

A heuristic for the α -neighbor p -center problem

Eduardo Salazar, Roger Z. Ríos

Department of Mechanical and Electrical Engineering
Universidad Autónoma de Nuevo León
San Nicolás de los Garza, N.L.

X Congreso de la Sociedad Mexicana de Investigación de Operaciones
October 19th, 2021

Outline

α -Neighbor p -Center Problem (α N p CP)

Problem formulation

Generalization of the classical p -center problem (p CP), in which the goal is to select p centers such that the maximum distance of a non-center point to its α -th nearest center is minimized.

Note that setting $\alpha = 1$ corresponds to the p CP.

α -Neighbor p -Center Problem (α NpCP)

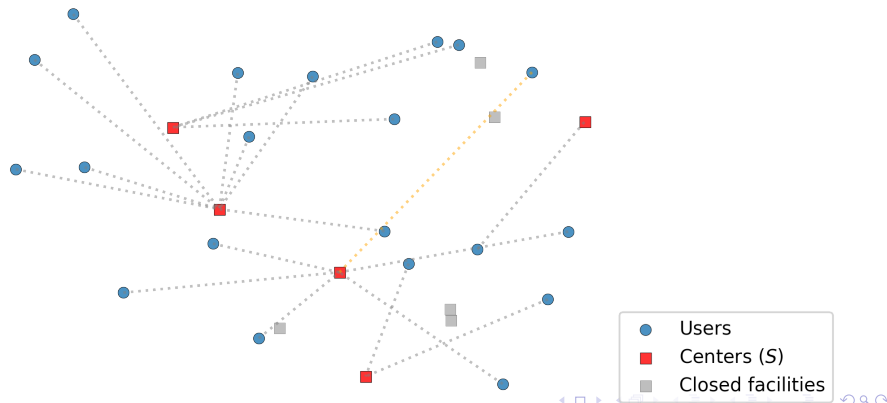
Problem formulation

$$n = 20$$

$$m = 10$$

$$p = 5$$

$$\alpha = 2$$



α -Neighbor p -Center Problem (α NpCP)

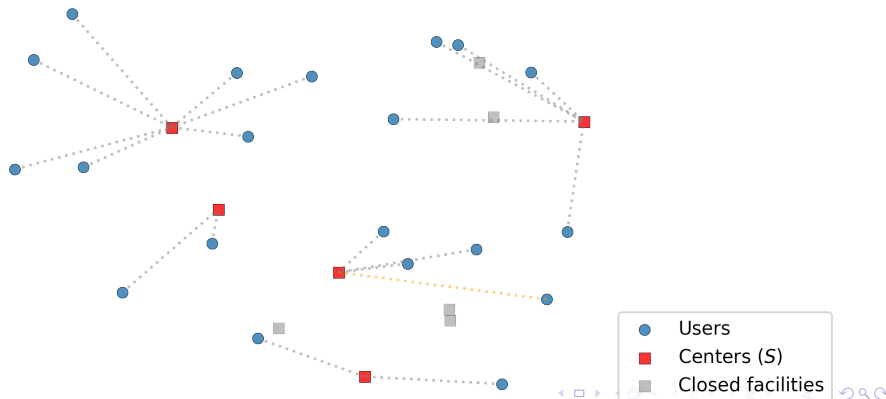
Problem formulation

$$n = 20$$

$$m = 10$$

$$p = 5$$

$$\alpha = 1 \text{ (PCP)}$$



Combinatorial model

Problem formulation

- Sets and parameters
 - $V = \{1, 2, \dots, n\}$, set of nodes
 - d_{ij} = distance between nodes i and j , $i, j \in V$
 - α = position of closest assigned center for all $v \in V \setminus P$
- Variables sets
 - $P = \{c_1, c_2, \dots, c_p\}$, set of centers

Combinatorial model

Problem formulation

Objective

$$\min_{P \in \Pi} \max_{v \in V \setminus P} \min_{S \subseteq P, |S|=\alpha} \max_{s \in S} d(s, v)$$

$$2 \leq \alpha \leq p \leq |V|$$

where Π is the collection of subsets of V of cardinality p , that is,
 $\Pi = \{P \subseteq V : |P| = p\}$.

Applications



The aim is to have a minimum guaranteed response time between a customer or demand point and its α -th closest center by considering a notion of fault tolerance, i.e., providing backup centers in case one of them fails to respond to an emergency situation.

This ensures that even if up to $\alpha - 1$ facilities fail, every customer has a functioning facility close to it.

The $\alpha NpCP$ has received less attention in the literature compared to the pCP . There are optimal, approximation, and relaxation algorithms. To the best of our knowledge, there's only one heuristic approach for the general uncapacitated $\alpha NpCP$ and it's used to get lower and upper bounds for an optimal algorithm.

Proposed heuristic

Input: $V, d(i, j), p, \alpha, \beta, i_{max}$

Output: $P^* :=$ set of p centers

```
1:  $P^* \leftarrow \emptyset$ 
2:  $f^* \leftarrow \infty$ 
3: while  $i_{max} > 0$  do
4:    $P \leftarrow \text{Construct}(V, d(i, j), \alpha, \beta)$ 
5:    $P \leftarrow \text{LocalSearch}(P)$ 
6:   if  $f(P) < f^*$  then
7:      $f^* \leftarrow f(P)$ 
8:      $P^* \leftarrow P$ 
9:   end if
10:   $i_{max} \leftarrow i_{max} - 1$ 
11: end while
12: return  $P^*$ 
```

A metaheuristic framework with a Greedy Randomized Adaptive Search Procedure (GRASP) using a value-based restricted candidate list (RCL).

Parameters:

- β : Threshold quality parameter
- i_{max} : Number of iterations

Two constructive heuristics were implemented and compared to determine which of them will be used in further experimentation, i.e. as a previous step for a local search improvement or as an initial component of a metaheuristic framework.

Construction phase

Greedy Dispersion (GD)

The first heuristic, called GD, is based on the p -dispersion problem (p DP), which consists in selecting p facilities from a given set of n candidates in such a way that the minimum distance between facilities is maximized. It is of interest here because of how the facilities scatter among clients.

p DP objective function

$$\max_{P \subseteq V} \min_{i,j \in P} d(i,j)$$

$$p \leq |V|$$

The first two facilities are $i^*, j^* \leftarrow \arg \max_{i,j \in V} d(i,j)$.

Then, iteratively choose a node seeking the farthest node to the current solution, $v^* \leftarrow \arg \max_{v \in V \setminus P} \min_{s \in P} d(v,s)$

Construction phase

Greedy Center (GC)

The second heuristic, called GC, evaluates the actual objective function of the $\alpha NpCP$ when choosing the next node to add to the solution.

$\alpha NpCP$ objective function

$$\min_{P \in \Pi} \max_{v \in V \setminus P} \min_{S \subseteq P, |S|=\alpha} \max_{s \in S} d(s, v)$$

$$2 \leq \alpha \leq p \leq |V|$$

`move(s, v)`

Interchange an assigned facility, $s \in P$, by an unassigned facility, $v \in V \setminus P$

Two strategies were coded and tested:

- Heuristic IF: move to the **first** improving neighbor.
- Heuristic IM: move to the **best** improving neighbor.

Experiments layout

Computational results

The heuristic was coded in Python 3.8. The source code can be found in the following repository:

<https://github.com/netotz/alpha-neighbor-p-center-problem/>.

The platform is Intel Core i5 2.3 GHz, 8 GB RAM under Windows 10, using CPython. For the experiments, instances were generated with nodes with random coordinates between 0 and 1000 for both axes.

Constructive algorithms comparison

Computational results

Data set

Size 1 20 instances with 50 total nodes and 5 to be located.

Size 2 20 instances with 400 total nodes and 20 to be located.

Instances			Avg. $f(P)$		Avg. time (s)	
n	p	α	GD	GC	GD	GC
50	5	2	700.4	589.3	0	0.03
		3	843.0	714.6	0	0.01
400	20	2	310.1	330.1	0.06	48.35
		3	381.0	410.5	0.06	38.98

Table: Summary of the experiment results

Based on these results, it was decided to use the p DP heuristic, GD, as the construction step for the GRASP framework.

Local search performance

Computational results

The performance of both strategies of the local search heuristic was tested by trying to improve the solutions found by the constructive heuristic GD.

Instances			Average $f(P)$			Average time (s)			Avg. rel. imp. (%)	
n	p	α	GD	IF	IM	GD	IF	IM	IF	IM
100	5	2	661.9	545.2	549.6	0	0.56	0.56	16.9	16.2
		3	844.8	713.6	707.7	0	0.47	0.56	15.3	16.0
	10	2	421.1	346.5	346.4	0	0.66	0.65	17.3	17.3
		3	543.3	465.1	467.5	0	0.69	0.74	14.2	13.7
	25	2	246.7	227.9	226.9	0.1	123.5	121.0	7.5	7.8
		3	309.4	280.8	279.4	0.1	147.9	151.9	9.2	9.6
500	50	2	185.4	161.0	159.0	0.4	64.7	66.2	12.9	14.0
		3	215.8	192.7	192.2	0.3	97.1	109.8	10.5	10.7

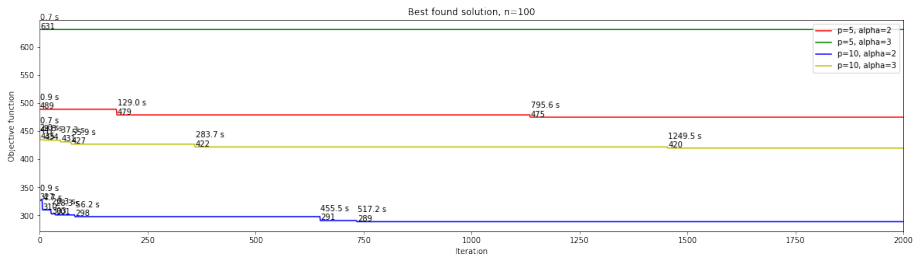
Table: Summary of the experiment results

Both local search strategies clearly improve the solution given by GD. Nevertheless, while time increases significantly, the improvement ratio does not.

Calibrating GRASP iterations

Computational results

In order to gain insight about when the solutions yielded by the GRASP algorithm stop improving, we run an experiment with $i_{max} = 2000$ and $\beta = 0.4$ for 4 instances of size 100.

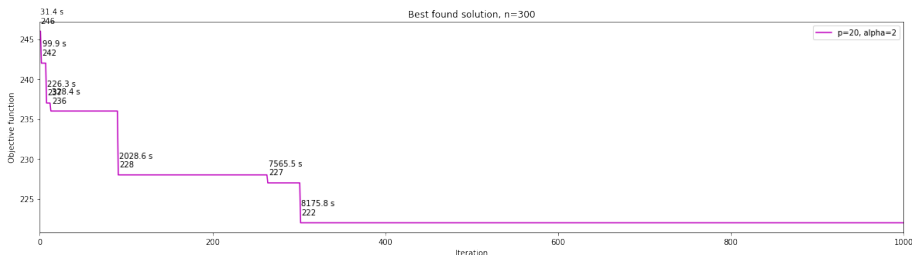


As expected, most of the improvements and the most significant ones occur in the first iterations of GRASP. In general, after 250 iterations the objective function barely improved.

Calibrating GRASP iterations

Computational results

For the same purpose as the previous one, another experiment was run, this time for one instance of size 300 with $i_{max} = 1000$ and $\beta = 0.4$.



Again, the relevant improvements took place in the first iterations. After 200 iterations there were no significant improvements. However, the time taken to reach the last improvement is extremely long.

Conclusions

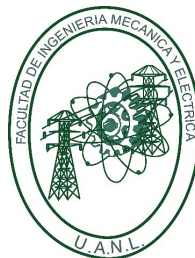
- Two constructive heuristics were proposed and tested
- Heuristics IF and IM showed better behaviour than heuristic GD
- A basic GRASP metaheuristic was presented
- On fine-tuning stage, it was observed that a value of $i_{max} = 250$ was good enough
- GRASP was applied to solve instances of up to $n = 100, p = 10$ in very reasonable running times

Future work





- Test heuristic against a lower bound to assess solution quality by using linear programming relaxation
- Improve the algorithm data structures to reduce running times
- Implement a reactive GRASP strategy
- Improve the metaheuristic by means of path relinking

Acknowledgments

- Thank you very much for your attention!
- Feedback is welcome :)
- Email: neto.otz@hotmail.com
- GitHub: <https://github.com/netotz>



References

-  Krumke, S. O. (1995). On a generalization of the p-center problem. Information processing letters, 56(2), 67-71.
-  Medal, H. R., Rainwater, C. E., Pohl, E. A., & Rossetti, M. D. (2014). A bi-objective analysis of the r-all-neighbor p-center problem. Computers & Industrial Engineering, 72, 114-128.
-  Erkut, E., Ülküsal, Y., & Yenicerioğlu, O. (1994). A comparison of p-dispersion heuristics. Computers & operations research, 21(10), 1103-1113.
-  Chen, D., & Chen, R. (2013). Optimal algorithms for the α -neighbor p-center problem. European Journal of Operational Research, 225(1), 36-43.