hospital for example are strongly dependent on time, and clustering techniques can help understand the dynamics of those networks. Organizing this data could have many applications, from epidemiology to marketing and urban planning. A possible modelization for this is via the Facility Location problem. The first setting in which this model was studied was for a warehouse locating situation with a heuristic first published in 1963 [KH76], in the static case. Since then it has become an important subject of research covered by hundreds of publications. Many different subcases exist depending on whether we have an evolution of the network (dynamic) or not, on how exactly the cost function behaves, on the structure of the underlying space (euclidean, metric, non-metric), and on other parameters. Most of those can be proved to be **NP**-complete so the main goal is finding approximation algorithms. Several constant factor approximation algorithms already exist in different settings.

Problem Studied

The Facility Location problem consists of assigning a set of n clients $\mathcal{C}$ to a set of m services $\mathcal{F}$ so that every client is served. Although the problem is **NP**-complete and **APX**-hard, some optimal bounds have been found, $O(\log n)$ in the general (non-metric) case and $O(1)$ in the metric case (where distances follow the triangle inequality). As opposed to the static model which has been extensively studied, the dynamic model is still partially unexplored – this model has the added complexity of having distances which vary through time as well as additional costs when clients change facilities. We introduce a variant of the classical setting, where the connection cost is the sum of the radii[1] of facilities instead of the sum of the distances between clients and facilities. This second variant is a natural expansion of the problem and had received less attention until now. The aim of this internship was to study it and come up with algorithms and approximation hardness results.

Proposed Contributions

This internship yielded four results. First, we proved (Theorem 1, page 8) that the Dynamic Facility Location with Radius (DFLR) problem in the general, non-metric case, cannot be approximated within a factor $(1 - o(1)) \log n$ unless $\mathbf{P} = \mathbf{NP}$. Second, we design and prove an algorithm working in expected polynomial time ($O(nm(n + m)^{1/2})$) which gives a $2 \log n$ approximation for DFLR in the non-metric case, and show that this method can be adapted to an algorithm from [EMS14] to obtain an $O(\log n)$ approximation for non-metric DFL without radii (down from their original $O(\log nT)$). For the sum of distances between facilities and clients the algorithm present in [ANS14] achieves a constant approximation in the metric case. We show (Theorem 5, page 17) that the natural adaptation to the sum of radii has an approximation ratio at least $\Omega(\log \log n)$. Our proof is based on a combinatorial lemma dealing with permutations in trees which is of independent interest.

[1] The radius is the distance to its farthest client

Arguments Supporting Their Validity

As opposed to using many static snapshots of the network and reconciling them, DFL has been shown to capture more closely the essence of a network's structure. With the sum of radii costs, this can be extended to problems where a sum of distances cost function makes little sense. For instance, with sum of distances as costs, a good solution can tolerate having one client making a big mistake without impacting the quality of the solution (as the client is just one in many), but in the radius setting a single such mistake can increase drastically the final cost, which means that efficient algorithms to solve it might be quite useful in settings with no error tolerance. We provide formal proofs for all our results, although we do present one algorithm for which we could not yet find any guaranteed upper bound.

Summary and Future Work

This work gives asymptotically optimal results for non-metric DFL problems in both settings, leaving open only the dynamic sum of radii setting. Although we provide an algorithm, we could not get any guarantee on its approximation ratio aside from a lower bound. The next steps would be to prove an upper bound for it, and either adapt it or find another way to get a constant factor approximation for that problem. If this were to succeed, getting the APX-hardness result would be the last step to show that all the algorithms in this field are asymptotically optimal.

Contents

1 Problem and State of the art

1.1 The Facility Location Problem[2]

The problem studied in this report is a special case of Facility Location (FL), itself a sort of clustering with additional constraints. Facility Location has been shown to be very efficient for clustering, in particular when the ideal number of clusters is not known in advance. As such, we have an environment with individuals (clients), a set of services (facilities), and we are trying to link each client to a service so that every client is served, while minimizing the costs. We have the following:

- A finite set of n clients $\mathcal{C}$
- A finite set of m facilities $\mathcal{F}$
- A finite ordered set of T time steps $\mathcal{T}$
- A function d indicating the distances between each pair $(i, j) \in \mathcal{F} \times \mathcal{C}$ at each time step $t \in \mathcal{T}$, which might not follow the triangle inequality

And we are trying to output a solution composed of:

- For each time step t, a set of open facilities $\mathcal{F}^t$
- An assignment $\mathcal{C} \times \mathcal{T} \to \mathcal{F}$ of every customer to open facilities at each time step

while minimizing a cost function. This function varies greatly with the variety of settings in which the problem has been studied but is essentially composed of three parts:

- The cost to open the facilities f_i. This cost is paid at each time step $t \in \mathcal{T}$ where we want the facility to be open. This is called the hourly model. If a facility has to be either open or closed for the whole duration, a $\Omega(\log T)$ approximation hardness result and $O(\log nT)$ approximation algorithm can be found in [EMS14].
- The cost g to make a client switch from a facility to another. In the static case ($|\mathcal{T}| = 1$) this has no impact but in the dynamic one each client pays g each time it changes facilities.
- The cost to link the facilities to the clients. Two distinct settings are studied with differing methods according to how the cost varies:
  - The sum of distances setting where the cost is the sum of the distances between facilities and their assigned clients. This has been extensively studied and current algorithms are all within a constant factor of the optimal bounds [EMS14, ANS14].
  - The sum of radii setting where the cost is the sum of the radii of all facilities, that is the sum of their distances to their furthest client. This has not been studied in the dynamic setting so far and is the main subject of this report. The method we get also an application in the non-metric sum of distances setting.

Even though this problem comes mostly from clustering, it is quite easy to see implications outside of this setting. One practical example would be a phone company trying to minimize its costs and deciding where to install antenna towers and how much power to give them to cover all their clients. The more powerful the antenna, the farther it reaches, and the more expensive it is, but installing each antenna also has a fixed cost.

[2] Formally, this corresponds to Uncapacited Facility Location, but as the notion of capacities – the maximal number of clients attached to a facility – is completely outside the scope of this work we dispense with the "Uncapacited" in the acronyms.

1.2 Completeness results

In their generality, all those problems are strongly **NP**-complete. Moreover, some of them don't even allow constant factor approximation algorithms. In the sum of distances setting, if no assumptions are made on the distances, there is no $(1 - o(1)) \log n$ approximation unless $\mathbf{P} = \mathbf{NP}$. If we restrict ourselves to metric spaces – where the distances follow the triangle inequality – there are constant factor approximations (the best current bound is between 1.46 and 1.49 [GK99, Li13]). However, this is only true in the hourly setting. If we have to keep any facility opened for the whole duration there is a $\Omega(1 - o(1)) \log T$ inapproximation hardness result [EMS14].

The sum of radii setting has been less extensively studied but some previous results hold, including the $(1-o(1)) \log T$ inapproximation hardness when costs are not hourly – that is, when facilities are either open or closed for the whole duration and the opening cost is paid once or not at all. We provide a $(1 - o(1)) \log n$ approximation hardness theorem (Theorem 1 page 7) in the non-metric case, based like other proofs on a reduction from SET-COVER.

1.3 Existing algorithms

There are many algorithms with guaranteed approximation bounds on facility location problems but our research was mostly inspired by three algorithms in [EMS14], [CP04] and [ANS14], which we'll present here. The first one is an $O(\log(nT))$ approximation algorithm to dynamic facility location (DFL) with sum of distances in the non-metric case. It also introduces preprocessing techniques which are used in [ANS14] and our algorithms. The second solves metric DFL with sum of distances with cost at most $14 \times OPT$ – that is 14 times the optimal cost – and the last solves static FL in the metric radius setting with a cost at most $3 \times OPT$.

1.3.1 Linear programming relaxation

All the algorithms used are based on designing a rounding scheme for a linear programming (LP) relaxation. As the objective function and constraints of DFL are linear, we can reformulate an instance as the following integer program (IP):
We have the following constants (given to us as input):

- $\mathcal{R}_i^t$ the set of all the radii from i to its different potential clients
- f_i the cost of opening facility i
- g the cost of changing facilities for a client

And we introduce the 0-1 variables :

- y_{ir}^t equal to 1 iff we open facility i with radius r at time t (y_i^t in the sum of distances setting)
- x_{ij}^t equal to 1 iff a client j chooses i at time t
- z_j^t equal to 1 iff j switched from an other facility to i between time $t-1$ and t

The aim is to minimize

- For the sum of distances problem:

$$\sum_{i \in \mathcal{F}, t \in \mathcal{T}} f_i \cdot y_i^t + \sum_{j \in \mathcal{C}, i \in \mathcal{F}, t \in \mathcal{T}} x_{i,j}^t \cdot d^t(i,j) + g \sum_{j \in \mathcal{C}, t \in \mathcal{T} \setminus \{0\}} z_j^t$$

- For the sum of radii problem:

$$\sum_{i \in \mathcal{F}, t \in \mathcal{T}, r \in \mathcal{R}_i^t} y_{ir}^t \cdot (f_i + r) + g \sum_{j \in \mathcal{C}, t \in \mathcal{T} \setminus \{0\}} z_j^t$$

With three sets of constraints:

$\forall j,t$ $\sum_i x_{ij}^t \geq 1$: every client must be at each time step

$\forall i,j,t$ $x_{ij}^t \leq \sum_{r \geq d^t(i,j)} y_{ir}^t$ ($x_{ij}^t \leq y_i^t$ in the sum of distances problem) : no client can be assigned to a closed facility

$\forall i,j,t \geq 1$ $z_{ij}^t \geq x_{ij}^t - x_{ij}^{t-1}$: if j changes facilities then z_{ij}^t must reflect that change and be equal to 1

With the restriction that all variables take their values in $\{0,1\}$, optimal solutions to this IP are not a priori findable in polynomial time (unless $\mathbf{P} = \mathbf{NP}$), but there is a cost-preserving one-to-one correspondance between solutions to the IP and to the original problem. By relaxing those constraints and allowing the variables to take values in $[0,1]$, we can make it a linear program whose optimum can be found in polynomial time by an interior point method [LS13]. The problem is that this solution has variables which are fractional and simply rounding them up can increase the cost exponentially. The following algorithms use the LP solution to build an IP solution with a constant factor increase in expected cost.

1.3.2 EMS Algorithm

This algorithm by Eisenstat, Mathieu and Schabanel was initially made to solve sum of distances DFL in the non-hourly model and has an $O(\log(nT))$ approximation guarantee. It first preprocesses the LP solution to obtain additional properties before solving t.

Algorithm EMS Preprocessing

```
for each client $j \in \mathcal{C}$ do
    Partition time greedily into $l_j$ intervals $[t_k^j, t_{k+1}^j)$ where $l_j$ and $(t_k^j)_{k \in [l_j+1]}$ are defined as follows:
    $t_{l+1}^j = T+1$, $t_1^j = 1$ and $t_{k+1}^j$ is defined inductively as the greatest $t \in (t_k^j, T+1]$ such that
        $\sum_{i \in F} \left( \min_{t_{a-1}^j \leq u \leq t} x_{ij}^u \right) \geq \frac{1}{2}$
end for
```

This preprocessing creates a partition of the time into a set of intervals with the following properties:

- For each j, $x_{i,j}^t$ does not change inside each of j's intervals
- The total number of intervals Z is such that $Z \leq 2 \cdot \sum_{i \in F, t \in T, j \in C} z_{ij}^t$

As we will also use this preprocessing; an explanation and proof can be found in the Appendix. Once we have a solution satisfying those properties, we simply have to use the EMS algorithm.

The authors then prove that with constant probability all clients are covered during all intervals, and with constant probability all the cost is within $O(\log(nT))$ times the optimal cost. Corollary 1 gives a variation of the algorithm which lowers that bound to $O(\log n)$.

[3] This is a slightly modified version of their algorithm, although both presentations are equivalent.

[4] That is a random variable X such that for $x \in \mathbb{R}^+$, $\Pr[X \geq x] = 1 - e^{-o_i x}$. Those have very useful properties as a random exponential variable of parameter λ_0 is the smallest among a set of random variables of parameters λ_i with probability $\frac{\lambda_0}{\sum \lambda_i}$

Algorithm EMS Algorithm[3]

```
Require: A preprocessed optimal fractional solution to the LP
  for each facility $i \in \mathcal{F}$ do
    draw an exponential random variable[4] $Y_i$ of parameter $2\log(2nT)$.
  end for
  for each time step $t \in \mathcal{T}$ do
    open all the facilities satisfying $Y_i \leq y_i^t$
  end for
  for each time interval $U = [t_k^j, t_{k+1}^j)$ do
    assign client $j$ to argument of $\min_{i \in F}\left(\frac{Y_i}{\min_{u \in U} x_{ij}^u}\right)$
  end for
```

1.3.3 ANS Algorithm

This algorithm by An, Norouzi-Fard, and Svensson works in the metric sum of distances setting. It uses the same assumptions[5] on the structure of the LP solution as the EMS as well as an additional one, which is explained in the preprocessing part of the Appendix:

- For every facility i there is an $o_i \in [0,1]$ such that for all j and t, $x_{i,j}^t \in \{0, o_i\}$

Algorithm ANS Algorithm

```
Require: The fractional preprocessed solution to the LP
  for each facility $i \in \mathcal{F}$ do
    draw a random variable $Y_i$ following an exponential distribution of parameter $o_i$
    Compute the list $L$ of facilities sorted by non-decreasing values of those variables.
  end for
  Compute $L'$, random uniform permutation of the clients
  for each time step $t \in \mathcal{T}$ do
    for each facility $i \in \mathcal{F}$ do
      find the first client $j$ in $L'$ that has $x_{ij}^t > 0$ in the LP solution and draw an arrow to it
    end for
    for each client $j \in \mathcal{C}$ do
      find the first facility in $L$ with $x_{ij}^t > 0$ and draw an arrow to it
    end for
    Starting from any client and following the arrows one always ends up in a cycle of length two
    (with one facility and one client). Assign each client to the facility in the corresponding cycle
  end for
```

Key ideas of the proof

This algorithm has at least a $\frac{1}{4}$ chance of attaining an approximation ratio better than 14. The proof is based on the fact that once the initial connections are made, the connection paths lengths are exponentially distributed with constant expectation.
The authors prove that for any edge the number of connection paths going through it is proportional to the cost paid for the edge in the LP solution. For essentially the same reasons, if a client changes facilities between two time steps it only affects a constant number of other clients – at most 7 in expecation. Finally, a facility i is open if and only if the client j it is pointing to points towards i.

[5] Some of those assumptions might not be necessary for the algorithms to work but are used in the proofs.

This happens with probability $x_{ij}^t \leq y_i^t$ so the expected opening cost for each time step is $\sum_i y_i^t$ which is optimal.

We propose a natural adaptation of this algorithm for metric DFLR and we explain the intuition on it in section 3.

1.3.4 The Pruning-Clustering Algorithm

The other algorithm we were inspired by is a primal-dual algorithm from [CP04] which introduced the LP for FL with sum of radii cost in the static setting and achieved a 3-approximation.
We start with the dual[6] of the LP:

Maximize $\sum_j \alpha_j$

With constraints

$$\forall i, r \sum_{j:d(i,j)\leq r} \alpha_j \leq r + f_i$$

where the dual variable α_j corresponds to the utility[7] of client j.
The algorithm increases the variables α_j arbitrarily until it can't increase any of the variables anymore (which is in this case equivalent to solving the dual LP). We call a facility-radius couple tight if the corresponding constraint is tight.

Algorithm Pruning-Clustering Algorithm

```
Let $\mathcal{T}$ be the set of tight $(i, r)$ and $\mathcal{F} := \emptyset$
**repeat**
    Let $(i, r)$ be the facility of largest radius in $\mathcal{T}$ (break ties arbitrarily)
    Let $\mathcal{N}$ be the set of all facility-radius couples $(i', r')$ intersecting $(i, r)$ , that is such that $d(i, i') \leq r + r'$ (including itself)
    Let $\mathcal{F} := \mathcal{F} \cup \{(i, 3r)\}$
    Let $\mathcal{T} := \mathcal{T} \setminus \mathcal{N}$
**until** $\mathcal{T} = \emptyset$
**return** $\mathcal{F}$
```

Key ideas of the proof

The idea here is that any client j is covered by at least one facility in $\mathcal{F}$ or the corresponding constraint wouldn't be tight and α_j could be increased. Moreover, each time we remove a facility-radius couple from $\mathcal{T}$, it's because it intersects (i, r). As it also has radius smaller than r, it is entirely included in $(i, 3r)$. Each client is then covered and can be assigned to any of the facilities covering it. A rigorous proof is found in the original paper.

Remark. Both of these algorithms only work in the metric case and fail completely when the triangle inequality is not satisfied (the first one by giving solutions potentially having an exponential increase in cost and the second by giving solutions where some clients aren't covered).

[6] Any linear program with a cost function to minimize can be transformed into a maximization problem such that the values of the best solutions are identical due to the duality theorem. Each constraint is transformed into a variable and vice-versa, by a polynomial algorithm. For more details see [Chv83] for example.

[7] Our interpretation of the dual is that here each client tries to maximize its utility but for any facility there can't be more total utility used by clients in its radius than the price paid to open the facility with this radius.

2 Non-Metric Dynamic Facility Location with Sum of Radii Cost

The three algorithms shown previously solve the dynamic sum of distances and the static radius cases. The natural question is what happens in the dynamic radius case, especially as this formulation might be a better suited model for some practical applications. We prove that the non-metric case does not admit $(1-o(1)) \times \ln n$ approximation polynomial time algorithms. Despite this we are able to provide an algorithm that is within a factor 2 of the optimal reachable approximation ratio. Our technique applies as to get an improvement of the current EMS for the DFL with sum of distances in the non-metric case.

2.1 Inapproximability

To prove hardness of approximation for DFL we will reduce from SET-COVER, which is the classical problem for logarithmic inapproximability results. We consider a family $\mathcal{A}$ of subsets of $\{1, 2, \dots n\}$ and we want to find the smallest collection of subsets $\mathcal{B} \subseteq \mathcal{A}$ such that every integer in $\{1, 2, \dots n\}$ is in at least one element of $\mathcal{B}$. It turns out that no known polynomial time algorithm can guarantee outputting a solution with cost under $\log n$ times the cost of the optimal solution, formally:

Fact 1. *[DS13] SET-COVER admits no $(1-o(1)) \times \ln n$ approximation unless $\mathbf{P} = \mathbf{NP}$.*

This was proven using projection games and gives us:

Theorem 1. *Non-metric sum of radii Dynamic Facility Location admits no $(1-o(1)) \times \ln n$ approximation unless $\mathbf{P} = \mathbf{NP}$.*

Proof. We consider a SET-COVER instance and create an instance of DFLR with $f_i = 1$, $g = \infty$ and $T = 1$. We take one facility per set in $\mathcal{A}$ and a set of $n = |\bigcup_{A \in \mathcal{A}} A|$ clients. We set the distances $d(i,j)$ to 0 if j is in the set corresponding to i and to ∞ if not. Any solution with cost less than ∞ is a collection of sets covering all the elements and is a solution of SET-COVER, and reciprocally. Moreover both their costs are equal. An algorithm that would guarantee a $(1-o(1)) \times \ln n$ approximation on DFL would then guarantee the same on SET-COVER which would imply $\mathbf{P} = \mathbf{NP}$ from Fact 1. $\square$

Remark. In the theorem we set g to ∞ for ease of reading but it is enough to set it to $|\mathcal{A}|$. Moreover, this result uses a single time step hence also holds for Static Facility Location with Sum of Radii.

2.2 Asymptotically optimal approximation algorithm

Our goal here is to get an algorithm whose approximation ratio is as close to the $(1-o(1)) \log n$ bound as possible. We first get an algorithm with approximation ratio $O(\log(nT))$ which we use as a subroutine in a second algorithm achieving an $O(\log n)$ approximation. This algorithm is our first major contribution in this report.

2.2.1 $\log(nT)$ approximation Algorithm

We start with a solution to the LP shown earlier, modify it slightly and then get a randomized algorithm which outputs a low cost solution with constant probability.

We will start with the preprocessing introduced in [EMS14]. For each client we want to have a partition of the time into a set of intervals with the following properties:

- For each j, $x_{i,j}^t$ does not change inside each of j's intervals
- The total number of intervals Z is such that $Z \leq 2 \cdot \sum_{i \in F, t \in T, j \in C} z_{ij}^t$

Algorithm 1

```
Require: The preprocessed solution to the linear program
  repeat
    for each facility $i \in \mathcal{F}$ do
      Draw $a_i$ uniformly in $[0,1]$
      Open the facility $i$ at each time $t$ with maximum radius $r$ such that $a_i \leq \sum_{p \geq r} y^t_{ip}$
    end for
    for each client $c \in \mathcal{C}$ and each of its intervals do
      if it is covered by at least one facility over the whole interval then
        Assign it to any covering facility
      end if
    end for
  until all clients are covered during all intervals (without closing facilities between repetitions)
```

The preprocessing is explained in the Appendix, and increases the cost of the linear program solution by at most 2. It allows us to show that if a client is covered by a single facility during each of its intervals, the changing cost is at most Z hence at most twice the optimal changing cost.

Theorem 2. *Algorithm 1 gives an $O(\log(nT))$ approximation with constant probability.*

Proof. Consider the probability that a client j is covered during a given interval. If a facility i has an a_i lower than $\sum_{r \geq d^t(i,j)} y^t_{ir}$, client j will be covered by i. From the LP we know that

$$\sum_{r \geq d^t(i,j)} y^t_{ir} \geq x^t_{ij}$$

so the probability of a_i being lower than $\sum_{r \geq d^t(i,j)} y^t_{ir}$ is at least x^t_{ij}.
If the client j is not covered then all of the facilities have $a_i \geq x^t_{ij}$ which happens with probability at most $\prod_{i \in F}(1 - x^t_{ij})$
The probability that a client is left uncovered in a time step is then at most $1 - \prod_{i \in F}(1 - x^t_{ij})$. By a simple recursion based on the fact that the maximum of $x \times (b - x)$ is reached when $x = b - x$, this function has its global minimum when all the x^t_{ij} are equal. This happens when they are all equal to $\frac{1}{n}$, hence the probability is greater than $1 - (1 - \frac{1}{n})^n \geq \frac{1}{2}$.

If we repeat this $\lceil \log_2(Z) \rceil + 1$ times, we get that each interval is not covered with probability at most

$$\frac{1}{2^{\lceil \log_2 Z \rceil + 1}} \leq \frac{1}{2Z}$$

By using the union bound, the probability that at least one client is not covered during one of its intervals is at most $\frac{1}{2}$. Hence with probability at least $\frac{1}{2}$ the algorithm repeats at most $\lceil \log_2(Z) \rceil + 1$ times. As in Algorithm 2 from [EMS14], at each repetition, the expected cost for each facility is equal to its cost in the LP solution. As we open each facility at each time step with the maximum radius over all repetitions, the expected cost is at most the sum of costs over all repetitions, and the same is true for the sum over all facilities. By Markov's inequality with probability at least $\frac{2}{3}$ the cost of the sum is not more than 3 times the sum of expectations, hence $3 \times (\lceil \log_2(Z) \rceil + 1) \times OPT$.

By using the union bound again, with probability at least $\frac{2}{3} - \frac{1}{2} = \frac{1}{6}$ all the clients are covered in less than $\log_2(Z) + 1$ repetitions and the cost is at most 3 times its expectation.

As $Z \leq n \times T$, we output with probability at least $\frac{1}{6}$ a solution with cost at most

$$6 \lceil \log_2(Z) \rceil \times OPT \leq 6 \left(1 + \lceil \log_2(nT) \rceil\right) \times OPT$$

□

Remark. By a careful analysis (see Appendix), we can get approximation ratios arbitrarily close to $\beta \times \ln(Z) \times OPT$, with

$$\beta = 2\log_2\left(\frac{e}{e-1}\right) \times \log_2(e) < 1.93$$

2.2.2 $\log n$ Approximation Algorithm

We can now use Algorithm 1 as a black box to get a better bound in the cases where $\log Z$ is much larger than $\log n$.

Algorithm 2

```
if $Z \leq n^3$ then
  Run Algorithm 1
else
  repeat
    Greedily find the biggest $t_1$ such that the number of clients changing facilities between $t_0$ and
    $t_1$ is less than $2n$
  until you have a partition of $\mathcal{T}$ into at most $\frac{Z}{n}$ chunks such that each chunk except potentially
  the last has between $n$ and $2n$ clients changing facilities.
  Solve each chunk independently using Algorithm 1
  return the assignment which corresponds on each chunk to what was given by Algorithm 1
end if
```

Theorem 3. *Algorithm 2 is an $O(\log n)$ approximation with constant probability for non-metric DFLR.*

Proof. First off, if $Z \leq n^3$ then $\log(Z) \leq 3\log n$ and we have a $25\log_2(n)$ approximation ratio with constant probability. Second, it is always possible to cut the solution in chunks with at most $2n$ clients changing facilities (or equivalently, intervals) because at each time step at most n new intervals start[8]. Each chunk is solved on a partial instance where Z' is at most $2n$, hence Algorithm 1 returns an approximation with ratio at most $6(\log n + 3)$. The total cost is then the cost of those approximations plus the cost of joining the solutions. As each joining costs at most $n \times g$ (the cost of n clients changing facilities), the total joining cost is at most $\frac{Z}{n} \times n \times g$ hence at most $Z \times g \leq 2 \times OPT$. We finally get an approximation ratio of $6 \cdot \log n + 20$. □

Remark. As with Algorithm 1, the ratio could be improved by only running Algorithm 1 when $Z \leq n^{1+\gamma}$ for small γ and also get a bound of $\beta \times \ln(n) \times OPT$. Moreover, by comparing the output of the LP to the cost of the solution and relaunching when the cost is too high we can get a Las Vegas algorithm with the same expected polynomial time, as the biggest time factor is due to the linear solver (using general interior point solvers). Using the solver from [LS13] we can then get an expected computation time of $O(nm(n+m)^{1/2})$.

2.3 Application to the non-metric sum of distances problem

In the non-metric dynamic sum of distances problem, Algorithm 3 from [EMS14] reached an $O(\log nT)$ approximation, with a $\Omega(\log n)$ hardness result from [Hoc82].

[8] To show by example, suppose we have $n = 4$ clients and $Z = 100$. We can maintain a counter and at each time step we increase the counter by the number of clients who switch. As soon as the counter becomes greater than n we finish the chunk and begin a new one, resetting the counter for the next time step. As the counter was under n before the last time step was added to the chunk, it contains strictly less than $2n$ clients changing and this is true for all chunks.

The authors proved that this algorithm gives in the worst case an $8 \log n$ approximation[9] with probability $\frac{1}{4}$. Using this algorithm as a black box instead of Algorithm 1 inside Algorithm 2 and following exactly the same proof as previously, one gets Algorithm 2B and the corollary:

Corollary 1. *Algorithm 2B gives a* $16 \log n$ *approximation with constant probability for non-metric DFL with sum of distances cost.*

Remark. One should notice that this method is of no use in the non-hourly model and cannot be applied to Algorithm 1 from the same paper (which also has an $O(\log nT)$ approximation ratio).

3 Metric Dynamic Facility Location with Sum of Radii Cost

We shall now consider the metric case, where distances between each pair of elements are defined and follow the triangle inequality. Because of this the reduction from SET-COVER cannot be implemented directly as we require distances to be either 1 or greater than n; hence we do not have the $\log n$ inapproximation hardness result. We can however show that a natural adaptation of the ANS algorithm does not yield the constant approximation ratio which might be reachable. We shall present the algorithm and the counterexample along with some preliminary results about it before getting to the proof that the algorithm's approximation ratio is at least $\Omega(\log \log N)$.

3.1 Modified ANS Algorithm

We propose a natural adaptation of the ANS algorithm to adapt it to the sum of radii setting. We use the same preprocessing (see Appendix) to make sure that:

- Each client has constant $x^t_{i,j}$ on each of its intervals, and the number of intervals is at most twice the changing cost of the optimal LP solution.
- Each facility is duplicated so that we have one virtual facility per radius, which is then decomposed into multiple virtual facilities[10] to ensure that for all i, j, t $x^t_{i,j} \in \{0, o_i\}$.

3.2 Lower bound for the modified ANS algorithm

Remark. From here until the end of the section we shall adopt a new notation with $N = 2^n$ clients as it is more comfortable to work with.

We did not manage to prove an upper bound for a natural adaptation of the algorithm presented in the previous section, but we did prove that it cannot yield an approximation better than $\Omega(\log \log n)$ in the worst case. This is shown by introducing a structure T_n and proving that it can "confuse" the LP solver so that every facility is taken with probability $\frac{1}{n}$. Then we can show that for any open facility i, with high probability a client far from i will connect to it. As this is true in expectation it means that the expected total cost will be much higher than the optimal. Finally, by modifying T_n slightly we can also get the algorithm to fail with high probability.

Before getting to the counterexample we need a general lemma which we'll prove at the end of the section:

Lemma 1. *Given a perfect binary tree* T *on* $N - 1 = 2^n - 1$ *nodes rooted in* r *and a random uniform permutation* p *on its internal nodes, the probability of there being a leaf* l *such that* $p(r) > p(i)$ *for each ancestor* i *of* l *is at least* $\frac{1}{14}$ *.*

[9] This factor 8 could be improved in a fashion similar to the constant in Theorem 3, as was mentionned by the original authors.

[10] This means that we create a polynomial number of facilities which are duplicates of the original ones, existing in the exact same place. Once we output the final assignment we regroup all of them. This makes for easier manipulation as the virtual facilities behave statically: either a client is attached to them with a constant fraction common to all clients or not at all.

Algorithm Modified ANS Algorithm

```
Require: a solution to the LP
  Preprocess the solution to get the required properties
  for each facility $i \in \mathcal{F}$ do
    draw a random variable $Y_i$ following an exponential distribution of parameter $o_i$
  end for
  Let $\Pi$ be the permutation of the facilities sorted by non decreasing $Y_i$
  Let $\sigma$ be a uniform random permutation of the clients
  for each time step $t \in \mathcal{T}$ do
    Link each client $j$ to the facility $i$ of lowest rank in $\Pi$ for which it has $x^t_{i,j} \neq 0$ (which we'll call
    its target)
    Link each facility $i$ to its client $j$ of lowest rank in $\sigma$ that has $x^t_{i,j} \neq 0$ (the facility's target)
    This gives a set of cycles with trees attached to them : open each facility in a cycle and connect
    all clients in the connected component to it
  end for
  return this assignment
```

Remark. We can already have a simple lower bound on that probability, by considering a fixed branch and seeing that as it corresponds to a uniform permutation of the branch, the probability that $p(r) < p(i)$ for all i in the branch is exactly $\frac{1}{n}$.

3.2.1 Counterexample preliminaries

Definition 1. We consider an arborescent structure T_n in n dimensions (Figure 1). It is composed of n different levels of hierarchy, with level 0 being a unique facility at the origin of the space and level $k \geq 1$ composed of all possible facilities at coordinates $(a_0, a_1, \dots, a_{k-1}, 0, 0, \dots, 0)$, where $a_l = \pm 2^{-l}$. The clients are at each possible tuple $(a_0, a_1, \dots, a_{n-1})$. We then have $N = 2^n$ clients and $2^n - 1$ facilities. We say that a client C (or a facility) is *under* a facility F if the non-zero coordinates of F are a prefix of those of C. We consider a metric space with the infinity norm L^∞. The distance from F to clients under it is exactly 2^{-k} where k is F's level in the hierarchy.

Lemma 2. *The fractional solution S_n to the linear program launched on T_n where all facilities are taken with mass $\frac{1}{n}$ and contribute $\frac{1}{n}$ to all clients under them has optimal cost.*

Proof. First, the optimal cost is 1. This can be proved by induction as follows: it is trivially true when $n = 1$. Suppose that it is true for a certain n, and look at T_{n+1}. That structure is composed of two instances A and A' of T_n where the distances have been divided by 2, and an additional facility i_0 between them. We have to cover the sets of clients both in A and A'. Suppose that any facility in A can cover a client in A'. This means that the facility has radius at least 2, and any mass on it can be moved instead to i_0 which can cover the same clients while having radius 1. Hence in no optimal solution can a facility in A (respectively A') cover a client in A' (respectively A). Each client is then covered by a mix of its optimal solution in A (or A') and i_0. Any mass y_{i_0} on i_0 reduces the mass needed on both A and A' by y_{i_0}, and the optimal cost is the minimum of $y_{i_0} + 2(1 - y_{i_0})\text{OPT}(A)$ where $\text{OPT}(A)$ is the cost of an optimal solution on A (which we pay for both A and A'), hence it is exactly 1.

As we have 2^k facilities at level k, if we open them with radius 2^{-k}, we cover all client with a total radius cost equal to 1. A solution consisting of a single level where all facilities are taken with mass 1 is called pure, and as those are all optimal, any linear combination of them such that the total of the masses assigned to each level is 1 is also optimal. S_n is then a solution of cost 1 and is optimal. It happens that in practice no linear solvers would give such a solution but we can modify very slightly the original instance to force this mixed solution to be the optimum, for details see the Appendix. □

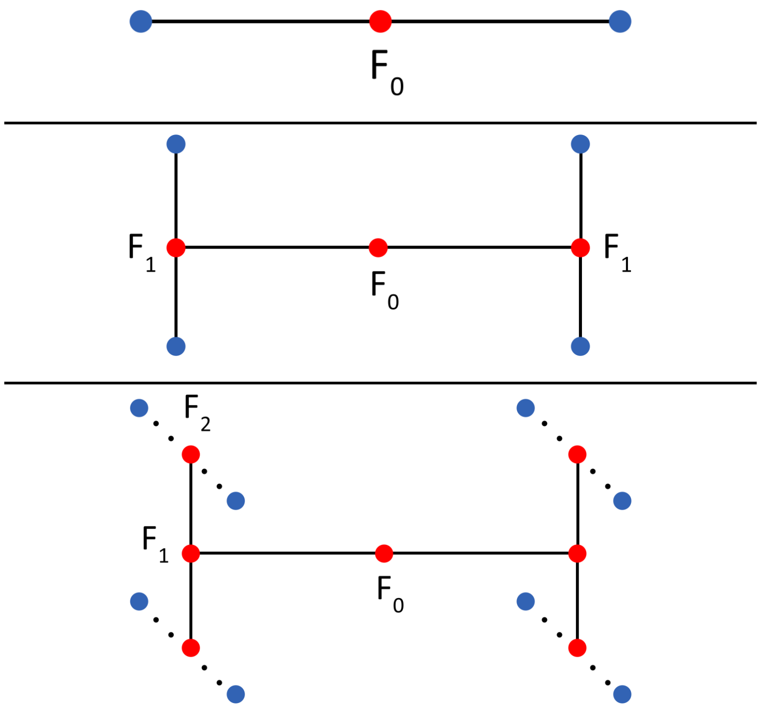

Figure 1: From top to bottom T_1, T_2 and T_3 with the facilities in red and the clients in blue. Lines represent the arborescent structure and in T_3 the dotted lines represent distances in the third dimension.

Lemma 3. *The cost of the modified ANS algorithm when launched on S_n is at most $\log N$.*

Proof. Suppose that facility i is open with radius 2^{-h}, with h smaller than i's level. For that to happen a client must be attached to i and be at distance 2^{-h}. As this client cannot be under i or under any of its ancestors with level at least $h+1$, it must be going first through a facility of level at most h before being redirected to i. Such a facility can either open itself or redirect a batch of clients to exactly one other facility under it, open with a radius at most 2^{-h}. This means that for any k there are at most 2^k facilities opened with radius 2^{-k}, and by summing all this we get a cost of at most 1 per level hence at most n. □

3.2.2 Proof of the lower bound

We now have the tools to prove the following lemma, which is the last step before prooving Theorem 4:

Lemma 4. *When launched on S_n, the modified ANS algorithm's cost expectation is $\Omega(\log\log N)$.*

Remark. This instance uses a single time step, and worse approximation might be obtained by using a dynamic counter-example.

Proof. First let's consider the cost to open facilities before increasing the radius by assigning clients which go through other facilities. The probability of a facility i opening is the probability of the client of lowest rank attached to i targeting i. This client has n ancestors all taken with mass $\frac{1}{n}$ in S_n, so will target i with probability $\frac{1}{n}$. Any facility is then chosen with probability equal to $\frac{1}{n}$, with an initial radius – before any clients are redirected to it – equal to 2^{-k}. As each of the 2^k facilities on level k is taken with probability $\frac{1}{n}$ with initial radius 2^{-k}, the sum over all levels and facilities is then 1. The increase between this cost and the final cost is due to clients far away from the facility choosing to be attached to it.

We now consider a facility that is chosen in the solution and we estimate its expected cost. A client C is attached to an open facility F if and only if:

- C is under F and F has the smallest rank among C's ancestors, OR
- there is a facility F_0, ancestor of both C and F such that F_0's target is a descendant of F, AND C's smallest ancestor is F_0 (Figure 2).

In the first case attaching the client doesn't change the radius of F. In the second, F has to open with radius at least 2^{-k} where k is the level of the closest common ancestor of F and C (we call that being opened at level k).

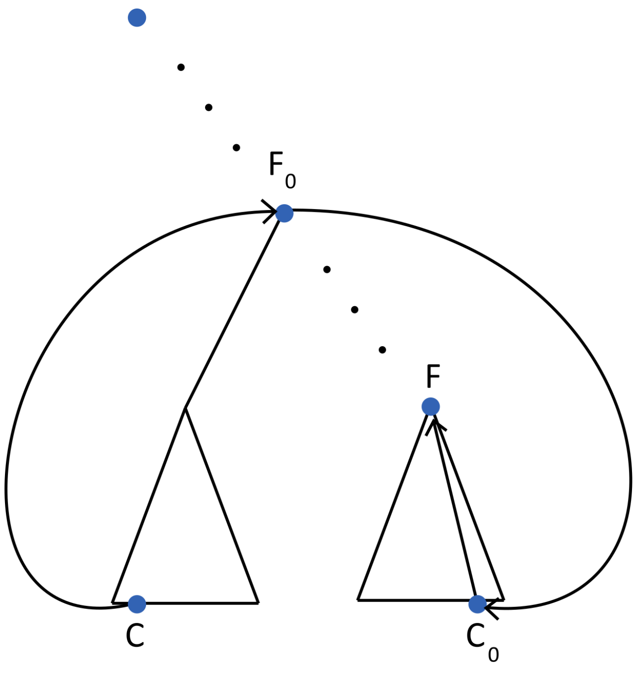

Figure 2: C is attached to F if and only if C_0, target of F_0, itself target of C is under F and targetting it.

If we consider an open facility F at level k then its expected cost will be

$$\sum_{l\leq k} \Pr[\text{opening at level } l] \times 2^{l}$$

which means that the ratio between the expected cost and the optimal cost -2^k for a facility at level k – will be

$$\sum_{l\leq k} \Pr[\text{opening at level l}] \times 2^{l-k}$$

Consider an ancestor of F, F_0, at level k, which has a smaller rank than its ancestors. For F to open at level k it means that a client C under F_0 goes through F_0 and is redirected to F. Moreover, C and F's closest common ancestor has to be F_0 or F would be open at a level higher than k . The probability of F opening at level k is then at least equal to the probability of a client under F_0 with no closer common ancestor with F going through F_0 to reach F. Which is to say the probability of F_0's target being under F, times the probability that a client that isn't under the subtree of F_0 containing F having F_0 for target. Those two events are not independent but are correlated so the probability of both happening is at least the product of probabilities.

The first one's probability is $\frac{|\text{clients under } F|}{|\text{clients under } F_0|}$, hence 2^{k-l}. The second is bounded with Lemma 1, as being the target of a client is exactly the same as having a leaf x satisfying $P(x)$, and we get a lower bound of $\frac{1}{14}$.

Moreover, the subtree not containing F has a probability at least $\frac{1}{2}$ of having its root bigger than F_0 and hence we have a probability at least $\frac{1}{28}$ of having a client targetting F_0.

The expected cost is then at least

$$\sum_{l<k} 2^{k-l} \times \frac{1}{28} \times 2^{l-k} \times \Pr[F_0 \text{ smaller than ancestors}]$$

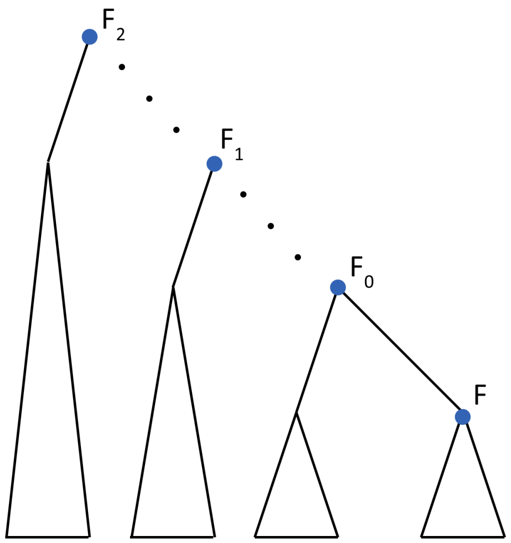

Figure 3: Each of the ancestors F_0, F_1, F_2 of F has a constant probability of having a client pointing to it (or higher) through the half of the tree not containing F. They also have a $\frac{1}{l}$ probability of being smaller than their ancestors. Hence the $O\left(\sum_{l<k} \frac{1}{k}\right)$ expectation for the increase in cost for F.

The probability of an ancestor of F at level l being smaller than its ancestor is equal to $\frac{1}{l}$, so the expected cost becomes at least

$$\sum_{l<k} \frac{1}{28} \times \frac{1}{l} \geq \frac{\ln(l)}{28}$$

For all facilities at levels greater than $\frac{n}{2}$ this at least $\frac{\ln(n)-1}{28}$ (Figure 3).

The expected cost of the output of the algorithm being the sum of the costs for opening the facilities, by linearity of expectation it is then also greater than

$$\sum_{F \text{ opened with level } \geq \frac{n}{2}} \mathbb{E}[\text{cost of F}]$$

which is finally greater than $\frac{\ln n}{57}$ for n large enough. This proves that the expected cost of the algorithm is in $\theta(\log n)$ with $N = 2^n$ clients, hence at best a $\log \log N$ approximation.

$\square$

Remark. We can improve the constants up to $\frac{1}{6} \log \log N$ using improved bounds for Lemma 1.

Our second major contribution is that with high probability, the modified ANS algorithm does not yield a better approximation than $\theta(\log \log N)$. Before proving this result we need one last tool, Hoeffding's inequality:

Theorem 4. *(Hoeffding [Hoe63]) Let $X_1, X_2, \ldots X_k$ be i.i.d. random variables which are almost surely bounded in $[a_i, b_i]$ and let $\overline{X}$ be their mean. Then*

$$\Pr\left(\left|\overline{X} - \mathbb{E}[\overline{X}]\right| \geq t\right) \leq 2\exp\left(-\frac{2k^2t^2}{\sum_{i=1}^{k}(b_i - a_i)^2}\right)$$

Theorem 5. *There is a constant c such that probability of the modified ANS algorithm giving a approximation smaller than $\frac{\log\log N}{c}$ is at most $\frac{1}{\log N}$.*

Proof. Consider an instance composed of k copies of the structure T_n, each far from the others. This means that the linear program will solve it just like multiple simple instances of T_n, and so will the algorithm, so the expected cost will be the sum of expected costs for S. However as all the different instances are evidently independent the distribution of the cost can be approximated, even if we only have very limited knowledge about it. We consider k disjoint instances, and we let the X_i be the cost of each. Then, using Lemma 3, the probability that the approximation will be better than half the expectation is at most $2\exp\left(\frac{-2k}{\log^2 N}\right)$.

By setting $k = \log^3 N$ we get that the probability of getting a good approximation becomes less than $\frac{1}{N}$. We now have a probability $1 - O(\frac{1}{N})$ of getting an average cost larger than $\frac{\log\log N}{c}$ but we increased the cost of the optimal solution by a factor $\log^3(N)$. This is however irrelevant as this means that with high probability the total approximation ratio is not better than $\frac{\log\log(N\times\log^3 N)}{c} \geq \frac{\log\log N}{c+1}$ for N big enough. □

Corollary 2. *Repeating the modified ANS algorithm a polynomial amount of times and taking the best solution is not enough to guarantee a solution better than $\frac{\log\log N}{c'}$ for a constant c'.*

Proof. By setting $k = N^2$ the total approximation ratio does not change by more than a $o(1)$ factor but the probability of getting a better approximation becomes exponentially small. □

3.2.3 Proof of Lemma 1

Recall the statement of Lemma 1: given a perfect binary tree T on $N - 1 = 2^n - 1$ nodes rooted in r and a random uniform permutation p on its internal nodes, the probability of there being a leaf l such that $p(r) > p(i)$ for each ancestor i of l is at least $\frac{1}{14}$.

Remark. In the following we can consider n to be even to have no rounding problems but as the probability decreases monotonously with n it is also true for odd n – this is due to the fact that the perfect binary tree of height $2k - 1$ is included in the tree of height $2k$ so any path from root to leaf in the first is also a path in the second, and the proportion of permutations in which you have one is strictly greater in the tree of height $2k - 1$. This also means that it is true for full binary trees which are not perfect[11].

Proof. The proof works in the following way: we are looking for a path from root to leaf where the root has the biggest rank, and we show that if the root's rank is supposed high, then with constant probability we can find a path from the root to a node at height $n/2$, and then another one from this node to the leaf. For that we rely on the fact that if we only look at the subtree from the root to height $n/2$, the number of nodes in such a subtree is an arbitrarily small proportion of the total nodes. If the root is among the top quarter of the ranks, then if we look at the children of any node in the subtree there is approximately a $\frac{3}{4}$ chance of its rank being smaller than the root's. This probability will stay close to $\frac{3}{4}$ even if we already found lots of such nodes because we are only looking at an ε proportion of the nodes.

[11] Every node in a full binary tree has either zero or two children but the leaves may be at different levels as opposed to the perfect binary tree of height n where all leaves are at height n.

This allows us to show that with constant probability there is a node at height $\frac{n}{2}$ whose ancestors all have values smaller than the root's. Moreover, with constant probability, on the subtree rooted in this node at least one leaf also has that property, meaning that one leaf of T has the property we wanted with constant probability. The trick is that as we only look at a very small fraction of T we are able to bound the effects of the dependence between the values drawn for the nodes.

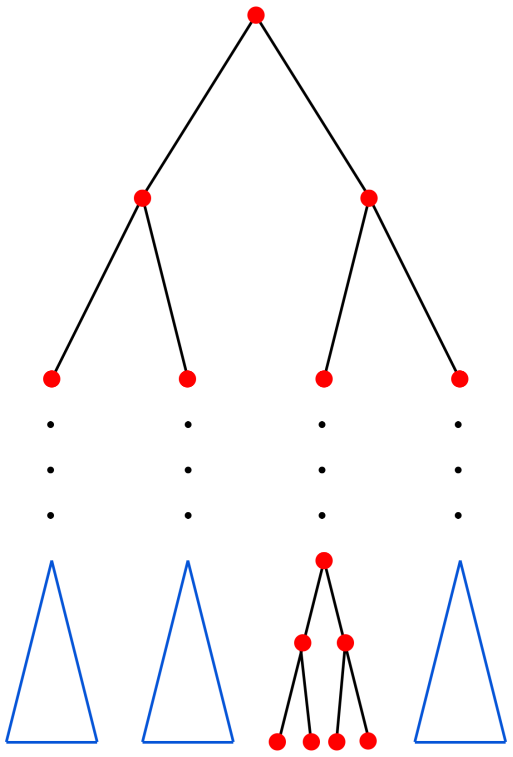

Figure 4: In the final tree we only look at the top half and one of the subtrees (in red), hence at most $2^{n/2+1}$ nodes out of $2^n - 1$

More formally, let $x = p(r)$. We define as $P(i)$ the property that all ancestors of i and i itself have a rank in the permutation smaller than x. We consider the probability that at least one internal node at height $n/2$ satisfies P. We only consider nodes at depth at most $n/2$ hence at most $2^{n/2}$ nodes. The probability each time we take a node that it ends up less than x is then at least $\frac{x}{2^n} - \varepsilon$ where ε is the proportion of nodes whose values we are considering, and the probability of being over x is at most $1 - \frac{x}{2^n} - \varepsilon$. Let $p(x, n, h, \varepsilon)$ represent the probability that in a perfect binary tree of height n, at least one leaf in a perfect binary subtree of height h satisfies P, relative to its root x, when we are only considering at most an ε proportion of the nodes.

Then, for $h \leq n/2$,

$$p(x, n, h+1, \varepsilon) \geq 2 \times p(x, n, h, \varepsilon) \times \Pr[A] + \left(1 - (1 - p(x, n, h, \varepsilon))^2\right) \times \Pr[B]$$

where A is the event in which a single child of the root has a lower rank than x, and B the event in which both children of the root have a lower rank than x.

Indeed, consider a node and its two children and subtrees. Then the node satisfies P if either exactly one of the two children is over x and its subtree satisfies P, or both children are over x and at least one of its subtrees satisfy P. Those two events being disjoint we can sum their probabilities.

By setting ε as $2^{-n/2}$, we know that $\Pr[B] \geq \left(\frac{x}{2^n} - \varepsilon\right)^2$ because the probability of any child being under x is equal to the proportion of ranks left under x. However as we have considered at most an ε proportion of the nodes, this proportion is at least $\frac{x}{2^n} - \varepsilon$. Similarly, we can see that $\Pr[A] \geq \left(\frac{x}{2^n} - \varepsilon\right)\left(1 - \frac{x}{2^n} - \varepsilon\right)$.

We then get that

$$p(x,n,h+1,\varepsilon) \geq 2\times p(x,n,h,\varepsilon)\times\left(\frac{x}{2^n}-\varepsilon\right)\left(1-\frac{x}{2^n}-\varepsilon\right)+\left(\frac{x}{2^n}-\varepsilon\right)^2\left(2\times p(x,n,h,\varepsilon)-p(x,n,h,\varepsilon)^2\right)$$

which is

$$p(x,n,h+1,\varepsilon) \geq p(x,n,h,\varepsilon)\times\left(\frac{2x}{2^n}-5\varepsilon\right)-\left(\frac{x}{2^n}-\varepsilon\right)^2 p(x,n,h,\varepsilon)^2$$

With probability at least $\beta-\alpha-4\varepsilon$, x verifies

$$\frac{1}{2}\leq\alpha\leq\frac{x}{2^n}-\frac{5}{2}\varepsilon\leq\frac{x}{2^n}+\varepsilon\leq\beta\leq 1$$

and we get that $p(x,n,h+1,\varepsilon)\geq 2\alpha\cdot p(x,n,h,\varepsilon)-\beta^2\cdot p(x,n,h,\varepsilon)^2$.

This is a quadratic recurrence relation. If we set $u_{n+1}=f(u_n)$ with $f(y)=2\alpha y-\beta^2y^2$ we can see that $f(y)>y$ if and only if $y<\frac{2\alpha-1}{\beta^2}$. Moreover, if $u_0>\frac{2\alpha-1}{\beta^2}$ then for all n, $u_n>\frac{2\alpha-1}{\beta^2}$ and we get

$$p(x,n,h,\varepsilon)>\frac{2\alpha-1}{\beta^2}$$

This means that the probability, when taking the subtree of depth $n/2$, of finding a node at depth $\frac{n}{2}$ satisfying P is at least $\frac{2\alpha-1}{\beta^2}\left(\beta-\alpha-4\varepsilon\right)$.

Now let's consider a tree T and a node at depth $n/2$ satisfying P which we have with probability $\frac{2\alpha-1}{\beta^2}$. We can look at the subtree beneath this node and with a probability $\frac{2\alpha-1}{\beta^2}$ there will also be a branch consisting entirely of nodes with values less than x. The only precaution to take care of is that the total number of nodes we have looked at has doubled hence $\varepsilon'=2\varepsilon$ (Figure 4). This gives a final probability of $\left(\frac{2\alpha-1}{\beta^2}\right)^2\left(\beta-\alpha-8\varepsilon\right)$. If we set x to be within $[\frac{5}{6},1]$, we get a final probability of $\frac{2}{27}-O(\varepsilon)$. As we have only considered a number of nodes at most equal to $2\times 2^{n/2}$, $\varepsilon\to 0$ and the probability of there being a leaf of the tree satisfying P is at least $\frac{1}{14}$. $\square$

Remark. By bounding the value at each step more precisely we can increase this probability to $\frac{1}{3}$ and then 0.35 at the cost of lengthier computations. It is also possible to manually get an upper bound of $\frac{39}{70}\leq 0.53$ by looking at a tree of height 3. Detailed computations can be found in the Appendix.

4 Conclusion

To sum up our contributions as well as the current state of the art, here are the approximation ratios and hardness bounds in the various cases, where * represents our results:

	Metric (algorithm/hardness)		Non-Metric (algorithm/hardness)	
	sum of distances	sum of radii	sum of distances	sum of radii
Static	1.52 [MYZ06]/ 1.46[GK99]	3/NP-Hard [CP04]	$O(\log n)$[EMS14]/$\Omega(\log n)$	$2\log n/(1-\varepsilon)\log n$ *
Dynamic	14 [ANS14]/ 1.46[GK99]	?/NP-Hard [CP04]	$O(\log n)$*/$\Omega(\log n)$[Hoc82]	$2\log n/(1-\varepsilon)\log n$ *

We also showed that a natural adaptation of the ANS algorithm from [ANS14] gives at best an $\Omega(\log\log n)$ approximation. The last result was a small but simple combinatorics result which could prove useful in other settings and doesn't appear in the literature to the extent of our knowledge.

The next step would be to prove a bound on approximation ratio of the modified ANS algorithm, and we believe a polyloglog bound might be reachable. Trying other methods to find a constant factor approximation algorithm is also in our goals, and we have a few ideas to test towards this (by forcing the clients to "cooperate" in a certain way, using additional preprocessing). Adapting methods from the Pruning-Clustering Algorithm could also yield this result but so far our efforts with it have been fruitless.

Acknowledgements

I would like to thank my advisor Nicolas Schabanel for his guidance and overall help, E. Hamel for our productive discussions, and C. Malaingre for help with the simulations. I would also like to thank S. Kachanovich and D. Betea for proofreading this report and A. Dahan for help with the LaTeX.

References

[ANS14] Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. *CoRR*, abs/1411.4476, 2014.

[BS12] Babak Behsaz and Mohammad R. Salavatipour. On minimum sum of radii and diameters clustering. In FedorV. Fomin and Petteri Kaski, editors, *Algorithm Theory - SWAT 2012*, volume 7357 of *Lecture Notes in Computer Science*, pages 71–82. Springer Berlin Heidelberg, 2012.

[Chv83] V. Chvátal. *Linear Programming*. Series of books in the mathematical sciences. W.H. Freeman, 1983.

[CP04] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci.*, 68(2):417–441, 2004.

[DS13] Irit Dinur and David Steurer. Analytical approach to parallel repetition. *CoRR*, abs/1305.1979, 2013.

[EMS14] David Eisenstat, Claire Mathieu, and Nicolas Schabanel. Facility location in evolving metrics. *CoRR*, abs/1403.6758, 2014.

[GK99] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, pages 228–248, 1999.

[Hoc82] DoritS. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[KH76] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. In *Mathematical Models in Marketing*, volume 132 of *Lecture Notes in Economics and Mathematical Systems*, pages 406–407. Springer Berlin Heidelberg, 1976.

[Li13] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45 – 58, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[LS13] Yin Tat Lee and Aaron Sidford. Matching the universal barrier without paying the costs : Solving linear programs with õ(sqrt(rank)) linear system solves. *CoRR*, abs/1312.6677, 2013.

[MYZ06] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM Journal on Computing*, 36(2):411–432, 2006.

[Vyg05] Jens Vygen. Approximation algorithms for facility location problems. Lecture Notes, 2005.

5 Appendix

5.1 Preprocessing

5.1.1 Determining when client change facilities

This is a classic preprocessing that was introduced in [EMS14] and used again in [ANS14]. We start with a solution (x, y, z) to the LP and want to output another one $(\overline{x}, \overline{y}, \overline{z})$ such that the total cost is at most twice as much and such that for each client j we can compute a division of $\mathcal{T}$ in a set of intervals satifsying the following:

- For the duration of each interval all the $\overline{x}^t_{i,j}$ are constant
- The number of intervals is at most $2 \times \sum z^t_{ij}$

To this end we set $t^j_k = 1$ and $a = 1$ and we are looking iteratively for the biggest $t \in (t^j_{k-1}, T+1]$ such that t

$$\sum_{i \in F} \left(\min_{t^j_{k-1} \leq u \leq t} x^u_{ij} \right) \geq \frac{1}{2}$$

If $t^j_k = T + 1$ we stop here, if not we increment a and create a new interval the same way. Now for each interval we can set each $\overline{x}^t_{ij}$ to $2 \times \min_{t^j_{k-1} \leq u \leq t_k} x^u_{ij}$, which guarantees that $\sum_{i \in F} \overline{x}^u_{ij} \geq 1$. By setting each $\overline{y}^t_i = 2 \times y^t_i$ we also make sure that $\overline{x}^t_{ij} \leq \overline{y}^t_i$.

Moreover, it is easy to see that if between t^j_k and t^j_{k+1} we have $\sum_{i \in F} \left(\min_{t^j_{k-1} \leq u \leq t} x^u_{ij} \right) \leq \frac{1}{2}$, it means that $\sum_{t \in (t^j_k, t^j_{k+1}]} z^t_{ij} \geq \frac{1}{2}$. This in turn means that for each interval the initial solution paid at least $\frac{1}{2}g$, and here we pay at most g (to completely change all the x^t_{ij} between one interval and the next).

Hence both the changing cost and the facility opening cost are at most multiplied by 2 to achieve the property, and this algorithm works in linear time in its input.

5.1.2 Forcing clients to have the same mass on facilities

This preprocessing is similar to the one used in [ANS14] and creates virtual copies of each facility to have the following property: for each i and each r there is an $o_{ir} \in [0, 1]$ such that for all j, t, $x^t_{ij} \in \{0, o_{ir}\}$ with $d(i, j) \leq r$.

First we create an additional facility with a different radius for each possible i, r and t. We greedily assign the j to the new facility of smallest radius possible with mass x_{irj} such that $x_{irj} \leq y_{ir}$, and then to the next radius until $\sum_r x_{irj} = x_{ij}$.

Then for each of the new facilities i, r (which now have a fixed radius) we create at most one new virtual facility per client attached to i, such that the first has mass $\beta_0 = \min_{j: x_{irj} > 0} x^t_{irj}$, the second mass $\beta_1 = \min_{j: x_{irj} - \alpha_0 > 0} x^t_{irj} - \beta_0$ and so on, and by assigning clients greedily to them so that each client is assigned with mass either β_j or 0.

If this is done after the first preprocessing we have the properties required, with a number of facilities at most multiplied by Tn^2.

5.2 Forcing the LP to find a mixed solution

If we give the structure T_n to the LP solver, even though it could give any mix of pure solutions (that is, consisting of a single level), it probably wouldn't as most solvers output a vertex of the polytope of solutions, and all mixed solutions are on the interior of the optimal facet of the polytope. We must then modify it slightly to add an incentive to be in the center of the facet, hence to have each facility in T_n be taken with probability $\frac{1}{n}$.

We now take an instance where $f_i = 0$ for all i, $g = 0$ and the time steps are separated in two groups. In the first one we have a modified structure T_n, and in the second we have a multitude of steps meant to force each client to attach to each of its facilities with mass $\frac{1}{n}$.

To do this consider the construction where you have an $(n+1)-$ simplex of side r, with clients at the vertices and facilities at the center of each facet (as there are $n+1$ facets we needed the additional facility). The optimal assignment here is easy to compute because if we take each facility with mass $\frac{1}{n}$ and radius $\frac{1}{2}$ all the clients are covered by n facilities, hence are completely covered and the total cost is $\frac{r}{2} \times \frac{n+1}{n}$. It is easy to see that this is optimal because if any facility were open with radius r and mass α the contribution to all clients would be α and by simply paying $\frac{\alpha}{n}$ on each facility we would have the same coverage with cost $\frac{r\alpha(n+1)}{2n}$ instead of $r\alpha$.

The instance then goes as follows: we have one last time step with the structure T_n where instead of each client there is a set of $n+1$ new clients and where we have the new set of facilities (one per group of clients) very far from T_n.

Before that we have 2^n time steps in which all the clients and facilities are in the same place at the center of the space, except for one client set and the $n+1$ corresponding facilities (the n the initial client was connected to, plus the one we added for the set).

At each of those time steps, the clients are the vertices of a simplex of side r at the center of the space and the facilities are at the center of each facet.

The facilities we added per group of client have mass $\frac{1}{n}$ until the last step where they all decrease to 0 and then have no influence on the last time step.

By making r small enough (2^{-2n}) we can make sure that the cost increase is a fraction of the initial cost, which also means that any client can only put mass on its corresponding initial n facilities. By construction the optimal for each client is to put mass $\frac{1}{n}$ on each of its initial facilities, and as this is still an optimal solution for T_n this is the only optimal solution for the whole LP instance. We then have an instance where the solvers output an assignment where each facility has mass $\frac{1}{n}$ as we wanted, and where we multiplied by n the number of clients, by $2+\epsilon$ the number of facilities and where we have $2^n + 1$ time steps instead of 1.

5.3 Better bound for Theorem 2

We recall that for each interval the probability that a client is covered during the interval is greater than $1 - (1 - \frac{1}{n})^n$, which is decreasing towards $1 - e^{-1}$.
By repeating the algorithm k times we then have a probability at most $\left(1 - e^{-1}\right)^k$ of each interval not being covered. If we set $k = \log_2(Z) \times \log_2(1 - e^{-1})$, the probability of each interval not being covered becomes $\frac{1}{2Z}$. Using the union bound, the probability of all clients being covered in k rounds is at least $\frac{1}{2}$. Then for any $\varepsilon > 0$, by using Markov's inequality, the probability of the total cost being more than $(2+\varepsilon)k \times OPT$ becomes $\frac{1}{2+\varepsilon}$.
This means that with probability $\frac{\varepsilon}{4+2\varepsilon}$ we get a solution with cost at most

$$(2+\varepsilon)\log_2(1 - e^{-1}) \times \log_2(Z) \times OPT$$

or alternatively $\beta \times \ln(Z) \times OPT$ with, for small enough ε

$$\beta = (2+\varepsilon)\log_2\left(\frac{e}{e-1}\right) \times \log_2(e) < 1.93$$

5.4 Better bounds for Lemma 1

5.4.1 By using integration

We know that the probability of finding a path of length $n/2$ is at least $p(x,n,h,\varepsilon) \geq \frac{2\alpha-1}{\beta^2}$, and the probability of finding the second half is then at least $\left(\frac{2\alpha-1}{\beta^2}\right)^2$ when we have

$$\frac{1}{2} \leq \alpha \leq \frac{x}{2^n} - \frac{5}{2}\varepsilon \leq \frac{x}{2^n} + \varepsilon \leq \beta \leq 1$$

As the probability of x being in the bounds is $(\beta - \alpha - 4\varepsilon)$, the proof just looked at good bounds for α and β to maximize $\left(\frac{2\alpha-1}{\beta^2}\right)^2 (\beta-\alpha-4\varepsilon)$. However we can instead “integrate” $\left(\frac{2x-1}{x^2}\right)^2$ between $\frac{1}{2}$ and 1. To do this we consider a finite sequence (α_i), with $\alpha_0 = \frac{1}{2}$ and $\alpha_{i+1} - \alpha_i = \gamma = 2^{-n/3}$. Then the probability of x being between α and β is at least $\gamma - 4\varepsilon$, and the probability of there being a path from root to leaf is at least

$$\sum_{0 \leq i < \left\lfloor \frac{1}{\gamma} \right\rfloor} \left(\frac{2\alpha_i - 1}{\alpha_{i+1}^2}\right)^2 \times (\gamma - 4\varepsilon)$$

which is equal to

$$\sum_{0 \leq i < \left\lfloor \frac{1}{\gamma} \right\rfloor} \left(\frac{2\alpha_{i+1} - 1 - 2\gamma}{\alpha_{i+1}^2}\right)^2 \times (\gamma - o(\gamma))$$

and at least

$$\left(\sum_{0 \leq i < \left\lfloor \frac{1}{\gamma} \right\rfloor} \left(\frac{2\alpha_{i+1} - 1}{\alpha_{i+1}^2}\right)^2 \times \gamma \right) - O(\gamma)$$

As this is a Riemann sum and admits a limit, when n (and $\frac{1}{\gamma}$) increases it tends towards

$$\int_{1/2}^{1} \left(\frac{2x-1}{x^2}\right)^2 dx = \frac{1}{3}$$

As we said earlier the probability of there being a path between root and leaf decreases with n, so if the probability of there being such a path were at some point strictly less than $\frac{1}{3}$ it would then stay under that value which is impossible as it tends towards $\frac{1}{3}$. The probability is then at least $\frac{1}{3}$.

5.4.2 By examining the distribution at height $n/2$

If we look at what happens at height $n/2$ it turns out that we more often have two paths from root to height $n/2$ than one – in practice either we should have many or none, depending on the value of the root. More precisely, if X is the number of paths at height $\frac{n}{2} - 1$, the number of paths at height $n/2$ is $\sum_{i=1}^{X} Y_i$ where the Y_i behave like i.i.d. random variables corresponding to the probability of finding either 0, 1 or 2 leaves smaller than the root. They are not truly independent but we can control all the dependences with an ε correcting factor. If we temporarily ignore this factor we can notice that by only considering the first i such that Y_i is not equal to zero we have a probability $\sim \frac{x}{2-x}$ to have $Y_i = 2$ and a probability $\sim 1 - \frac{x}{2-x}$ to have $Y_i = 1$. Hence the probability of having at least 2 paths from root to nodes at level $n/2$ is at least $\left(\frac{2x-1}{x^2}\right)\left(1 - \frac{x}{2-x}\right)$.

If we input this in the previous expression, we finally get a probability of getting a path through T at least equal (modulo ε factors) to

$$\int_{1/2}^{1}\left(\frac{2x-1}{x^2}\right)\left(\left(\frac{2x-1}{x^2}\right)\left(1-\frac{x}{2-x}\right)+\left(1-\left(1-\left(\frac{2x-1}{x^2}\right)\right)^2\right)\left(\frac{x}{2-x}\right)\right)dx$$

As the ε factors disappear for the same reasons as before, we see that the value[12] of this integral is

$$\frac{2+27\cdot\tanh^{-1}\left(\frac{1}{2}\right)}{48}\approx 0.3506$$

5.4.3 Experimental upper bounds

As said before by doing a detailed case analysis on the tree of height 3, one can get an upper bound of $\frac{39}{70}$. Computing the exact value for height 4 is feasible but prohibitive as there are more than 10^{12} permutations to test. By numerical simulations on trees of height 4 to 10 we get the following (for 10 batches of 10^7 tests):

Height	4	5	6	7	8	9	10
Empirical probability	0.504	0.473	0.453	0.441	0.431	0.423	0.418
maximal difference within the 10 batches	0.0023	0.0008	0.0018	0.0015	0.0013	0.0021	.0017

5.4.4 Empirical efficiency of modified ANS algorithm

Before getting the proof that the algorithm's approximation ratio was $\Omega(\log\log n)$ we ran a simulation on S_n to check if the cost would indeed increase with n. This turned out to be true, even though those results have to be taken with a grain of salt as computing it for big n was not possible so we have a limited number of datapoints. All the code for this can be found at https://github.com/koliaza/ANS-algorithm-counterexample-test.

For the following we used 10 batches of 10^4 runs.

n	3	4	5	6	7	8	9	10	11
Average approximation ratio	1.033	1.126	1.190	1.259	1.330	1.398	1.462	1.523	1.576
Maximal difference	0.145	0.066	0.021	0.011	0.007	0.008	0.008	0.007	0.010

For bigger values of n we only used 10 batches of 10^3 runs as the computation time took a few hours on a i7@3.5GHz. This explains the increased difference between the batches .

n	12	13	14	15	16	17	18	19
Average approximation ratio	1.635	1.692	1.737	1.759	1.766	1.768	1.775	1.773
Maximal difference	0.034	0.039	0.030	0.035	0.037	0.047	0.030	0.040

[12] Checked with a computer algebra system (WolframAlpha),