Exact Exponential Algorithms for the Center Closing Problem

Samo Metličar
samo.metlicar@hotmail.com
Faculty of Computer and
Information Science,
University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia

Jurij Mihelič
jurij.mihelic@fri.uni-lj.si
Faculty of Computer and
Information Science,
University of Ljubljana
Večna pot 113
SI-1000 Ljubljana, Slovenia

ABSTRACT

In this paper, we focus on the center closing problem which is similar to the well-known k-center problem. Both problems are defined on a network with the goal of optimizing the worst-case service time for the clients, but with the difference that in the center closing problem several existing centers are closed in order to optimize total cost of operation. First, we show the $\mathcal{NP}$-hardness of the problem. Afterwards, we describe several exact exponential algorithms for solving the problem. Finally, experimentally evaluate these algorithm on two test scenarios.

KEYWORDS

center closing problem, exact algorithm, combinatorial optimization, experimental evaluation, set cover

ACM Reference Format:
Samo Metličar and Jurij Mihelič. 2022. Exact Exponential Algorithms for the Center Closing Problem. In *Proceedings of Student Computing Research Symposium (SCORES'22).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Facility location problems deal with the optimality of the placement of objects into space. Its roots stretch at least back to the 17th century in the form of Fermat's problem [5] and it has since branched out significantly. In this paper, we discuss one such problem – the center closing problem. We define it and its main properties and propose several exact algorithms for solving it. Those are also implemented and tested using different benchmarks.

The problem deals with closing a set number of centers (e.g. post offices) that offer some service to consumers (e.g. towns) and is represented by a network (e.g. road network).

A lot of other prominent facility location problems can also be found in the literature. One such example is the k-suppliers problem, analyzed by Hochbaum and Shmoys [6]. A similar problem where the metric is limited to the L_2 space is examined by the authors of [11]. There are also more complex variations of the problem where objects are more dynamic and can move or change over time [7, 8]. Such problems are often solved with integer linear programs instead of algorithmically.

SCORES'22, October 6, 2022, Ljubljana, Slovenia
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 PRELIMINARIES

We begin by introducing basic concepts used in the paper. The center closing problem first requires a network, which we define as a graph with weighted edges. For each pair of nodes u and v the length of the shortest path between them is denoted by $d(u, v)$. For each node v in the network and an arbitrary bound B we also define its bounded open neighbourhood $N_B(v)$ as a set of nodes connected to v by a path of length at most B, excluding v itself. Finally, let S be a set. We say that the family of sets $\mathcal{P}$ forms a partition of set S if the following statements hold:

- $\emptyset \notin \mathcal{P}$,
- $\bigcup_{A \in \mathcal{P}} A = S$ in
- $\forall A, B \in \mathcal{P} \colon A \neq B \implies A \cap B = \emptyset$.

From this the formal definition of the problem can follow.

Definition 2.1 (Center closing problem). Let $G = (V, E)$ be a complete undirected network, sets of centers S and consumers C form a partition of V, $h : C \to \mathbb{R}_0^+$ be a non-negative weight function and k be a positive integer not greater than $|S|$. For every subset $R \subseteq S$ define the cost as

$$\text{cost}(R) = \max_{c \in C} \min_{s \in S \setminus R} h(c) d(c, s).$$

The problem is to find such a set $R \subseteq S$, where $|R| \geq k$, which minimizes the cost.

The problem can also be presented in the decision version by adding a bound B and asking if there exists a solution of cost at most B.

3 HARDNESS OF THE PROBLEM

Because of the similarity to the k-center [10] problem and other facility location problems we assume that it is also $\mathcal{NP}$-hard. We prove the assumption by reducing the k-center problem to the center closing problem in polynomial time. Let $G' = (V, E)$ be a complete undirected network and k' be a non-negative integer not greater than $|V|$. The k-center problem is to find a set $P \subseteq V$, where $|P| \leq k'$, which minimizes the cost

$$\text{cost}(P) = \max_{v \in V} \min_{p \in P} d(v, p).$$

We reduce the k-center problem to the center closing problem as follows:

- network G contains 2 copies of each node in V and sets the weight of the edge connecting them to 0. The rest of the edges are copied from E,
- $S = C = V$, ie., the set of nodes represents both the set of centers and the set of consumers,
- $h = \mathbb{1}_C$, ie., all consumer weights are 1, and
- $k = |S| - k'$, ie., the number of closed centers as opposed to the number of opened centers.

Lemma 3.1. *Set P^* solves the k-center problem if and only if set $R^* = S \setminus P^*$ solves the center closing problem.*

Proof. Let us denote by cost_1 and cost_2 the cost functions of the k-center problem and the center closing problem. Let P be some (not necesserily optimal) solution to the k-center problem and $R = S \setminus P$. Then it holds that $|P| \leq k'$ and $|R| \geq k$ and R is thus also a solution to the center closing problem.[1] We now observe the cost functions:

$$\begin{aligned}\text{cost}_2(R) &= \max_{c \in V} \min_{s \in V \setminus R} \mathbb{1}_V(c) d(c, s) \\ &= \max_{v \in V} \min_{p \in P} d(v, p) \\ &= \text{cost}_1(P).\end{aligned}$$

Lets say there exists $R' \neq R^*$ s.t. $\text{cost}_2(R') < \text{cost}_2(R^*)$. Since the costs are equivalent it follows that $\text{cost}_1(V \setminus R') < \text{cost}_1(P^*)$. Therefore P^* does not solve the k-center problem – a contradiction. □

We have shown how to reduce the k-center problem to the center closing problem in polynomial time. We have also shown that the cost functions match in both problems, therefore the decision versions of them can use the same boundary and will return equivalent results. From [9] it then follows that since the decision version of the k-center problem is $\mathcal{NP}$-complete, this must also hold for the decision version of the center closing problem. Further it follows from [12] that since the decision version of the center closing problem is $\mathcal{NP}$-complete, its optimization version must be $\mathcal{NP}$-hard.

4 EXACT EXPONENTIAL ALGORITHMS

4.1 Brute-force Enumeration

The most straightforward approach to solving the problem is generating all subsets R of set S. Let $n = |C|$, $m = |S|$, and $k = |R|$, then there are $\binom{m}{k}$ possible subsets in total. Each has to be evaluated resulting in the time complexity $T_k(n, m) = \binom{m}{k}(m - k)n$. For a fixed k this results in a polynomial time of $O(m^{k+1}n)$, however in the worst case scenario where $k = \frac{m}{2}$ the time complexity is exponential, namely $O(2^m \sqrt{m} n)$ (derived using the Stirling formula). This algorithm will later be referred to with the abbreviation `bf`.

4.2 Backtracking

Here, we represent the search space of the problem with a rooted binary tree, where each inner node represents a set of closed centers and outgoing connections represent available actions – the center can be closed (the left branch) or kept open (the right branch). Leaf nodes represent solutions and are, thus, evaluated. The algorithm, denoted with `bt`, traverses the tree using depth-first search introducing two pruning methods:

- if there are not enough centers left in the branch to fulfill the minimum required number of closed centers, the branch is pruned;
- if there are already enough centers closed, the branch below the current node is pruned. The currently visited node becomes a leaf.

The backtracking approach was also implemented in the algorithm, denoted with `btd`, that solves the decision version of the problem allowing for an additional pruning method. In particular, by tracking intermediate costs at each node, we can take advantage of the fact that those costs are non-decreasing and prune the branch if the intermediate cost exceeds the boundary B.

4.3 Branch and Bound

We upgrade the backtracking algorithm with the *branch and bound* strategy, denoted with `bnb`. During the *branching* part of the algorithm, a priority queue is maintained determining the order in which centers are processed. On each step, the intermediate solution is evaluated enabling the use of the *bound* part of the strategy. It takes advantage of the fact that the cost is non-decreasing with respect to closing additional centers. Therefore, if the intermediate evaluation ever exceeds the current lowest cost found, the whole branch can be pruned.

The algorithm starts without an upper bound, meaning it might not prune much until a decent upper bound is determined. To address this, the algorithm first solves the problem with an approximate algorithm and uses that solution as the starting bound. This alteration of the algorithm will be abbreviated with `bnb+`.

4.4 Reduction to the Set Cover Problem

Here, we develop an algorithm based on the reduction to the decision version of the set cover problem, which is defined as follows. Let U be a universal set, $\mathcal{S}$ be a family of subsets of U, and $\ell \in \mathbb{N}$ a boundary. The question is whether there exists such a family $\mathcal{M} \subseteq \mathcal{S}$ of size $|\mathcal{M}| \leq \ell$, that covers the universal set U, i.e., $\bigcup_{M \in \mathcal{M}} M = U$. We reduce the decision version of the center closing problem to the set cover problem as follows:

- $U = C$, i.e., the set of suppliers represents the universal set,
- $\mathcal{S} = \{N_B(s) \mid s \in S\}$, i.e., the family consists of sets of neighbors of the existing centers, and
- $\ell = m - k$, i.e., the bound for the set cover problem.

Lemma 4.1. *Family $\mathcal{M}^* \subseteq \mathcal{S}$ solves the set cover problem if and only if subset $R^* \subseteq S$, where $\{N_B(s) \mid s \in S \setminus R^*\} = \mathcal{M}^*$, solves the center closing problem.*

Proof. ($\Rightarrow$) Let $S^* = \{s \in S \mid N_B(s) \in \mathcal{M}^*\}$. Clearly, $S^* \subseteq S$ and by the definition of the set cover neighborhoods of selected centers cover the whole set C, meaning the cost is at most B. From $R^* = S \setminus S^*$ it also follows that $|R^*| \geq k$ and thus R^* is a valid solution.

($\Leftarrow$) Let $S^* = S \setminus R^*$. For every $c \in C$ it holds that there exists a center $s \in S^*$ s.t. $h(c)d(c, s) \leq B$, meaning $c \in N_B(s)$. Because

[1] This also holds in reverse where R is a solution to the center closing problem and $P = V \setminus R$.

$|S^*| \le m - k = \ell$, the family $\mathcal{M}^* = \{N_B(s) \mid s \in S^*\}$ is a valid solution. $\square$

Now, we introduce the set cover algorithm, see Algorithm 1, that is derived from [13] where its correctness as well as its time complexity of order $O(1{,}5169^m)$ is shown.

Algorithm 1 Set Cover

```
Input: Universal set $U$, family $\mathcal{S}$ and bound $\ell$
Output: YES or NO
  procedure SetCover($U, \mathcal{S}, \ell$)
    if $\ell = 0$ then
      if $\mathcal{S} = \emptyset$ and $U = \emptyset$ then return YES
      else return NO
      end if
    end if
    $S \leftarrow \arg\max\{|S'| \mid S' \in \mathcal{S}\}$.
    if $\exists u \in U$ s.t. it is contained in exactly one set $R \in \mathcal{S}$ then
      return SetCover($U \setminus R$, $\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\}$, $\ell - 1$)
    else if $\exists Q, R \in \mathcal{S}$ s.t. $R \subseteq Q$ then
      return SetCover($U$, $\mathcal{S} \setminus \{R\}$, $\ell$)
    else if $|S| \le 2$ then
      Let $C$ be the set cover computed in polynomial time by using maximum matching
      if $|C| \le \ell$ then return YES
      else return NO
      end if
    end if
    return SetCover($U \setminus S$, $\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\}$, $\ell - 1$) or SetCover($U, \mathcal{S} \setminus \{S\}, \ell$)
  end procedure
```

The algorithm can easily be expanded to solve the optimization version of the problem. To do this, we take into account that the cost of the center closing problem must always be equal to the cost between some consumer and center. All unique costs are therefore ordered allowing for bisection in the ordered list to be used to find the lowest cost for which the solution of the problem exists.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setting

Algorithms were implemented in `C` programming language and compiled using `gcc 10.2.0` with flags `-O3 -std=gnu17`. Tests were run on a computer running `Windows 10` on an `AMD Ryzen 5 2600x` cpu with a frequency of 3,6 GHz. It has 6 cores with a total of 576 KB L1 cache, 3 MB of L2 cache and 16 MB of L3 cache. The system has 16 GB of available DDR4 RAM with a frequency of 2966 MHz. In the next two sections, we present results of the experimental evaluation on two different test scenarios.

5.2 Scale-free Graphs

Here, we consider a test scenario consisting of scale-free networks which are common in the real world. Here, the Barabási–Albert model [1] is used to generate random base graphs. The m nodes with the highest degree are selected as the centers while the rest become consumers. Distances between consumers and centers are set as the length of the shortest path between the two. Additionally, the consumer weight is determined by its degree. Generating random networks allowed us to isolate input parameters and test them independently of each other.

First, we focus on the the number of closed centers k in order to determine its effect on the time complexity for each algorithm. For this, 100 random scale-free networks were generated with $n = 100$ and $m = 20$. The results, shown in Figure 1, show that most algorithms are behaving similarly to what we derived in Section 4.1, meaning they are the slowest around $k = \frac{m}{2}$. The only exception to this observation is the `sc` algorithm which performs the best for high values of k and the worst for low values.

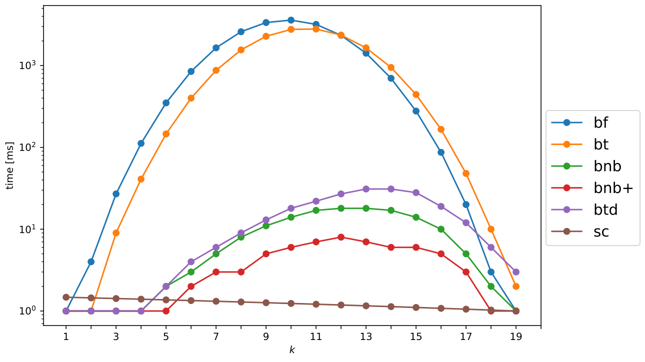

Figure 1: Average execution times for the scale-free networks using exact algorithms in relation to k.

Next, we focus on the total number of centers m. For this benchmark 100 networks with $n = 100$ consumers were generated for each value of m. Algorithms were closing $k = 3$ centers and the results can be seen in Figure 2. Observe that several algorithms, such as bnb, are outperforming the set cover algorithm quite drastically. This coincides with the observation from the previous test, where set cover algorithm performs poorly for low values of k while the rest of the algorithms exceed in that range.

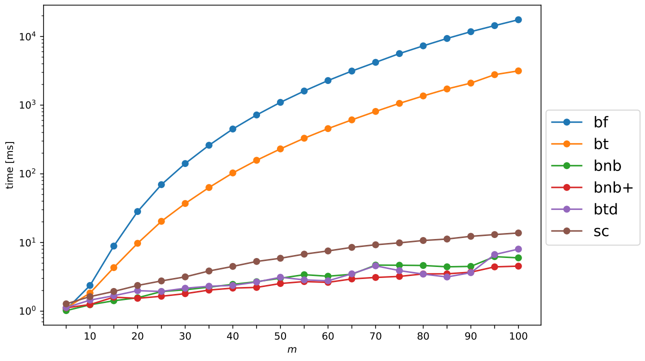

Figure 2: Average execution times for the scale-free networks with $n = 100$ consumers using exact algorithms for parameter $k = 3$ in relation to m.

Finally, the number of consumers n was isolated. 100 networks were generated for each value of n, all with $m = 20$ centers. The number of closed centers was set to $k = 3$. The results, shown in figure 3, indicate that the impact of parameter n on the time complexity is linear for all observed algorithms. This once again coincides with the calculations from 4.1.

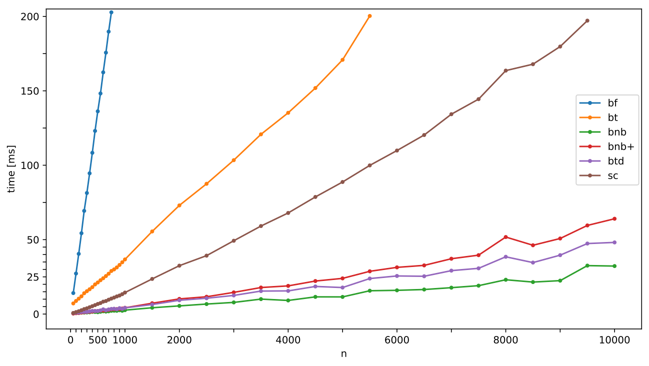

Figure 3: Average execution times for the scale-free networks with $m = 20$ and $k = 3$ using exact algorithms in relation to n.

#	n, m	k	opt	time	#	n, m	k	opt	time
1	100	95	127	426	21	500	495	40	
2	100	90	98	387	22	500	490	38	
3	100	90	93	2503	23	500	450	22	
4	100	80	74	18	24	500	400	15	948
5	100	67	48	22	25	500	333	11	1351
6	200	195	84	26695	26	600	595	38	
7	200	190	64	43614	27	600	590	32	
8	200	180	55	14321	28	600	540	18	
9	200	160	37	97	29	600	480	13	1377
10	200	133	20	132	30	600	400	9	1857
11	300	295	59	2365	31	700	695	30	
12	300	290	51	563473	32	700	690	29	
13	300	270	36	4930	33	700	630	15	
14	300	240	26	291	34	700	560	11	2923
15	300	200	18	297	35	800	795	30	
16	400	395	47	644	36	800	790	27	
17	400	390	39		37	800	720	15	
18	400	360	28		38	900	895	29	1707288
19	400	320	18	11966	39	900	890	23	
20	400	267	13	748	40	900	810	13	

Table 1: Table showing case parameters, optimal solution and solving time in ms for the set cover algorithm on test cases `pmed` from `OR-Library`.

5.3 Pmed Benchmark Library

One of the widely used benchmarking libraries to test algorithms for solving facility location problems is `OR-Library` [3]. It contains the set, denoted with `pmed`, of 40 test cases [2] for the p-median problem [4] which also became the standard for testing the algorithms for k-center problem [10], from which the center closing problem originates.

Test cases contain networks with nodes counting from 100 to 900. Graphs were originally generated by adding $|V|^2/50$ edges at random with discrete uniformal distribution of lengths from the interval $[1, 100]$.

Algorithms were tested by reducing the k-center problem to the center closing problem. In turn this meant that $C = S = V$ and the high number of centers caused most algorithms to be too slow. This was futher exagarated by the fact that tests required a high amount of those to be closed. This meant that only the set cover algorithm was able to solve some of the tests, which also then serve as an additional test of correctness of the algorithm, since the solutions to all 40 test cases are publicly available.

The algorithm was given 30 minutes for each test case to find the solution. Solve times as well as the solutions can be seen in Table 1. We observe that for large m solutions are rare as expected from Figure 2 as well as from the theoretical time complexity.

6 CONCLUSIONS

In this paper we have presented the center closing problem. We proved it is $\mathcal{NP}$-hard and presented several exact algorithms with different approaches to solving the problem. We showed how to reduce a well known k-center problem to center closing problem as well as how to reduce the latter to the set cover problem. Finally, in the paper we have shown the results of experimental evaluation of discussed algorithms tested on different types of networks.

REFERENCES

[1] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74 (Jan 2002), 47–97. Issue 1. https://doi.org/10.48550/arXiv.cond-mat/0106096
[2] J.E. Beasley. 1985. A note on solving large p-median problems. *European Journal of Operational Research* 21, 2 (1985), 270–273. https://doi.org/10.1016/0377-2217(85)90040-2
[3] J.E. Beasley. 1990. *OR-LIBRARY.* http://people.brunel.ac.uk/~mastjjb/jeb/info.html
[4] Mark S. Daskin and Kayse Lee Maass. 2015. *The p-Median Problem.* Springer International Publishing, Cham, 21–45. https://doi.org/10.1007/978-3-319-13111-5_2
[5] Heinrich Dörrie. 1965. *100 Great Problems of Elementary Mathematics: Their History and Solution.* Dover Publications, Inc., 361–363.
[6] Dorit S. Hochbaum and David B. Shmoys. 1986. A Unified Approach to Approximation Algorithms for Bottleneck Problems. *J. ACM* 33, 3 (may 1986), 533–550. https://doi.org/10.1145/5925.5933
[7] Sanjay Dominik Jena, Jean-François Cordeau, and Bernard Gendron. 2015. Modeling and solving a logging camp location problem. *Annals of Operations Research* 232 (2015), 151–177. https://doi.org/10.1007/s10479-012-1278-z
[8] Sanjay Dominik Jena, Jean-François Cordeau, and Bernard Gendron. 2016. Solving a dynamic facility location problem with partial closing and reopening. *Computers & Operations Research* 67 (2016), 143–154. https://doi.org/10.1016/j.cor.2015.10.011
[9] D. S. Johnson M. R. Garey. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness* (first edition ed.). W. H. Freeman. https://doi.org/10.1137/1024022
[10] Jurij Mihelič and Borut Robic. 2005. Solving the k-center Problem Efficiently with a Dominating Set Algorithm. *CIT* 13 (09 2005), 225–234. https://doi.org/10.2498/cit.2005.03.05
[11] Viswanath Nagarajan, Baruch Schieber, and Hadas Shachnai. 2013. The Euclidean k-Supplier Problem. In *Integer Programming and Combinatorial Optimization*, Michel Goemans and José Correa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 290–301.
[12] B. Robič. 2009. *Aproksimacijski algoritmi.* Fakulteta za računalništvo in informatiko.
[13] Johan M.M. van Rooij and Hans L. Bodlaender. 2011. Exact algorithms for dominating set. *Discrete Applied Mathematics* 159, 17 (2011), 2147–2164. https://doi.org/10.1016/j.dam.2011.07.001