Approximation Algorithms for k-Median Problems on Complex Networks: Theory and Practice

Roldan Pozo(✉)

National Institute of Standards & Technology, Gaithersburg, MD 20899, USA
pozo@nist.gov
https://www.nist.gov/people/roldan-pozo

Abstract. Finding the k-median in a network involves identifying a subset of k vertices that minimize the total distance to all other vertices in a graph. While known to be computationally challenging (NP-hard) several approximation algorithms have been proposed, most with high-order polynomial-time complexity. However, the graph topology of complex networks with heavy-tailed degree distributions present characteristics that can be exploited to yield custom-tailored algorithms. We compare eight algorithms specifically designed for complex networks and evaluate their performance based on accuracy and efficiency for problems of varying sizes and application areas. Rather than relying on a small number of problems, we conduct over 16,000 experiments covering a wide range of network sizes and k-median values. While individual results vary, a few methods provide consistently good results. We draw general conclusions about how algorithms perform in practice and provide general guidelines for solutions.

1 Introduction

The k-median problem is an important and fundamental problem in graph theory, and various application areas. Given a connected network, it seeks to find k vertices which are the closest (in terms of average distance) to the remaining vertices in the network graph. This is crucial in the spread of viral messages in social networks, disease contagion in epidemiological models, operation and distribution costs for goods and services, marketing and advertising, design layout of communication networks, and other wide-ranging applications.

Finding such an optimal set of vertices is referred to as the influence maximization problem [8] and numerous algorithms have been proposed to address this issue. Although these algorithms were not explicitly designed for the k-median problem, they share mathematical similarities, as they attempt to find influential vertices that are similarly well-connected and can quickly disseminate information throughout the network. The level of *influence* can be measured in various ways, typically employing diffusion models such as independent cascade model [4] or epidemiological models of the Susceptible-Infected (SI) and

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1143, pp. 89–101, 2024.
https://doi.org/10.1007/978-3-031-53472-0_8

Susceptible-Infected-Recovered (SIR) [5] types. Formally, the k-median problem on network graphs has been shown to be NP-hard (in terms of k) by reduction to the dominant cover problem [7] and is computationally intractable for large network graphs.

2 Notation and Definitions

Definition 1. k-median problem*: Given an integer k and a connected graph $G = (V, E)$, find a set of S of k vertices that minimize the summation of distances from S to the remaining vertices of G.*

Using $d(v, u)$ as the distance (length of shortest path) between vertices v and u, we can define the distance $d(v, S)$ between a single vertex and a vertex set S as the minimum distance between v and any vertex in S. We can then define the *average distance* $A(S)$ as

$$A(S) \doteq \frac{\sum_{v \in V} d(v, S)}{|V - S|} \tag{1}$$

In this context, we can restate the k-median problem on a network graph G to be the identification of a set of k-vertices which minimize the average distance to the remaining $|V| - k$ vertices:

$$M^*(k) \doteq \min_{|S|=k} A(S) \tag{2}$$

We refer to $M^*(k)$ as the true *optimal value* of the k-median problem on a graph G, and let $M(k)$ denote an approximation to this solution using the methods described in Sect. 3. When necessary, we use $M_{\texttt{method}}(k)$ to avoid ambiguity.

3 Approximation Algorithms and Related Problems

The field of approximation methods to the k-median problem is quite large. Resse [17] provides a comprehensive overview of over 100 methods from linear programming, genetic algorithms, simulated annealing, vertex substitution, and other approaches. In general, the algorithm with the best guaranteed approximation ratio is a local search and swap method [1] with provides a bound of $3+2/p$ where p is the number of vertices simultaneously swapped. Its computation time is $O(n^p)$, where n is the number of vertices. Thus, even for a quadratic-order complexity $O(n^2)$, which is quite limiting for large networks, the best guarantee we can get is a factor of four from optimal. While these approaches were adequate for small networks, the higher-order polynomial time complexity makes them unfeasible for networks with thousands or million of vertices [18].

Instead, researchers have turned their attention to algorithms for finding effective spreaders in connected networks with heavy-tailed degree distributions, often employing an Susceptible-Infected (SI) or Susceptible-Infected-Recovered

(SIR) model of spread [14,15], where $I(t)$ denotes the number of infected nodes at time t, and $I(0)$ is the number of initially infected nodes. The k-median can be thought of a special case of an SI model where the probability of an infected node transmitting the disease to a susceptible neighbor is 1.0, or an SIR model with the probability of recovery for each node is 0.0. In which case, the solution to the k-median problem can be thought of as maximizing the integral of the number of infected nodes over the propagation steps, starting with k initial infected vertices.

3.1 Degree Ordering

Approximating the k-median solution as the top k hubs of the network is perhaps the most straightforward approach:

$$X_{\texttt{degree}}\,(k) \;\leftarrow\; \underset{v \in V}{\text{argmax}}[k]\, \deg(v) \tag{3}$$

The idea here is that the hubs (high-degree vertices) serve as efficient spreaders since they are connected to large number of neighbors. This is countered by the notion that there may be significant overlap among their aggregate neighborhoods, with other vertices potentially covering the graph more effectively. This is a common criticism of degree ordering for this problem, but our experimental results show that this may not be as critical an issue in practice (see Sect. 6).

3.2 Extended Degree Ordering

A more sophisticated approach is the extended degree ordering, which measures the sum of degrees for neighboring vertices, and uses the top k values as an approximation of the k-median solution:

$$X_{\texttt{degree+}}\,(k) \;\leftarrow\; \underset{v \in V}{\text{argmax}}[k] \sum_{x \in N(v)} \deg(x) \tag{4}$$

This is a semi-local algorithm, utilizing more information about the network's topology by analyzing the second-level neighborhood, i.e. neighbors of neighbors. The motivation for this centrality measure is that it uses more information about the graph topology and can lead to an improved metric for identify candidate vertices for the set S.

3.3 PageRank Ordering

PageRank is a variant of the eigenvalue centrality, which treats the network as a flow graph and values vertices with high eigenvalues. It is the basis for some commercial web search engines. [14]. Given a damping factor, $0 \leq \delta \leq 1$, the PageRank centrality is given as the convergence of the iteration

$$\text{PageRank}(v) \;=\; (1-\delta) \;+\; \delta \sum_{u \in N(v)} \frac{\text{PageRank}(u)}{\deg(u)} \tag{5}$$

typically a value of $\delta = 0.85$ is used in these calculations [14]. The corresponding approximation for the k-median solution is

$$X_{\texttt{PRank}}\,(k) \;\leftarrow\; \underset{v \in V}{\text{argmax}}[k]\;\; \text{PageRank}(v) \tag{6}$$

3.4 VoteRank Ordering

A method developed by Zhang *et al.* [18], is the VoteRank algorithm, which uses an iterative voting methodology to determine the best influencer nodes. Each vertex i has a pair of values (S_i, T_i) denoting the collective (incoming) votes from neighbors S_i and the number of (outgoing) votes to give out in each voting round, T_i. At each voting round (complete pass through the graph) a vertex with the maximum (incoming) vote score is selected (i^*) and its (S_i^*, T_i^*) values are set to zero, effectively taking it out of future voting in subsequent rounds. The neighbors of vertex i^* have their respective T_i votes reduced by a fixed value f, and the process is repeated until k vertices are found. In their paper, the authors use $f = 1/\langle d \rangle$, where $\langle d \rangle$ is the average degree of the graph, and this value is fixed throughout the algorithm. Typically, one would choose f such that $kf \ll 1$ but this implementation does not allow the T_i values to go negative. The `VRank` k-median approximation is given by

$$X_{\texttt{VRank}}\,(k) \;\leftarrow\; \underset{v \in V}{\text{argmax}}[k]\;\; \text{VoterRank}(v) \tag{7}$$

3.5 Coreness Ordering

Another vertex centrality measure that has been proposed for finding effective spreaders is based on the degeneracy of network graphs. The i-core of a graph is collection of connected components that remain after all vertices with degrees less than n have been removed. (This is often referred to as the k-core of a graph in the literature, but we use i to avoid conflict with the k used in the k-median formulation.) To compute the i-core of a graph, we remove all vertices of degree $i - 1$ or less. This process is repeated until there are no vertices of the graph with degrees less than n. The notion here is that vertices in higher value i-cores represent the inner backbone of the network, as opposed to lower-valued i-cores which lie at its periphery, and serve as better-connected vertices to efficiently spread information throughout the network. The *core-number* of a vertex is the largest value i which it belongs to the i-core of the graph. Although one could use this centrality to identify candidate vertices [9], one problem that has been noted is that the core values of the highest vertices are often the same and hence are not distinguishable to form a proper ordering [2]. To remedy this, a slightly extended centrality has been proposed that replaces the i-shell value of a vertex with the sum of its neighbors' core-number. That is, if $c(v)$ is the core-number of v, then the `core` algorithm is

$$X_{\texttt{core}}\,(k) \;\leftarrow \underset{v \in V}{\text{argmax}}[k] \sum_{u \in N(v)} c(u) \tag{8}$$

3.6 Extended Coreness Ordering

Extensions to the $C(v)$ centrality have also been proposed [2] as an improved measure for influence. In a similar manner to $\deg^+$, *neighborhood coreness*, or core^+, uses the values of its neighbor's core centrality. We refer to this algorithm as `core+`

$$X_{\texttt{core+}}\,(k) \;\leftarrow\; \underset{v\in V}{\text{argmax}}[k] \sum_{u\in N(v)} C(u) \tag{9}$$

3.7 H-Index Ordering

The Hirsch index or H-index [6], originally intended to measure the impact of authors and journals by way of citations, has also been studied as centrality to measure ranking of influence and its relation to other centralities [12]. The original measure for an author or journal was determined as the number of n publications that have at least n citations. In terms of a network graph, the Hirsch index of a vertex v, given as $H(v)$, can be represented as the maximal number of n neighbors that each have at a degree of n or more. That is, if $h(v,n)$ is the number of neighbors of v with degree at least n,

$$h(v,n) \doteq \{e \mid \deg(e) \geq n,\ e \in N(v)\} \tag{10}$$

then

$$H(v) \;\doteq\; \max_{n} \{|h(v,n)| \geq n\} \tag{11}$$

The k-median approximation can be then be given as the `H-index` algorithm:

$$X_{\texttt{H-index}}\,(k) \;\leftarrow \underset{v\in V}{\text{argmax}}[k]\;\; H(v) \tag{12}$$

3.8 Expected Value (Random)

The **mean** average-distance of every k-element vertex set is simply the expected value of all possible combinations:

$$E^*(k) \doteq \frac{1}{\binom{|V|}{k}} \sum_{|S|=k} A(S) \tag{13}$$

That is, the average value of a **random** guess chosen from a uniform distribution of all $\binom{|V|}{k}$ possible sets. This can be computed exactly by brute force for small networks and small k-values. For larger cases, the expected value is approximated by sampling a finite subset of these possibilities and use of the Central Limit Theorem (CTL).

4 Experiments and Methodology

For these experiments, we focused on connected simple graphs that were undirected, unweighted, with no self-loops or multi-edges. This represents the least common denominator for graph topologies, as not all datasets have edge weights and other metadata. Directed graphs were represented as undirected by making each edge bi-directional. For disconnected networks, we used the largest connected component. Additionally, the input networks had their vertices renumbered to be contiguous for optimized operations, and therefore did not necessarily match the vertex numbers in the original sources.

Table 1. Application network topologies used in this study (largest connected component of undirected graph). The average degree is $\langle d \rangle$ and the maximum degree is Δ.

Network	Application	$\|V\|$	$\|E\|$	$\langle d \rangle$	Δ	$\Delta/\langle d \rangle$
Zebra	animal contact network	23	105	9.13	14	1.5
Dolphin	animal contact network	62	159	5.13	12	2.3
Terrorist network	social network	64	243	7.59	29	3.8
High School	social network	70	274	7.83	19	2.4
MIT students	mobile social network	96	2,539	52.90	92	1.7
Hypertext 2009	social interaction	113	2,196	38.9	98	2.5
Florida ecosystem wet	food network	128	2,075	32.42	110	3.4
PDZBase	metabolic network	161	209	2.59	21	8.1
Jazz	collaboration network	198	2,742	27.79	100	3.6
GE_200	top-level web graph	200	1,202	12,02	124	10.3
Chevron_200	top-level web graph	200	5.450	54.50	189	3.5
Abilene218	computer network	218	226	2.07	10	4.8
Bethesda	top-level web graph	255	422	3.31	81	24.5
C. Elegans	neural network	297	2,148	14.46	134	9.3
NetScience	co-authorship	379	914	4.82	34	7.0
Arenas-email	email communications	1,133	5,451	9.62	71	7.4
FAA air traffic	infrastructure	1,226	2,408	3.9	34	8.6
Human protein	protein interaction	2,217	6,418	8.94	314	26.5
ca-GrQc	co-authorship	4,158	13,422	6.46	81	12.5
ca-HepTh	co-autorship	8,638	24,806	5.74	65	11.3
ca-HepPh	co-authorship	11,204	117,619	23.38	491	21.0
ca-CondMat	co-authorship	21,363	91,286	8.54	279	32.6
email-Enron	email communications	33,696	180,811	10.73	1,383	128.9
cit-HepPh	citation network	34,401	420,784	24.46	846	34.6
flickrEdges	online social network	105,722	2,316,668	43.83	5,425	123.8
email-EuAll	email communications	224,832	339,925	3.02	7,636	2,525.3
com-YouTube	online social network	1,134,890	2,987,624	5.27	28,754	5,631.3
soc-Pokec	online social network	1,632,803	22,301,964	27.32	14,854	543.8
soc-LiveJournal	online social network	4,846,609	42,851,237	17.68	20,333	1,149.9

The dataset comprised a wide range of application areas, including social, mobile, metabolic, neural, email, biological, and collaboration networks listed in Table 1. Examples were collected from network databases Konect [10], SNAP [11], and UC Irvine [3], as well as several webgraphs generated by examining public websites. Network sizes ranged from less than 100 vertices (for exact verification of k-median problems) to networks with over 1 million vertices, with most networks containing several thousand vertices. This study focused on 32 of these networks, comparing the eight algorithms from Sect. 3 for k-values from 1 to 100, resulting in roughly 16,000 experiments of graph, algorithm, and k-value combinations. This provided a clearer view of the performance landscape and algorithm behavior.

Table 2. Average error (%) to true-optimal for small graphs ($1 \leq k \leq 5$)

Network	`random`	`degree`	`degree+`	`VRank`	`PRank`	`core`	`core+`	`H-index`
Abilene218	68.9	11.1	3.4	10.2	54.0	4.4	3.4	8.4
USAir87	60.0	2.6	2.7	2.6	3.1	2.7	2.7	10.7
ca-HepTh	46.4	4.7	5.8	4.7	4.7	26.2	26.2	34.9
ca-netscience	68.6	32.7	65.3	18.0	17.0	56.0	67.0	66.8
celegans	50.2	1.8	2.1	1.8	3.3	1.8	2.1	25.9
faa	48.4	10.7	5.0	10.7	8.9	4.8	5.3	12.6
foodweb_florida_wet	53.8	5.6	5.6	5.6	5.6	5.6	5.6	5.0
hypertext_2009	38.8	3.3	0.6	3.0	3.3	0.6	0.6	7.5
jazz	43.7	7.7	10.1	7.7	5.8	10.1	10.1	15.0
pdzbase	64.3	10.7	22.9	10.7	10.7	20.0	14.2	32.6

Computational experiments were conducted on a desktop workstation, running Ubuntu Linux 5.15.0-46, with an AMD Ryzen 7 1700x (8-core) processor running at 3.4 GHz, and outfitted with 32 MB of RAM[1]. The algorithms were coded in C++, and compiled under GNU g++ 11.1.0 with the following optimization and standardization flags: `[-O3 -funroll-loops -march=native -std="c++11"]`. Modules were used from the NGraph C++ library and Network Tookit [16], as well as optimized C++ implementations of algorithms noted in the paper.

5 Results

The results of the computational experiments on real networks showed significant variations. Despite claims made for any particular approach, we did not see a

[1] Certain commercial products or company names are identified here to describe our study adequately. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products or names identified are necessarily the best available for the purpose.

single method consistently producing a winner in every case. Instead, we were presented with trade-offs for varying network topologies. Nevertheless, we were able to form general observations about the expected behavior on classes of networks and provide some guidelines for choosing appropriate methods.

5.1 Comparisons with Optimal k-median solutions

For small networks, we compared the accuracy of the approximation algorithms by running them for various values, up to $k = 5$, except where limited by the computation effort. The results reveal several interesting patterns: (1) guessing a solution (`random`) performs, on average, within a factor of 2 from optimal, (2) for $k > 2$ some approximation methods (`degree+`, `core`, `core+`, `H-index`) can perform **worse** than random guessing, (3) most methods (excluding `random`) stay within a 1.5 factor of optimal, (4) `VRank` and `PRank` seem to be perform best, staying within 1.2 of optimal for $1 \leq k \leq 5$, and (5) `core`, `core+`, and `H-index`, typically perform worse, underperformed only by `random`.

The *C. Elegans* network, for example, shows extremely good approximations (relative error less than 5%) for all methods, except for `random` and `H-index`. In the *Abilene218* network, we encounter quite different behavior: the `PRank` method performs substantially worse than every other one, except random guessing. This is in sharp contrast to other examples, where `VRank` and `PRank` methods perform similarly and are often outperformed other algorithms. Finally, the *USAir87* network illustrates that the approximation algorithms are capable of calculating good-quality solutions (even `H-index`) that are significantly better than random guessing.

Table 3. Ranking of methods by actual error (%) in small graphs: average relative errors for each method in Table 2

method	error (%)
`VRank`	7.5
`degree`	9.1
`PRank`	11.6
`degree+`	12.3
`core`	13.2
`core+`	13.7
`H-index`	21.9
`random`	54.3

Table 2 provides a tabular form of similar results from a larger study of 10 networks. From this data we see that the behavior of these methods on real networks can vary significantly. Ignoring `random` and `H-index` momentarily, the remaining competitive methods can be quite accurate for these networks. For example,

C Elegans, *hypertext_2009*, *USAir87* and *foodweb_florida_wet* all exhibit approximations that are within 5% of optimal. The *ca-netscience* network was the sole outlier, with the best methods of the group exhibiting roughly a 20% error.

Taking the average error for each method across the networks we arrive at (Table 3) illustrating that `VRank` performed the best overall, with the other methods not too far behind. In this experiment, `degree+`, `core`, and `core+` did not perform as badly, while `H-index` and `random` fared significantly worse. This table represents the analysis for small network comparisons with exact solutions. It illustrates that the top methods generally work quite well, typically *within 10% to 20% of optimal* and seem reasonable candidates for testing on larger networks.

5.2 Case Studies: Million-Node Networks

Here we focus on three larger examples with millions of vertices and edges used in the study of large social networks [13]. For the *YouTube* network (V = 1,134,890 E = 2,987,624), the various heuristics perform about the same: roughly 35% better than a random guess. In this case, all methods yield nearly identical values, and one can simply use the fastest one (`degree`) to generate competitive results.

Table 4 lists the how each method ranked in the top 1%, 10%, and 100% of solutions. For example, `degree` scored in the top 10% solutions about 3/4 of the time, while random guessing always remained within a factor 2 of the best solution. Using the fastest method (`degree`) as a reference, we see that `VRank` and `PRank` provided the best solutions, but at a computational cost of nearly three orders of magnitude.

Similar results are seen for the *soc-pokec* social network (V=1,632,803, E=22,301,964). From these two examples, one may be tempted to conclude that the algorithms perform equally well for large networks. However, computations for the *LiveJournal* social network (V=4,846,609 E= 42,851,237) show a significant difference between various methods, with `PRank`, `VRank`, and `degree` performing better than most other heuristics and roughly 30% better than `random`.

Table 4. Percentage of cases where each method scored within x% of best solution ($k = 1, \ldots, 100$) for million-vertex network (YouTube). Nearly all methods are within a factor of two of best solution.

method	0% (best)	1%	10%	100%
`degree`	12.5	24.0	75.7	100.0
`degree+`	8.5	11.5	42.7	99.9
`VRank`	20.4	44.2	91.9	100.0
`PRank`	18.5	33.7	91.6	100.0
`core`	9.1	15.5	50.1	99.0
`core+`	6.3	8.7	41.0	97.8
`H-index`	3.9	6.0	39.7	95.5
`random`	0.6	3.1	12.3	99.4

5.3 Overall Results

Table 5 describes the overall performance of approximation algorithms on the 32 networks under consideration. The values are described as relative error to the best solution for each method. For example, a value of 10 signifies that particular method performed on average within 10% above the best possible heuristic for each k from one to one hundred. From here we see that `VRank` and `PRank` come in first and second position, respectively, for the majority of cases while `degree` comes in a close third position.

Table 5. Quality of methods for large graphs ($k = 1, 2, \ldots 100$). Relative performance (%) from best solution. A value of 10 signifies that on average that method performed 10% above best possible value from all heuristics.

Network	`degree`	`degree+`	`VRank`	`PRank`	`core`	`core+`	`H-index`	`random`
Abilene218	8.2	11.1	3.2	10.1	8.6	12.8	13.4	79.7
USAir87	10.0	18.1	0.4	1.8	18.0	18.9	19.7	43.7
amazon0302	0.5	3.0	0.3	0.4	0.8	3.0	3.9	48.0
areans_email	3.3	9.4	0.0	0.8	7.3	11.1	10.9	30.9
as20000102	1.4	12.5	0.0	0.5	4.5	11.4	7.0	76.1
bethesda	3.6	10.9	0.8	1.5	17.4	11.5	40.7	76.8
ca-CondMat	4.3	11.3	1.3	0.0	11.6	13.9	12.6	40.0
ca-GrQc	40.1	51.3	1.3	1.4	51.7	51.7	56.2	37.9
ca-HepPh	11.7	13.1	1.4	1.1	14.1	14.7	16.6	20.2
ca-HepTh	4.8	21.1	0.2	0.9	61.5	63.0	47.4	34.4
ca-netscience	12.6	34.9	0.3	3.2	35.6	59.5	48.0	50.9
celegans	0.9	6.9	0.1	0.9	1.6	6.6	7.8	25.0
chevron_top200	0.0	0.5	0.0	0.0	0.5	1.1	1.1	7.8
cit-HepPh	3.2	11.3	2.3	0.0	8.8	15.2	18.8	38.6
com-youtube	0.8	2.7	0.1	0.4	2.5	3.2	3.6	51.3
dlmf	3.1	7.6	0.0	2.0	8.5	7.5	8.3	67.2
email-Enron	2.7	11.6	0.7	0.4	9.0	12.1	13.3	59.1
email-EuAll	2.4	11.8	2.4	3.2	5.0	10.5	12.5	66.9
faa	8.6	31.0	0.7	1.8	18.0	41.1	30.8	40.3
flickrEdges	8.2	79.4	2.3	0.5	88.8	89.2	89.5	35.6
foodweb_florida_wet	0.1	0.6	0.1	0.1	0.4	0.7	1.5	8.7
ge_top200	3.0	14.3	0.3	0.3	9.4	17.4	22.8	36.0
human_protein_gcc	2.7	30.4	0.1	1.3	6.2	30.0	20.6	58.4
hypertext_2009	0.3	0.0	0.3	0.3	0.0	0.0	0.8	8.5
jazz	5.1	12.4	1.2	0.1	14.2	14.8	13.6	11.2
p2p-Gnuetalla31	0.3	1.5	0.0	1.5	0.6	2.6	5.5	32.3
pdzbase	7.1	69.6	0.1	3.7	19.3	70.9	55.2	71.1
roadNet-PA	8.8	17.5	8.7	1.2	19.1	32.1	351.3	3.5
soc-Epinions	1.2	4.0	0.1	0.2	3.3	4.4	5.0	48.8
soc-Slashdot0922	0.8	1.8	0.2	0.5	1.1	2.8	3.6	45.8
web-Stanford	2.2	21.3	0.8	0.7	12.4	20.9	26.5	37.0
wiki-Vote	1.6	2.0	0.6	0.1	2.8	2.0	2.2	38.5

Table 6. Efficiency of k-median approximations on large networks: computation time (secs)

Network	`degree`	`degree+`	`VRank`	`PRank`	`core`	`core+`	`H-index`	`random`
soc-LiveJournal	0.01	0.8	50.8	35.2	5.4	11.5	5.6	166.9
soc-pokec	0.01	0.5	24.4	20.0	2.2	4.7	3.28	62.5
com-youtube	0.01	0.04	2.1	1.4	0.1	0.32	0.75	7.0

Table 7. Ranked performance of k-median approximations. A value of x signifies that $M_{method}(k)$, on average, was within x% of the best solution for each graph and k-value combination from Table 5.

method	performance (%)
`VRank`	0.9
`PRank`	1.3
`degree`	5.1
`core`	14.4
`degree+`	16.7
`core+`	20.5
`H-index`	30.3
`random`	41.6

Table 7 summarizes these results, where we compute the overall error of each method from the best solution for each k-value. For example, `degree` is typically about 5% greater than the best solution, while `random` produced, on average, a solution that was less than 50% greater than the best algorithm.

6 Conclusion

We have compared eight k-median approximation methods for various k-values (typically 1 to 100) on 32 networks over a diverse range of application areas. After conducting thousands of experiments, we have observed patterns and formulated guidance for solving the k-median problem on a broad range of application network problems. Overall, these approximation algorithms are efficient and some can produce good-quality solutions on complex networks. However, they do not replace traditional methods[17] for general graphs without heavy-tailed degree distributions.

We have demonstrated that the algorithms in this study can indeed yield high-quality results on smaller networks where we can compute the optimal solution explicitly (Sect. 5.1, Table 3) with `degree`, `VRank`, and `PRank` achieving roughly a 1.1 factor of the true solution. By contrast, the best algorithms for general graphs provide a guaranteed factor of 3 or higher.

For larger networks, the exact optimal solution is not computationally tractable and we can only compare the approximation methods against themselves, and one may reach an incomplete or premature conclusions by examining only a small number of networks. Exploring a larger and more diverse dataset, however, reveals certain patterns that aid in algorithm choices.

Like many approximation heuristics, the practical question comes down to a trade-off between performance (computational cost) and quality of solution. If one is willing to accept a factor of 2 from the best methods, then simply choosing k random vertices from an uniform distribution may suffice. (This may come as a unexpected result, as hard problems typically do not behave in this manner.) If a higher quality solution is need, then we can consult Table 7 which summarizes the results of over 16,000 experiments. Here we see that `VRank` and `PRank` perform, on average, within about a 1.01 factor of the best method in every k-value in the [1:100] range. The simple `degree` method yields results on average within about 1.05 factor of the best method while exhibiting a performance speedup of three orders of magnitude over `VRank` and `PRank`.

Thus, we can form a general best-practices guide for choosing the appropriate algorithms:

- if a quality factor of 2 is sufficient, choose k random vertices from uniform distribution (`random`)
- if a better quality solution is needed, choose the top k hubs (`degree`)
- if quality still not sufficient, use `VRank` or `PRank` for slight improvement (at a 10^2 to $10^4 x$ computational cost)

In practice, these approximation algorithms remain efficient, even for networks containing millions of elements. The more expensive algorithms (`VRank`, `PRank`) require about a minute to approximate k-median solutions for up to $k = 100$ on a personal computer (Table 6). Thus, one possible approach would create an amalgamate *super-algorithm* which would run the seven methods (`degree`, `degree+`, `VRank`, `PRank`, `core`, `core+`, `H-index`) concurrently and choose the best one for each k-value. A final step could compare this to the expected value (`random`) to give an indication how well the approximation methods have improved the solution.

In summary, the methods presented here do a reasonable job at estimating the k-median problem on complex networks. Despite the challenges of this fundamental problem, these methods provide a reasonable approximation and can be used efficiently to formulate approximations to this important problem, providing researchers with practical tools in studying large-scale complex networks.

References

1. Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., Pandit, V.: Local search heuristic for k-median and facility location problems. In: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, pp. 21–29 (2001)

2. Bae, J., Kim, S.: Identifying and ranking influential spreaders in complex networks by neighborhood coreness. Physica A: Stat. Mech. Appl. **395**, 549–559 (2014). https://doi.org/10.1016/j.physa.2013.10.047
3. DuBois, C.L.: UCI Network Data Repository. University of California, School of Information and Computer Sciences, Irvine, CA (2008). http://networkdata.ics.uci.edu
4. Granovetter, M.: Threshold models of collective behavior. Am. J. Sociol. **83**(6), 1420–1443 (1978)
5. Hethcote, H.W.: The mathematics of infectious diseases. SIAM Rev. **42**(4), 599–653 (2000)
6. Hirsch, J.E.: An index to quantify an individual's scientific research output that takes into account the effect of multiple coauthorship. Scientometrics **85**(3), 741–754 (2010). https://doi.org/10.1007/s11192-010-0193-9
7. Kariv, O., Hakimi, S.L.: An algorithmic approach to network location problems. i: The p-centers. SIAM J. Appl. Math. **37**(3), 513–538 (1979)
8. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146 (2003)
9. Kitsak, M., et al.: Identification of influential spreaders in complex networks. Nat. Phys. **6**(11), 888–893 (2010)
10. Kunegis, J.: Konect - The koblenz network collection. In: Proceedings of the International Web Observatory Workshop, pp. 1343–1350. Boston, MA (2013). http://konect.uni-koblenz.de/networks
11. Leskovec, J., Krevl, A.: Snap datasets: stanford large network dataset collection (2014). http://snap.stanford.edu/data
12. Lü, L., Zhou, T., Zhang, Q.M., Stanley, H.E.: The h-index of a network node and its relation to degree and coreness. Nature Communications **7**(1), 10, 168 (2016). https://doi.org/10.1038/ncomms10168
13. Mislove, A., Marcon, M., Gummadi, K.P., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07). San Diego, CA (2007)
14. Newman, M.: Networks. Oxford University Press, Oxford (2018)
15. Porter, M.A., Gleeson, J.P.: Dynamical Systems on Networks. FADSRT, vol. 4. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-26641-1
16. Pozo, R.: NGraph Network Toolkit (2019). https://math.nist.gov/~RPozo/ngraph
17. Reese, J.: Solution methods for the p-median problem: an annotated bibliography. Networks **48**(3), 125–142 (2006). https://doi.org/10.1002/net.20128
18. Zhang, J.X., Chen, D.B., Dong, Q., Zhao, Z.D.: Identifying a set of influential spreaders in complex networks. Sci. Rep. **6**(1), 27, 823 (2016). https://doi.org/10.1038/srep27823