A Fast Approximate Algorithm for k-Median Problem on a Graph

Keisuke Todo
Hokkaido University
Sapporo, Japan
todo@ist.hokudai.ac.jp

Atsuyoshi Nakamura
Hokkaido University
Sapporo, Japan
atsu@ist.hokudai.ac.jp

Mineichi Kudo
Hokkaido University
Sapporo, Japan
mine@ist.hokudai.ac.jp

ABSTRACT

We propose a fast approximate algorithm for k-median problem on a graph, which is a problem of finding a set S of k vertices that minimizes the length sum of the shortest paths from all the vertices to their nearest vertex in S. Starting from an initial set S of k vertices, our algorithm iteratively updates S so as to improve the shortest-path-length sum for S. In each iteration, the algorithm calculates the shortest-path forest whose roots are vertices in S and replace S with S' that are the centers of the component tree in the forest. According to our experiments using pmed datasets, our algorithm is significantly faster than CPLEX and achieves better approximation ratio than the degree-centrality or betweenness-centrality based methods.

CCS CONCEPTS

• **Information systems** → *Data mining*; • **Theory of computation** → *Shortest paths*; *Facility location and clustering*;

KEYWORDS

k-median, centrality, graph algorithm, approximate algorithm

ACM Reference Format:
Keisuke Todo, Atsuyoshi Nakamura, and Mineichi Kudo. 2019. A Fast Approximate Algorithm for k-Median Problem on a Graph. In *MLG '19: 15th International Workshop on Mining and Learning with Graphs, August 05, 2019, Anchorage, AK*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

There are many networks in the real world such as communication, co-authorship, and social networks, and such networks can be represented as weighted graphs that are composed of a vertex set, an edge set and an edge weight function, where an edge is a related pair of vertices and the edge weight function indicates strength of the relationship between them. So, finding important vertices on graphs is useful in many applications and a lot of studies on it have been done so far.

The k-median problem is a problem to find k vertices in a graph that are important in terms of the cost of transportation through edges. It is a classical facility location problem with no cost of facility building and generally defined as a problem in a metric space. But the problem can be considered for connected graphs with positive edge length because distance between two vertices provided by the shortest path length between them satisfies the definition of metric. Even for the graph version of k-median problem, it is well-known as NP-hard. The k-median problem on a graph is a problem of finding a set S of k vertices that minimizes the length sum of the shortest paths from all the vertices to their nearest vertex in S. Since the real world networks are becoming larger and larger, solving the problem exactly for such networks is becoming more and more difficult.

In this paper, we propose a fast approximate algorithm for k-median problem on a graph using iterative method. This algorithm initially selects a set S of k vertices at random and then iteratively updates S so as to decrease the length sum of the shortest paths. To update S efficiently, the algorithm calculates the shortest-path forest $\mathcal{F}$ whose roots are vertices in S, then replaces S with S' which are composed of the center of the component trees in $\mathcal{F}$.

According to our experiments using pmed dataset, our algorithm is significantly faster than the exact method using Warshall-Floyd [8] and CPLEX[1], and achieved better approximation ratio than the method based on either degree centrality or betweenness centrality.

2 RELATED WORK

The k-median problem on a graph was proved to be NP-hard by Kariv and Hakimi [11]. By calculating the shortest path length between every pair of vertices, this graph version problem can be converted to the metric version problem, but the calculation takes $O(n^3)$ time and $O(n^2)$ space using Warshall–Floyd algorithm [8], where n is the number of vertices. Thus, for huge graphs, it is hard to solve the graph version problem by converting to the corresponding metric version problem.

The metric k-median problem can be formalized as a 0-1 integer programming problem, and as methods for calculating exact solutions, branch-and-bound algorithms based on Lagrangian relaxation and subgradient optimizations [6, 13] were successfully applied to small-sized problems.

The approximate algorithm with the best guaranteed approximation ratio is the local search with swap [2] for which the approximation ratio of $3 + 2/p$ can be guaranteed while the running time is $O(n^p)$, where p is the number of vertices to be swapped simultaneously.

Approximate algorithms using metaheuristics such as simulated annealing [5] and genetic algorithm [1] have been also developed, but those algorithms are also computationally heavy because the calculation of the length sum of the shortest path to the nearest

MLG '19, August 05, 2019, Anchorage, AK

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

[1] https://www.ibm.com/analytics/cplex-optimizer

vertex in S, must be done for many k-sets S to decide the improved next candidate.

Heuristic approximate algorithm developed by Maranzana [12] repeats partitioning the set V of vertices into $\{V_1, \ldots, V_k\}$, where V_i is the set of vertices whose nearest vertex in the current k-set $S = \{v_1, \ldots, v_k\}$ is v_i, and updating S to S' that is composed of exact 1-medians v'_i of the graphs induced by V_i ($i = 1, \ldots, k$). Our proposed algorithm is similar to but more efficient than this algorithm by calculating not exact but approximate 1-median of each subgraph.

3 k-MEDIAN PROBLEM ON A GRAPH

We consider a connected undirected graph $G = (V, E)$ which is composed of the set of n vertices $V = \{v_1, \ldots, v_n\}$, and the set of edges $E \subseteq V \times V$. An Undirected edge is written as (u, v) which is identical to (v, u). Let $\ell : E \to (0, \infty)$ denote an edge-length function. A path between u and v is an edge sequence $(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), \cdots, (v_{i_{h-1}}, v_{i_h})$ with $v_{i_1} = u$ and $v_{i_h} = v$ and its length is defined as $\sum_{j=1}^{h-1} \ell((v_{i_j}, v_{i_{j+1}}))$. The shortest path between u and v is the path between them with the minimum length, and its length is denoted by $L(u, v)$. Note that $L(u, u) = 0$ for any $u \in V$.

For $S \subseteq V$, define the noncentrality $\mathrm{NC}_G(S)$ of S in G as

$$\mathrm{NC}_G(S) = \sum_{v \in V} \min_{u \in S} L(u, v).$$

Then k-median problem is defined as follows.

Problem 1. *Given a connected undirected graph $G = (V, E)$ with length function $\ell : E \to (0, \infty)$, find the set of k vertices $S(\subseteq V)$ which minimizes its noncentrality $\mathrm{NC}_G(S)$.*

4 ALGORITHM

4.1 Algorithm IDNC

In this paper, we propose an approximate algorithm IDNC (Iteratively Decreasing NonCentrality) that iteratively decreases the noncentrality of current vertex k-set S. The pseudocode of Algorithm IDNC is shown in Algorithm 1. For a given connected undirected graph $G = (V, E)$ with edge length function $\ell : E \to (0, \infty)$, the algorithm initially selects a set S of k vertices at random. Then in each iteration, the set S is updated so as to decrease $\mathrm{NC}_G(S)$ by the following procedure.

First, the shortest path forest of G for the set of roots $S = \{v_{i_1}, \ldots, v_{i_k}\}$, which is composed of k trees $(V_1, E_1), \ldots, (V_k, E_k)$, is calculated by Procedure SPF (Shortest Path Forest), where V_j is the set of vertices in V whose nearest vertex in S is v_{i_j}, and E_i is the set of edges in E that are contained in the shortest paths from v_i to all the vertices v in V_i. Note that SPF is Dijkstra's algorithm [7] for multiple roots. Then, for each component tree (V_j, E_j) ($j = 1, \ldots, k$), the algorithm calculates the vertex v_{i_j} among $v \in V_j$ that minimizes the noncentrality $\mathrm{NC}_{(V_j, E_j)}(\{v\})$, that is, the algorithm solves 1-median problem for tree (V_j, E_j) with edge length function ℓ. The 1-median problem for a tree with n vertices is known to be solvable in $O(n)$ time by Goldman's algorithm [10]. The algorithm stops outputting S when the decrease of $\mathrm{NC}_G(S)$ stops.

For a connected undirected graph with n vertices and m edges, a shortest path forest can be calculated in $O(m + n \log n)$ time and $O(m)$ space by Dijkstra's algorithm, and the calculation of v'_{i_j} at Line 7 takes $O(|V_j|)$ time and space for $j = 1, \ldots, k$ by Goldman's algorithm, thus the Line 6-9 for-loop takes $O(n)$ time and space, where $|V_j|$ is the number of vertices in V_j. Therefore, one iteration of the Line 2-10 repeat-loop takes $O(m + n \log n)$ time and $O(m)$ space. So, Algorithm IDNC runs in $O(r(m + n \log n))$ and $O(m)$ space if the number of executing the repeat-loop is at most r.

Algorithm 1 IDNC(G)

Input: $G = (V, E)$: connected undirected graph with length function $\ell : E \to (0, \infty)$
Output: S: approximate solution of k-median problem

1: $S' \leftarrow$ the set of randomly selected k vertices in V
2: **repeat**
3: $S \leftarrow S'$
4: $(V_1, E_1), \ldots, (V_k, E_k) \leftarrow \mathrm{SPF}(G, S)$
5: $S' \leftarrow \emptyset$
6: **for** $j = 1$ to k **do**
7: $v'_{i_j} \leftarrow \arg\min_{v \in V_j} \mathrm{NC}_{(V_j, E_j)}(\{v\})$
8: $S' \leftarrow S' \cup \{v'_{i_j}\}$
9: **end for**
10: **until** $\mathrm{NC}_G(S) \le \mathrm{NC}_G(S')$
11: **return** S

Procedure SPF(G, S)

Input: $G = (V, E)$: connected undirected graph with length function $\ell : E \to (0, \infty)$
$S = \{v_{i_1}, \ldots, v_{i_k}\} \subseteq V$: set of roots
Output: $(V_1, E_1), \ldots, (V_k, E_k)$: shortest path forest for G with the set of roots S

1: $L_S(v) \leftarrow \infty$ for all $v \in V \setminus S$
2: **for** $j = 1$ to k **do**
3: $V_S(v_{i_j}) \leftarrow j$, $V_j \leftarrow \{v_{i_j}\}$, $E_j \leftarrow \emptyset$
4: **for** $v \in V$ with $(v_{i_j}, v) \in E$ **do**
5: **if** $L_S(v) > \ell((v_{i_j}, v))$ **then**
6: $L_S(v) \leftarrow \ell((v_{i_j}, v))$, $P(v) \leftarrow v_{i_j}$
7: **end if**
8: **end for**
9: **end for**
10: $Q \leftarrow V \setminus S$
11: **while** $Q \neq \emptyset$ **do**
12: $u \leftarrow \arg\min_{v \in Q} L_S(v)$, $Q \leftarrow Q \setminus \{u\}$
13: $j \leftarrow V_S(P(u))$
14: $V_S(u) \leftarrow j$, $V_j \leftarrow V_j \cup \{u\}$, $E_j \leftarrow E_j \cup \{(P(u), u)\}$
15: **for** $v \in Q$ with $(u, v) \in E$ **do**
16: **if** $L_S(v) > L_S(u) + \ell((u, v))$ **then**
17: $L_S(v) \leftarrow L_S(u) + \ell((u, v))$, $P(v) \leftarrow u$
18: **end if**
19: **end for**
20: **end while**
21: **return** $(V_1, E_1), \ldots, (V_k, E_k)$

Remark 1. *The number of iterations r can be upper bounded by ${}_nC_k \le n^k$ due to IDNC's nonincreasing guarantee of noncentrality (Theorem 4.1). Though this upper bound looks too large, r is expected to be very small for real-world datasets. In fact, r is at most 15 for pmed datasets according to our experiment.*

4.2 Guarantee for NC Nonincrease

In each iteration, a set S' of k vertices is calculated from the current set S of k vertices. We can guarantee that $\mathrm{NC}_G(S')$ is always at most $\mathrm{NC}_G(S)$.

Theorem 4.1. *In the Line 2-10 repeat of Algorithm IDNC, Ineq.* $\mathrm{NC}_G(S') \le \mathrm{NC}_G(S)$ *always holds at the end of the repeat (Line 10).*

Proof. Let $S = \{v_{i_1}, \ldots, v_{i_k}\}$ and $S' = \{v'_{i_1}, \ldots, v'_{i_k}\}$ denote the two sets S and S' at Line 10 of Algorithm IDNC. In the repeat, $\mathrm{NC}_G(S) = \sum_{j=1}^{k} \mathrm{NC}_{(V_j,E_j)}(\{v_{i_j}\})$ holds for the shortest path forest $(V_1, E_1), \ldots, (V_k, E_k)$ of G for the set S of roots, which is calculated by SPF(G, S) at Line 4. Since $v'_{i_j} = \arg\min_{v \in V_j} \mathrm{NC}_{(V_j,E_j)}(\{v\})$ by Line 7, $\mathrm{NC}_{(V_j,E_j)}(\{v'_{i_j}\}) \le \mathrm{NC}_{(V_j,E_j)}(\{v_{i_j}\})$ holds for $j = 1, \ldots, k$. It is trivial that $\mathrm{NC}_G(S') \le \sum_{i=1}^{k} \mathrm{NC}_{(V_j,E_j)}(\{v'_{i_j}\})$ holds, so $\mathrm{NC}_G(S') \le \mathrm{NC}_G(S)$ always holds. □

5 EXPERIMENT

5.1 Experimental Settings

We used the pmed dataset in OR-Library [4], which is a synthetic dataset used in [3]. The dataset is composed of 40 undirected graphs with n vertices and $n^2/50$ edges generated by randomly connecting pairs of different vertices for $n = 100, 200, \ldots, 900$. The length of each edge was uniformly generated from $\{1, 2, \ldots, 100\}$. The dataset contains the optimal (minimum) noncentrality of k-median problem for each graph and specific k assigned to the graph, and we can calculate approximation ratio using it. In our experiment, we only use the 5 graphs (pmed16~20) with 400 vertices and 9 graphs (pmed1,6,11,16,21,26,31,35,38) whose assigned k is 5.

All the experiments were conducted using an Ubuntu 16.04.2 LTS OS machine with a Intel(R) Core(TM)2 Quad CPU Q9550 2.83GHz and 4GB memory. We implemented IDNC and Warshall–Floyd (WF) algorithms using the C++ language.

5.2 Results

5.2.1 Number of Iterations. The running time of Algorithm IDNC depends on the number of executing the repeat-loop in Algorithm 1, though a vertex with near optimal noncentrality can not be expected for the case with too few iterations.

We conducted experiments to check frequency distribution of the number of repeat-loop iterations using 5 pmed graphs with 400 vertices by running IDNC 1000 times with different seeds of randomization. The results are shown in Table 1 and 2. The number of the iterations is small: 3.78 ~ 7.74 on average for $k = 5 \sim 133$, so Algorithm IDNC is practically fast. The number of the iterations increases as k increases, but it does not change as the number of vertices n increases.

5.2.2 Approximation Ratio. The approximation-ratio distribution of solutions outputted by Algorithm IDNC is checked by running the algorithm 1000 times with different seeds of randomization for each of 5 graphs with 400 vertices and originally-assigned $k = 5, 10, 40, 80, 133$.

The Box plot of the approximation ratio is shown in Figure 1. In the Figure, the result for randomly selected k-set is also shown for comparison. IDNC works properly since the randomly selected k-set can be considered as its initial k-set.

We also compared IDNC performance with those of *Degree* and *BC*, which are the methods that select the k vertices with the top

Table 1: Frequency distributions of the number of repeat-loop iterations over 1000 runs of Algorithm IDNC for each of the 5 pmed graphs with 400 vertices.

#Iteration	$k = 5$ pmed16	$k = 10$ pmed17	$k = 40$ pmed18	$k = 80$ pmed19	$k = 133$ pmed20
1	4	0	0	0	0
2	151	38	0	0	0
3	335	220	17	3	0
4	256	304	138	34	14
5	136	222	220	146	70
6	80	114	266	226	169
7	24	55	174	219	229
8	9	24	113	198	215
9	5	13	41	86	152
10	0	8	18	51	78
11	0	2	7	27	44
12	0	0	3	7	16
13	0	0	3	3	9
14	0	0	0	0	2
15	0	0	0	0	2
Ave.	3.78	4.542	6.122	7.052	7.742

Table 2: Frequency distributions of the number of repeat-loop iterations over 1000 runs of Algorithm IDNC for 5-median problem using each of the 9 pmed graphs.

Graph	$n = 100$ pmed1	$n = 200$ pmed6	$n = 300$ pmed11	$n = 400$ pmed16	
Ave. #Ite.	3.737	3.285	3.779	3.78	
	$n = 500$ pmed21	$n = 600$ pmed26	$n = 700$ pmed31	$n = 800$ pmed35	$n = 900$ pmed38
	3.539	3.568	3.691	3.549	3.638

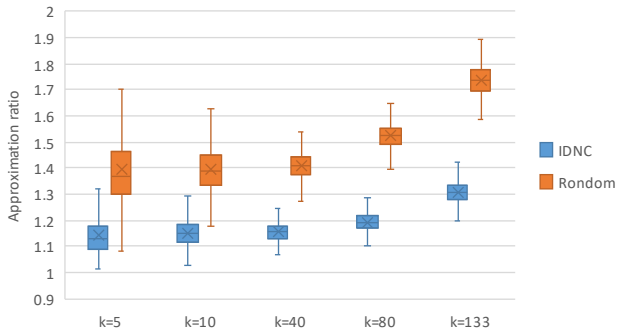

Figure 1: Box plot of the approximation ratio for each of the 5 pmed graphs with 400 vertices over 1000 runs

Table 3: Running time averaged over 1000 runs for 5-median problem using each of the 9 pmed graphs. OOM means that the algorithm cannot output a solution due to CPLEX's out-of-memory error.

Graph	$n = 100$ pmed1	$n = 200$ pmed6	$n = 300$ pmed11	$n = 400$ pmed16	$n = 500$ pmed21	$n = 600$ pmed26	$n = 700$ pmed31	$n = 800$ pmed35	$n = 900$ pmed38
IDNC	0.001336	0.004496	0.011487	0.020563	0.030748	0.045127	0.064287	0.082286	0.107724
Degree	0.000177	0.000576	0.001223	0.002495	0.003556	0.005104	0.017607	0.029295	0.031605
BC	0.000177	0.050645	0.051632	0.092465	0.184076	0.304972	0.457461	0.579401	0.833501
WF+CPLEX	0.450000	8.100000	21.890000	75.000000	39.600000	320.230000	1433.420000	OOM	OOM
WF	0.014334	0.100239	0.329614	0.779646	1.516490	2.613990	4.140870	6.175410	8.785920

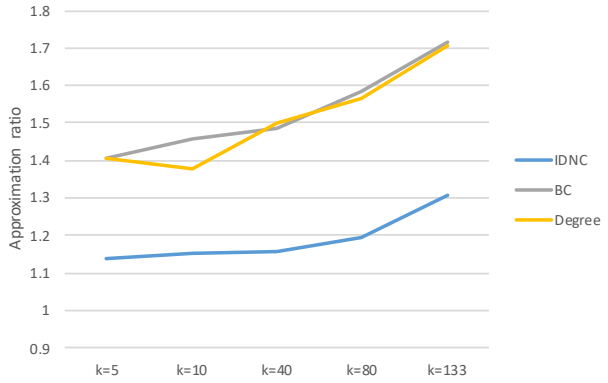

Figure 2: Approximation ratio averaged over 1000 runs for each of the 5 pmed graphs with 400 vertices

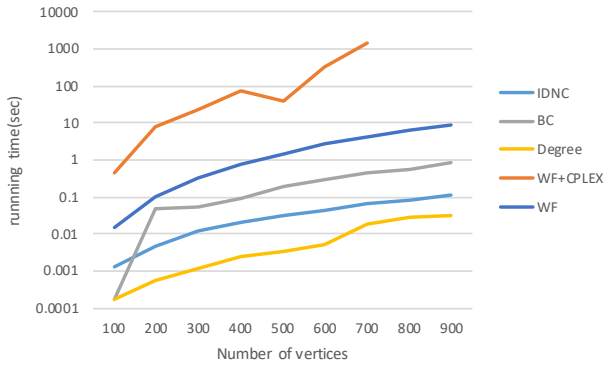

Figure 3: Curve of the running time averaged over 1000 runs for 5-median problem. Note that the y-axis is log-scaled.

scores in terms of simple centrality measures, *degree centrality* (degree centrality of v: the number of vertices having an edge connected to v) and *betweenness centrality* (betweenness centrality of v: the number of vertex pairs between which the shortest path passes v) [9]. The results are shown in Figure 2. We can see that the approximation ratio achieved by IDNC is significantly better than those achieved by the above two methods using simple centrality measures. The superiority of IDNC seems caused by taking account of edge lengths, which are considered in neither Degree nor BC.

5.2.3 Running time. The running time of IDNC was also demonstrated for 5-median problem by running IDNC 1000 times for each of the 9 graphs with different number of vertices. For comparison, we also executed the exact algorithm WF+CPLEX, and the two simple algorithms used in Sec. 5.2.2 (Degree and BC). To know the conversion time from edge-weighted graph to 0-1 integer programming problem, the running time of WF alone was measured, too.

The results are shown in Figure 3 and Table 3. The algorithm for exact solutions, WF+CPLEX is too slow to solve the problem for large-sized graphs. Even WF algorithm alone cannot be used for large-sized problems. Algorithm IDNC is faster than BC but slower than Degree, both of which achieved the approximate ratio worse than IDNC.

6 CONCLUSION

We proposed an efficient approximate algorithm for k-median problem on a graph and demonstrated its effectiveness in computation time and approximation ratio using the pmed dataset. Checking effectiveness for large real-world networks is our future work.

REFERENCES

[1] Osman Alp, Erhan Erkut, and Zvi Drezner. 2003. An Efficient Genetic Algorithm for the p-Median Problem. *Annals of Operations Research* 122, 1 (2003), 21–42.
[2] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. 2001. Local Search Heuristic for K-median and Facility Location Problems. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*. 21–29.
[3] J. E. Beasley. 1985. A note on solving large p-median problems. *European Journal of Operational Research* 21, 2 (1985), 270–273.
[4] J. E. Beasley. 1990. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society* 41, 11 (1990), 1069–1072.
[5] Fernando Chiyoshi and Roberto D. Galvão. 2000. A statistical analysis of simulated annealing applied to the p-median problem. *Annals of Operations Research* 96, 1 (2000), 61–74.
[6] N. Christofides and J. E. Beasley. 1982. A tree search algorithm for the p-median problem. *European Journal of Operational Research* 10, 2 (1982), 196–204.
[7] E. W. Dijkstra. 1959. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* 1, 1 (1959), 269–271.
[8] Robert W. Floyd. 1962. Algorithm 97: Shortest Path. *Commun. ACM* 5, 6 (1962), 345.
[9] Linton C. Freeman. 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry* 40, 1 (1977), 35–41.
[10] A. J. Goldman. 1971. Optimal Center Location in Simple Networks. *Transportation Science* 5, 2 (1971), 212–221.
[11] O. Kariv and S. L. Hakimi. 1979. An Algorithmic Approach to Network Location Problems. II: The p-Medians. *SIAM J. Appl. Math.* 37, 3 (1979), 539–560.
[12] F. E. Maranzana. 1964. On the Location of Supply Points to Minimize Transport Costs. *Journal of the Operational Research Society* 15, 3 (1964), 261—-270.
[13] Subhash C. Narula, Ugonnaya I. Ogbu, and Haakon M. Samuelsson. 1977. Technical Note—An Algorithm for the p-Median Problem. *Operations Research* 25, 4 (1977), 709–713.