Experimental Evaluation of Approximation and Heuristic Algorithms for Maximum Distance-Bounded Subgraph Problems

Noname manuscript No.
(will be inserted by the editor)

Experimental Evaluation of Approximation and Heuristic Algorithms for Maximum Distance-Bounded Subgraph Problems .

Yuichi Asahiro · Tomohiro Kubo · Eiji Miyano

Received: date / Accepted: date

Abstract In this paper we consider two distance-based relaxed variants of the maximum clique problem (Max Clique), named Max d-Clique and Max d-Club for positive integers d. Max 1-Clique and Max 1-Club cannot be efficiently approximated within a factor of $n^{1-\varepsilon}$ for any real $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$ since they are identical to Max Clique [17,36]. Also, it is $\mathcal{NP}$-hard to approximate Max d-Clique and Max d-Club to within a factor of $n^{1/2-\varepsilon}$ for any fixed integer $d \geq 2$ and any real $\varepsilon > 0$ [6,3]. As for approximability of Max d-Clique and Max d-Club, a polynomial-time algorithm, called $\texttt{ReFindStar}_d$, that achieves an *optimal* approximation ratio of $O(n^{1/2})$ for Max d-Clique and Max d-Club was designed for any integer $d \geq 2$ in [3,4]. Moreover, a simpler algorithm, called $\texttt{ByFindStar}_d$, was proposed and it was shown in [6,4] that although the approximation ratio of $\texttt{ByFindStar}_d$ is much worse for any *odd* $d \geq 3$, its time complexity is better than $\texttt{ReFindStar}_d$.

In this paper we implement those approximation algorithms and evaluate their quality empirically for random graphs. The experimental results show that (i) $\texttt{ReFindStar}_d$ can find larger d-clubs (d-cliques) than $\texttt{ByFindStar}_d$ for odd d, (ii) the size of d-clubs (d-cliques) output by $\texttt{ByFindStar}_d$ is the same as ones by $\texttt{ReFindStar}_d$ for even d, and (iii) $\texttt{ByFindStar}_d$ can find the

A preliminary version of this paper appeared in Proceedings of the Joint 8th International Conference on Soft Computing and Intelligent Systems and the 17th International Symposium on Advanced Intelligent Systems, 892-897, 2016 [5]

Y. Asahiro
Department of Information Science, Kyushu Sangyo University, Fukuoka 813-8503, Japan.
E-mail: asahiro@is.kyusan-u.ac.jp

T. Kubo
Department of Systems Design and Informatics, Kyushu Institute of Technology, Fukuoka 820-8502, Japan.

E. Miyano
Department of Systems Design and Informatics, Kyushu Institute of Technology, Fukuoka 820-8502, Japan.
E-mail: miyano@ces.kyutech.ac.jp

same size of d-clubs (d-cliques) much faster than $\texttt{ReFindStar}_d$. Furthermore, we propose and implement two new heuristics, $\texttt{Hclub}_d$ for Max d-Club and $\texttt{Hclique}_d$ for Max d-Clique. Then, we present the experimental evaluation of the solution size of $\texttt{ReFindStar}_d$, $\texttt{Hclub}_d$, $\texttt{Hclique}_d$ and previously known heuristic algorithms for random graphs and Erdős collaboration graphs.

Keywords Maximum distance-bounded subgraph problems · d-clique · d-club · Approximation algorithms · Heuristic algorithms

1 Introduction

1.1 Background

Social network analysis (SNA) is the process of investigating social structures by using the network and graph theories. In SNA, undirected graphs are utilized to represent relationships or friendships between members of a social network. In those graphs, vertices represent the members of the network, and then edges represent some "positive" relationships between the members. In the following, an undirected graph is denoted by $G = (V, E)$, where V and E denote the set of vertices and the set of edges, respectively. $V(G)$ and $E(G)$ also denote the vertex set and the edge set of G, respectively. In this paper we assume that $|V(G)| = n$ and $|E(G)| = m$.

An important task in SNA is to detect as large a community as possible, where the community is formed by members such that those within a *cohesive group* interact with each other more frequently than with those outside the group. The *clique* concept to model the notion of a cohesive group is introduced by Luce and Perry in 1949 [21], where a *clique* in a graph G is a subset of pairwise adjacent vertices. This concept of clique and the associated maximum clique problem (Max Clique) have been studied extensively from different perspectives in various fields including graph theory, operations research, and theoretical computer science. Especially, the decision version of the Max Clique was one of Karp's original 21 problems shown to be $\mathcal{NP}$-complete in [18]. Although a lot of different problems have been modeled using cliques, one common observation is that they are extreme cohesive groups in which every member is connected with each other. This constraint might be too restrictive in some applications. For example, we can accept a situation such that any vertex in a group is reachable in at most d hops from any vertex, while direct connection between any two vertices is required in a clique. Therefore, the clique concept is relaxed into various forms in order to fit in settings of several problems and applications [1,31,32].

1.2 Distance-bounded subgraph problems

In this paper, we consider the maximum distance-d clique problem (Max d-Clique) and the maximum diameter-d club problem (Max d-Club) for a

positive integer d [22,2,25]. A *distance-d clique* (d-*clique* for short) in a graph G is a subgraph S of G such that for pairs of vertices $u, v \in S$, the distance between u and v is at most d in G. A *diameter-d club* (d-*club* for short) in a graph G is a subgraph S' of G such that for pairs of vertices $u, v \in S'$, the distance between u and v is at most d in S', i.e., the diameter of S' is at most d. MAX d-CLIQUE (MAX d-CLUB, resp.) is the problem of finding a maximum d-clique (maximum d-club, resp.) in a given graph G. For $d = 1$, MAX 1-CLUB is the same problem as MAX 1-CLIQUE, i.e., MAX 1-CLUB is simply MAX CLIQUE. Therefore, MAX d-CLIQUE and MAX d-CLUB are distance-based generalizations of MAX CLIQUE.

1.3 Previous results

As mentioned above, the original MAX CLIQUE is one of the most important and most investigated *hard* computational problems and thus considerable effort has been devoted to the development of various exact (exponential-time) algorithms and heuristic ones to solve MAX CLIQUE. An early and well-known exact algorithm was developed by Carraghan and Pardalos [11], which is based on the branch-and-bound (`BB` for short) strategy. Afterwards, a huge number of "refined" `BB` methods have been designed. For example, Östergård [27] proposed the `BB` based on solving sub-clique problems, Tomita and Seki [34] and Tomita, Sutani, Higashi, Takahashi, and Wakatsuki [35] the `BB` based on subgraph coloring, and Maslov, Batsyn, and Pardalos [24] the `BB` based on MAXSAT using the local search method. Also, many greedy-based and local-search-based heuristics have been proposed, e.g., [16,19].

It is well known that MAX CLIQUE, MAX d-CLIQUE and MAX d-CLUB are hard even to approximate; since MAX 1-CLIQUE and MAX 1-CLUB are identical to MAX CLIQUE, they cannot be efficiently approximated within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$ [17,36]. For any $\varepsilon > 0$ and a fixed $d \geq 2$, it can be shown that it is $\mathcal{NP}$-hard to approximate MAX d-CLUB to within a factor of $n^{1/3-\varepsilon}$ by using the gap preserving reduction provided by Marinček and Mohar [23] (although they assumed that $\mathcal{NP} \neq \mathcal{ZPP}$ in their original proof). Then, [6] improved the inapproximability from $\Omega(n^{1/3-\varepsilon})$ to $\Omega(n^{1/2-\varepsilon})$. As for approximability of MAX d-CLIQUE and MAX d-CLUB, [3,4] presented a polynomial-time algorithm, called $\texttt{ReFindStar}_d$, which achieves an *optimal* approximation ratio of $O(n^{1/2})$ for MAX d-CLIQUE and MAX d-CLUB for any integer $d \geq 2$. Furthermore, they proposed a simpler algorithm, called $\texttt{ByFindStar}_d$, and proved that although the approximation ratio of $\texttt{ByFindStar}_d$ is much worse for any *odd* $d \geq 3$, its theoretical time complexity is better than $\texttt{ReFindStar}_d$ in [3,4].

For MAX d-CLUB ($d \geq 2$), several heuristic algorithms are also known. Bourjolly, Laporte, and Pesanto [10] proposed three heuristic algorithms, named `CONSTELLATION`, `DROP`, and `k-CLIQUE&DROP` and showed their experimental evaluation. Shahinpour and Butenko [33] provided a variable neighborhood search heuristic algorithm and an exact one.

1.4 Our contributions

Max d-Clique and Max d-Club on several restricted graphs, such as bipartite and chordal graphs, are very difficult even to approximate. Actually, the approximation ratios of $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$ are very large, $O(n^{1/2})$ as described above. That is, from the theoretical view point, it is hard to design good algorithms running in polynomial time for those problems. On the other hand, however, $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$ might work much better for more practical situations since the approximation ratios are only the *worst-case guarantees*. Thus one of the goals of this paper is to implement those approximation algorithms and to evaluate their quality empirically for *random graphs* $G_{n,p}$, which have n vertices and each edge appears with probability $0 \leq p \leq 1$.

The other goal is to implement simple heuristic algorithms, called $\texttt{Hclique}_d$ and $\texttt{Hclub}_d$, for Max d-Clique and Max d-Club, respectively. Then, we compare the solution sizes and the execution times of $\texttt{ReFindStar}_d$, $\texttt{Hclique}_d$ and $\texttt{Hclub}_d$ with the sizes of the previously known two heuristics, $\texttt{PowerAndCliqueHeu}_d$ and $\texttt{DROP}_d$ [10] for random graphs and Erdős collaboration graphs [15, 7], where $\texttt{PowerAndCliqueHeu}_d$ is designed to find d-cliques by modifying MAXCLIQUEHEU proposed for Max Clique in [29].

The contributions of this paper are summarized as follows:

- We implement two approximation algorithms, $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$, and four heuristic algorithms, $\texttt{Hclique}_d$, $\texttt{Hclub}_d$, $\texttt{PowerAndCliqueHeu}_d$ and $\texttt{DROP}_d$, then examine the sizes of d-clubs (d-cliques) found for random graphs [12] and Erdős collaboration graphs [15, 7], and compare their running times.
- The experimental results show that $\texttt{ReFindStar}_d$ of approximation ratio $O(n^{1/2})$ can find larger d-clubs (d-cliques) than $\texttt{ByFindStar}_d$ of approximation ratio $O(n^{2/3})$ for both diameters $d = 3$ and $d = 5$, similarly to the approximation ratios theoretically guaranteed.
- We verify empirically that the sizes of d-clubs (d-cliques) output by $\texttt{ReFindStar}_d$ are the same as ones by $\texttt{ByFindStar}_d$ for $d = 2$ and 4.
- Although $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$ have the same approximation ratio for even d, the experimental comparisons of the running time show that $\texttt{ByFindStar}_d$ can find the same size of d-clubs (d-cliques) much faster than $\texttt{ReFindStar}_d$.
- The experimental evaluation shows that, generally speaking, the performance of $\texttt{ReFindStar}_d$ is worse for dense graphs, but better for sparse graphs than the performance of the heuristic algorithms.

The remainder of this paper is organized as follows: In Section 2 we introduce the notation and the definitions of d-cliques and d-clubs. Section 3 shows the approximation algorithms proposed in [6, 3, 4]. Section 4 presents our new and previously known heuristic algorithms. Section 5 provides experimental evaluation.

2 Preliminaries

Let $G = (V, E)$ be a connected undirected graph. An edge with endpoints u and v is denoted by $\{u, v\}$. $\Delta(G)$ and $\delta(G)$ respectively represent the maximum and minimum degree of a graph G. The set of vertices adjacent to a vertex v in G, i.e., the open neighborhood of v is denoted by $N_G(v)$. Then, $N_G[v]$ denotes the (closed) neighborhood of v, i.e., $N_G[v] = N_G(v) \cup \{v\}$.

A graph S is a subgraph of a graph G if $V(S) \subseteq V(G)$ and $E(S) \subseteq E(G)$. For a subset of vertices $U \subseteq V$, $G[U]$ denotes a subgraph of G induced by U. If every pair of two vertices in a graph S is adjacent, then S is called a *clique*.

For a pair of vertices u and v in G, the *distance* between u and v is defined as the length of the shortest path from u and v, and is denoted by $dist_G(u, v)$. Let $N_G^d[v] = \{u \mid dist_G(u, v) \leq d\}$ be the distance-d neighborhood of v. The *diameter* of G, denoted by $diam(G)$, is defined as $\max_{u,v \in V(G)}\{dist_G(u, v)\}$.

For a positive integer $d \geq 1$ and a graph G, the d-th power of G is the graph in which all pairs of vertices $u, v \in V(G)$ with $dist_G(u, v) \leq d$ are adjacent. The d-th power of G is denoted by G^d.

Definition 1 A subgraph S of G is a *d-clique* if $dist_G(u, v) \leq d$ holds for every pair of $u, v \in V(S)$.

The diameter of a d-clique may be greater than d. A d-clique whose diameter is at most d is called d-club:

Definition 2 A subgraph S of G is a *d-club* if $diam(S) \leq d$, i.e., $dist_S(u, v) \leq d$ holds for every pair of $u, v \in V(S)$.

Although in [25], d-club (or d-clique) is originally defined as a maximal subgraph, i.e., no super set of vertices forms a d-club (or d-clique), we do not restrict our attention to maximal ones in this paper since it is known to be $co\mathcal{NP}$-complete to answer whether a given d-club is maximal or not [28].

An algorithm `ALG` is called an α-approximation algorithm and `ALG`'s approximation ratio is α if $OPT(G)/ALG(G) \leq \alpha$ holds for every input graph G, where $OPT(G)$ and $ALG(G)$ are the numbers of vertices in obtained subgraphs by an optimal algorithm and `ALG`, respectively.

3 Approximation algorithms

3.1 Optimal approximation algorithm in [3, 4]

In this section we give a brief explanation on the $O(n^{1/2})$-approximation algorithm $\texttt{ReFindStar}_d$ proposed in [3, 4]. Note that its approximation ratio is the best possible in the sense that the lower bound of the approximation ratio is $\Omega(n^{1/2-\varepsilon})$ for any positive ε as shown in [6]. The algorithm $\texttt{ReFindStar}_d$ works for both Max d-Clique and Max d-Club and its main idea is as follows: Given a graph G, $\texttt{ReFindStar}_d$ first inserts an extra vertex into each edge

and obtains a new graph, say, H. Then it constructs the d-th power graph H^d of the newly obtained graph H by a simple polynomial-time algorithm `PowerOfGraph`, i.e., `PowerOfGraph` first computes $dist_G(u, v)$ for any pair of vertices $u, v \in V$, and then adds an edge $\{u, v\}$ if $2 \leq dist_G(u, v) \leq d$. Finally the algorithm outputs a 2-club in H^d corresponding to a d-club in the original G, which is also a d-clique. The following is a description of $\texttt{ReFindStar}_d$.

Algorithm $\texttt{ReFindStar}_d$
Input: An undirected graph $G = (V, E)$
Output: A subset of vertices $S \subseteq V$
Step 1. Insert one vertex $w_{u,v}$ into each edge $\{u, v\}$. The newly inserted vertices are called *white vertices* and the set of white vertices is denoted by V_W, while the original vertices in V are called *black vertices*. The obtained graph is denoted by $H = (V \cup V_W, E_H)$, where $E_H = \{\{u, w_{u,v}\}, \{v, w_{u,v}\} \mid \{u, v\} \in E\}$.
Step 2. Obtain the d-th power H^d of the graph H by applying $\texttt{PowerOfGraph}(H, d)$.
Step 3. For H^d, find a star $T = (V', E')$ having the maximum number of black vertices.
Step 4. Output $S = V' \cap V$, i.e., the set of the black vertices in V'.

One can see that for an input graph G having n vertices and m edges, $\texttt{ReFindStar}_d$ first constructs a graph H having $n + m$ vertices and $2m$ edges, and then computes the *all-pairs-shortest-paths* of H in order to obtain the d-th power H^d of H in Step 2. Thus, the dominant part in the running time of $\texttt{ReFindStar}_d$ is due to executing Step 2. Step 2 can be implemented in $O((n + m)^{2.3729})$ by merging the algorithm shown by Galil and Margalit [13, 14] and Seidel [30] and the one shown by Le Gall [20]. That is, if $m = \Omega(n^2)$, the running time grows up to $O(n^{4.7458})$. However, a simpler implementation can be done by *breadth-first-searches* from each vertex, whose running time is $O((n + m)^2)$ since H has only $2m$ edges. If $m = \Omega(n^2)$, this running time grows up to $O(n^4)$ which is smaller than $O(n^{4.7458})$ above.

As for the approximation ratio of $\texttt{ReFindStar}_d$, the following theorem holds [3,4]:

Theorem 1 ([3,4]) *Given an n-vertex graph and a fixed integer $d \geq 2$, $\texttt{ReFindStar}_d$ is a polynomial-time $\lceil n^{1/2} \rceil$-approximation algorithm with a running time of $O((n + m)^2)$ for* Max d-Club *and* Max d-Clique.

3.2 Faster approximation algorithms for even d in [6,4]

For $d \geq 2$, the following simpler and faster algorithm called $\texttt{ByFindStar}_d$ was proposed in [6,4], in which the $\lfloor d/2 \rfloor$-th power of the input graph G is constructed. The running time of $\texttt{ByFindStar}_d$ is $O(\min\{n^{2.3729}, nm\})$, where

again the dominant part is due to computing *all-pairs-shortest-paths* on the input graph G with n vertices and m edges in Step 1; the first term $n^{2.3729}$ comes from the implementation based on [13,14,30,20] and the second term, nm, comes from doing breadth-first-searches as above.

Algorithm $\texttt{ByFindStar}_d$
Input: An undirected graph $G = (V, E)$.
Output: A subset of vertices $S \subseteq V$.
Step 1. Obtain the $\lfloor d/2 \rfloor$-th power $G^{\lfloor d/2 \rfloor}$ of the graph G by applying $\texttt{PowerOfGraph}(G, \lfloor d/2 \rfloor)$.
Step 2. Find a vertex v having the maximum degree $\Delta(G^{\lfloor d/2 \rfloor})$, and then obtain the subgraph $N_{G^{\lfloor d/2 \rfloor}}[v]$, that is, the vertex v and its $\Delta(G^{\lfloor d/2 \rfloor})$ many neighbors.
Step 3. Output $S = N_{G^{\lfloor d/2 \rfloor}}[v]$.

It was shown that the approximation ratio of $\texttt{ByFindStar}_d$ for even d is $O(n^{1/2})$:

Theorem 2 ([6,4]) *For a fixed even integer $d \geq 2$ and a graph with n vertices and m edges, $\texttt{ByFindStar}_d$ is an $(n^{1/2}+O(1))$-approximation algorithm with a running time of $O(\min\{n^{2.3729}, nm\})$ for* Max d-Club *and* Max d-Clique.

Unfortunately, however, the approximation ratio of $\texttt{ByFindStar}_d$ becomes worse for odd d.

Theorem 3 ([6,4]) *For a fixed odd integer $d \geq 3$ and a graph with n vertices and m edges, $\texttt{ByFindStar}_d$ is an $(n^{2/3} + O(n^{1/3}))$-approximation algorithm with a running time of $O(\min\{n^{2.3729}, nm\})$ for* Max d-Club *and* Max d-Clique.

4 Heuristic algorithms for Max d-Clique and Max d-Club

In this section, we briefly explain newly proposed and previously known heuristic algorithms for the problems Max d-Clique and Max d-Club. First we propose simple heuristic algorithms for Max d-Clique and Max d-Club, respectively in Sections 4.1 and 4.2, based on the following simple observation. Then, Section 4.3 gives a heuristic algorithm for Max d-Clique based on a heuristic algorithm for Max Clique proposed in [29]. In Section 4.4, a heuristic algorithm for Max d-Club proposed in [10] is introduced.

Proposition 1 *If v is a vertex in a d-clique (or d-club), then the d-clique (or the d-club) includes vertices only from $N_G^d[v]$.*

4.1 A simple heuristic algorithm for Max d-Clique

Now we propose a simple heuristic algorithm; the following is a detailed description of our heuristic algorithm for Max d-Clique, called $\texttt{Hclique}_d$, based on Proposition 1, i.e., the algorithm applies the intuition that a vertex having many neighbors would be included in a solution.

Algorithm $\texttt{Hclique}_d$

Input: An undirected graph $G = (V, E)$.
Output: A subset of vertices $S \subseteq V$.
Step 1. For every vertex v, obtain the distance-d neighbor $N_G^d[v]$.
Step 2. Let $u = \arg\max_{v \in V}\{|N_G^d[v]|\}$, and then let $M = N_G^d[u]$.
Step 3. If $N_G^d[v] \cap M = M$ holds for every $v \in M$, then output $S = M$.
Step 4. Let $V' = \{v \mid v \in M, |N_G^d[v] \cap M| < |M|\}$. Then choose a vertex u such that $u = \arg\max_{v \in V'}\{|N_G^d[v] \cap M|\}$. Set $M = N_G^d[u] \cap M$ and then goto Step 3.

We can guarantee that the output of the above algorithm must be a d-clique as follows. In Step 3, we check whether $N_G^d[v] \cap M = M$ holds for every $v \in M$. This is equivalent to checking whether $G[M]$ is a d-clique, because if there exists a vertex $w \in M$ with $d_G(v, w) > d$, then $N_G^d[v] \cap M \neq M$ holds.

Now we bound the running time of $\texttt{Hclique}_d$. For each vertex v, $N_G^d[v]$ is constructed in $O(m)$ time, and so Step 1 can be done in $O(nm)$ time. Then, Step 2 requires $O(n)$ time. Since $|N_G^d[v]| \leq n$ and $|M| \leq n$, Steps 3 and 4 can be done in $O(n^2)$ time. Since the size of M decreases by the update in Step 4, the number of iterations of Steps 3 and 4 is at most $O(n)$. In total, the running time of $\texttt{Hclique}_d$ is $O(n^3)$.

4.2 A simple heuristic algorithm for Max d-Club

Next, we consider a heuristic algorithm $\texttt{Hclub}_d$ for Max d-Club. The difference from $\texttt{Hclique}_d$ is that $G[M]$ is used instead of G when checking the neighbors of a vertex in Steps 3 and 4.

Algorithm $\texttt{Hclub}_d$

Input: An undirected graph $G = (V, E)$.
Output: A subset of vertices $S \subseteq V$.
Step 1. For every vertex v, obtain the distance-d neighbor $N_G^d[v]$.
Step 2. Let $u = \arg\max_{v \in V}\{|N_G^d[v]|\}$, and then let $M = N_G^d[u]$.
Step 3. If $N_{G[M]}^d[v] \cap M = M$ holds for every $v \in M$, then output $S = M$.
Step 4. Let $V' = \{v \mid v \in M, |N_{G[M]}^d[v] \cap M| < |M|\}$. Then choose a vertex u such that $u = \arg\max_{v \in V'}\{|N_{G[M]}^d[v] \cap M|\}$. Set $M = N_{G[M]}^d[u] \cap M$ and then goto Step 3.

The output of $\texttt{Hclub}_d$ is a d-club which is observed by a similar argument to the one for $\texttt{Hclique}_d$. $\texttt{Hclub}_d$ also runs in $O(n^3)$ time by the very similar estimation to the one for $\texttt{Hclique}_d$.

4.3 A heuristic algorithm for Max d-Clique based on [29]

For Max Clique (i.e., $d = 1$), Pattabiraman, Patwary, Gebremedhin, Liao, and Choudhary proposed a heuristic algorithm `MAXCLIQUEHEU` in [29]. More precisely, it outputs only the size of the obtained clique, but it can be transformed to output the set of vertices in the obtained clique. It is not difficult to see that for a graph G the vertex set of a d-clique in G forms a clique in the d-th power G^d of G. Hence, we can use the algorithm for Max Clique as a subroutine in an algorithm for Max d-Clique after constructing G^d from the original graph G. Our heuristic algorithm $\texttt{PowerAndCliqueHeu}_d$ implements those ideas. The following is a description of `MAXCLIQUEHEU` and its subroutine `CLIQUEHEU` proposed in [29], assuming that $V = \{v_1, v_2, \ldots, v_n\}$. Note that max is a global variable used in both of `MAXCLIQUEHEU` and `CLIQUEHEU`, where `CLIQUEHEU` updates the value of max.

Algorithm `MAXCLIQUEHEU`

Input: An undirected graph $G = (V, E)$
Output: The size of a (found) clique in G
Step 1. Set $max = LB$ if a lowerbound LB of the size of a maximum clique in G is known; otherwise, set $max = 0$. Then set $i = 1$.
Step 2. If $|N_G(v_i)| \geq max$, then proceed to Step 3; otherwise goto Step 6.
Step 3. Set $U = \emptyset$.
Step 4. For each $v_j \in N_G(v_i)$, if $|N_G(v_j)| \geq max$ then $U = U \cup \{v_j\}$.
Step 5. Apply `CLIQUEHEU` to $(G, U, 1)$ as input.
Step 6. Set $i = i + 1$ if $i < n$; otherwise, i.e, if $i = n$, then output max.

Algorithm `CLIQUEHEU`

Input: An undirected graph $G = (V, E)$, a subset $U \subseteq V$, and an integer $size$
Goal: Update max
Step 1. If $U = \emptyset$, then proceed to Step 2; otherwise, goto Step 3.
Step 2. If $size > max$, then set $max = size$. Then halt.
Step 3. Select a vertex $u \in U$ of maximum degree in G, and update $U = U \setminus \{u\}$.
Step 4. Let $N'(u) = \{w \mid w \in N(u), |N_G(w)| \geq max\}$. Then apply `CLIQUEHEU` to $(G, U \cap N'(u), size + 1)$ as input.

The following is a detailed description of $\texttt{PowerAndCliqueHeu}_d$, which finds a clique in the d-th power G^d of the input graph G:

Algorithm $\texttt{PowerAndCliqueHeu}_d$

Input: An undirected graph $G = (V, E)$
Output: A subset of vertices $S \subseteq V$.
Step 1. Construct the d-th power graph G^d from G.
Step 2. Find a clique Q in G^d by applying `MAXCLIQUEHEU` to G^d.
Step 3. Output $S = V(Q)$.

One can see that the running time of $\texttt{PowerAndCliqueHeu}_d$ is $O(n^3)$ since `PowerOfGraph` and `MAXCLIQUEHEU` run in $O(n^2)$ time and $O(n \cdot (\Delta(G))^2) = O(n^3)$ time, respectively.

4.4 A heuristic algorithm for Max d-Club proposed in [10]

In this section, we give a brief description of the previously known heuristic algorithm $\texttt{DROP}_d$ for Max d-Club proposed in [10]:

Algorithm $\texttt{DROP}_d$

Input: An undirected graph $G = (V, E)$
Output: A subset of vertices $S \subseteq V$.
Step 1. Compute the shortest path length between every pair of vertices, and let $V' = V$.
Step 2. Compute for each vertex v_i of V' the number q_i of vertices of V' whose shortest path to v_i has a length of at least $d + 1$. If $q_i = 0$ for every vertex v_i, then $G[V']$ is a d-club and output $S = V'$.
Step 3. Let W be the set of vertices for which q_i is maximized. Determine a vertex $v_i^* \in W$ of the minimum degree in V. Remove v_i^* from V'.
Step 4. Update the shortest path lengths between pairs of vertices in the modified graph $G[V']$ and goto Step 2.

The running time of $\texttt{DROP}_d$ is $O(n^3 m)$.

5 Experimental results

This section is devoted to showing the experimental evaluations of the performances of two approximation algorithms, $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$, and four heuristic algorithms, $\texttt{Hclique}_d$, $\texttt{Hclub}_d$, $\texttt{PowerAndCliqueHeu}_d$, and $\texttt{DROP}_d$. All the experimental evaluations are conducted on a computer equipped with Intel(R) Xenon(R) CPU X5460 3.16GHz, 16GB of memory, and Windows10.

5.1 Random graphs

In the following, by *random graph* we mean the Erdős-Rényi [12] random graph, which is defined as follows:

Definition 3 ([12]) Given a positive integer n and a probability value $0 \leq p \leq 1$, we define a *random graph* $G_{n,p}$ to be the undirected graph on n vertices whose edges are chosen as follows: For all the $\binom{n}{2}$ pairs of vertices $u, v \in V(G_{n,p})$, there exists an edge (u, v) with probability p independently of all other edges.

That is, we first create a set of n vertices, and for each possible pair of vertices we flip a (biased) coin to determine if we should add an edge connecting them. One can see that $G_{n,p}$ follows a probability distribution over the set of all possible graphs on n vertices. It is known that there is a close relationship between the maximum size of a clique and a pair of n and p. For example, an expected value of the sizes of the maximum clique on random graphs increases drastically when the probability p exceeds a certain value, which is known as *phase transition* [26]. If p is small, it is known that the expected size of the maximum clique on random graphs can be estimated by the following [8]:

$$2\log_{1/p} n - 2\log_{1/p}\log_{1/p} n + 2\log_{1/p}(e/2) + 1 + o(1).$$

5.2 Comparisons between $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$ on Random Graphs

In this section we show the detailed experimental comparisons between the performances of two approximation algorithms, $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$, i.e., the sizes of d-clubs/d-cliques that the algorithms find, and their running times for random graphs. Although the experimental results for Max d-Clique are omitted, they are quite similar to the results for Max d-Club.

5.2.1 Experimental setup

We conduct mainly the following two experiments: (1) As mentioned in Sect. 3, if d is odd, then $\texttt{ReFindStar}_d$ is superior to $\texttt{ByFindStar}_d$ from the point of view of approximability for Max d-Club (and Max d-Clique). In this paper we evaluate their practical performances; we first examine the sizes of d-clubs that $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$ find for random graphs of n vertices, where $d = 3, 5$ and $n = 50, 100$. For each pair of n and p, we generate 1000 random graphs and calculate the average size of d-clubs output in the 1000 trials. (2) If d is even, the approximability of $\texttt{ByFindStar}_d$ is the same as the approximability of $\texttt{ReFindStar}_d$. Here, we evaluate the running time of $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$ for $2 \leq d \leq 5$ and random graphs $G_{n,p}$.

Remark that $\texttt{ByFindStar}_d$ (or $\texttt{ReFindStar}_d$) intermediately makes a $\lfloor d/2 \rfloor$-th (or d-th) power of a graph, which implies that we have to compute $dist_G(u, v)$ for any pair of vertices $u, v \in V$. Here, we use Johnson's algorithm implemented by the Boost Graph Library [9] to compute it. This algorithm has a time complexity of $O(|V||E| \log |V|)$.

5.2.2 Results

Figures 1 through 3 show the experimental results of the sizes of d-clubs in random graphs. The horizontal axis and the vertical axis denote the probability p and the average sizes of obtained d-clubs for each p, respectively.

Figure 1 shows the sizes of 3-clubs obtained by $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$ in random graphs with 50 and 100 vertices, respectively, in the

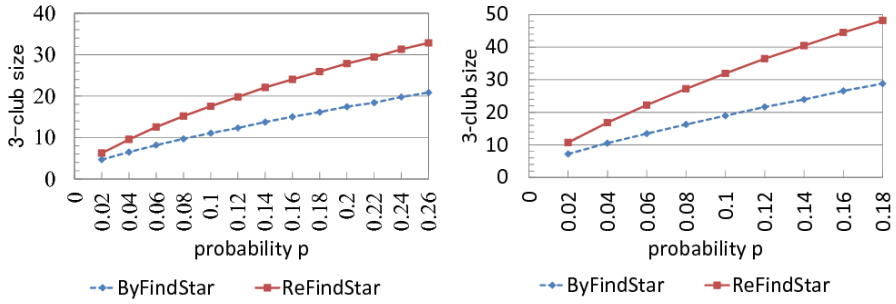

Fig. 1 The sizes of 3-clubs. (Left) $n = 50$, $p \leq 0.26$. (Right) $n = 100$, $p \leq 0.18$.

left chart and the right one. Similarly, Figure 2 shows the sizes of obtained 5-clubs in random graphs with 50 and 100 vertices. The experimental results show that $\texttt{ReFindStar}_d$ works better than $\texttt{ByFindStar}_d$ for both diameters $d = 3$ and $d = 5$, similarly to the approximation ratios theoretically guaranteed. As the numbers of vertices get larger, the gaps between them get larger. One of the remarkable results is that the output size of $\texttt{ReFindStar}_d$ is approximately 1.78 times larger than the output size of $\texttt{ByFindStar}_d$ when $d = 3$, $n = 100$, and $p = 0.08$. For $d = 3$, as the probability p becomes higher, the sizes of the obtained 3-clubs get bigger (in other words, "linearly increase"). For $d = 5$, however, it is shown that the sizes of the obtained 5-clubs increase slowly when the probability p becomes larger than a certain value.

For even $d = 2$ and 4, we show the experimental results on the sizes of the obtained d-clubs and running times by $\texttt{ByFindStar}_d$ and $\texttt{ReFindStar}_d$. First, these two algorithms output the same d-clubs in the experiments as shown in Figure 3. Figures 4 and 5 show the experimental results of the running times. The horizontal axis also denotes the probability p. The vertical axis denotes the running time to find d-clubs for all random graphs. Note that the number of random graphs generated is 1000 for each pair of n and p, and the unit of the running time is second. For $d = 2$, Figure 4 shows the results for graphs with 50 and 100 vertices. Similarly, Figure 5 shows the results for the case $d = 4$ and graphs with 50 and 100 vertices.

There is a large gap between the running time of the two algorithms. As the probability p becomes higher, the running time of $\texttt{ReFindStar}_d$ shows a drastic increase. This is because $\texttt{ReFindStar}_d$ inserts an extra vertex into each edge of the input graph $G = (V, E)$ (making a graph H). The graph H has $|V| + |E|$ vertices and $2|E|$ edges, thus, Johnson's algorithm runs in $O(2(|V| + |E|)|E| \log |V| + |E|)$. It can be seen that as the larger the number of edges of an input graph becomes, the running time of $\texttt{ReFindStar}_d$ increases quartically. That is, although the two algorithms have the same approximation ratio, $\texttt{ByFindStar}_d$ could find d-clubs much faster than $\texttt{ReFindStar}_d$ for even d. As a result, we should use $\texttt{ReFindStar}_d$ only for odd d; on the other hand, $\texttt{ByFindStar}_d$ should be used for even d.

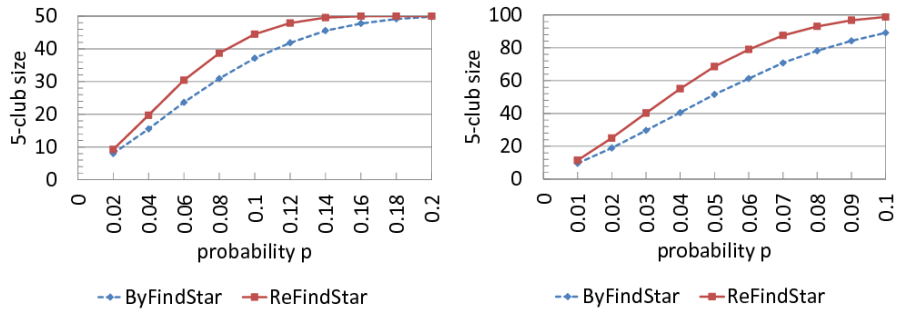

Fig. 2 The sizes of 5-clubs. (Left) $n = 50$, $p \leq 0.2$. (Right) $n = 100$, $p \leq 0.1$.

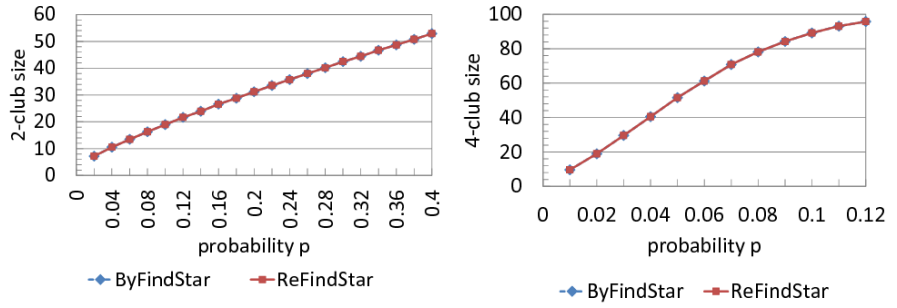

Fig. 3 (Left) The sizes of 2-clubs, $n = 100$, $p \leq 0.4$. (Right) The sizes of 4-clubs, $n = 100$, $p \leq 0.12$.

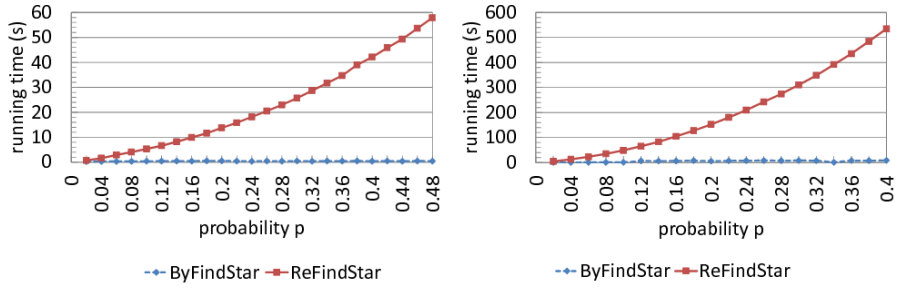

Fig. 4 The running time to obtain 2-clubs by `ByFindStar` and `ReFindStar`. (Left) $n = 50$. (Right) $n = 100$.

5.3 Comparisons between $\texttt{ReFindStar}_d$ and heuristics

In this section, we compare $\texttt{ReFindStar}_d$ with other heuristic algorithms. First, in Sect. 5.3.1, random graphs are given as input to the algorithms as in the previous section. Next, Sect. 5.3.2 shows the experimental results using Erdős collaboration graphs.

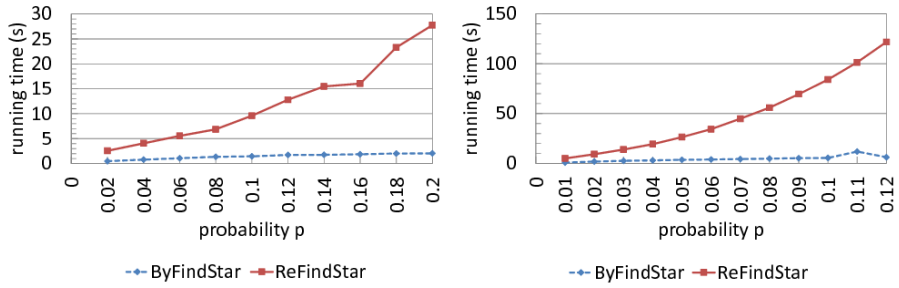

Fig. 5 The running time to obtain 4-clubs by `ByFindStar` and `ReFindStar`. (Left) $n = 50$. (Right) $n = 100$.

5.3.1 Results on random graphs

(Max d-Clique) Figures 6 and 7 show the experimental results on the sizes of obtained d-cliques in random graphs with 100 vertices for $d = 2$, 3, 4, and 5. The heuristic algorithms $\texttt{Hclique}_d$ and $\texttt{PowerAndCliqueHeu}_d$ obtain almost the same results, and they are better than $\texttt{ReFindStar}_d$. Only for $d = 2$ and $p \leq 0.08$, are the results of $\texttt{ReFindStar}_d$ better than the two heuristic algorithms.

In Figure 8 we show the running times of $\texttt{Hclique}_d$, $\texttt{ReFindStar}_d$ and $\texttt{PowerAndCliqueHeu}_d$ to find d-cliques in random graphs with 100 vertices for $d = 2$ and 4. Recall that the running time of $\texttt{ReFindStar}_d$ heavily depends on the number of edges in the intermediately constructed graph H (or H^d). Therefore, its running time increases as p gets larger. On the other hand, $\texttt{Hclique}_d$ and $\texttt{PowerAndCliqueHeu}_d$ need longer running times for sparse graphs than for dense graphs.

(Max d-Club) As for Max d-Club, Figures 9 and 10 show the experimental results of the sizes of obtained d-cliques in random graphs with 100 vertices for $d = 2$, 3, 4, and 5. Similarly to the above experiments for Max d-Club, the heuristic algorithms $\texttt{Hclub}_d$ and $\texttt{DROP}_d$ obtain almost the same results, and they are better than $\texttt{ReFindStar}_d$. Only for $d = 2$ and $p \leq 0.16$, are the results of $\texttt{ReFindStar}_d$ better than the two heuristic algorithms.

In Figure 11 we show the running times of $\texttt{Hclique}_d$, $\texttt{ReFindStar}_d$ and $\texttt{DROP}_d$ to find d-clubs in random graphs with 100 vertices for $d = 2$ and 4. One can see that it is harder for $\texttt{Hclique}_d$ and $\texttt{DROP}_d$ to find d-clubs in sparse random graphs.

5.3.2 Results on Erdős collaboration graphs

In the Erdős collaboration graph, Erdős, the researchers who collaborated with Erdős (they have Erdős number 1), and their coauthors (they have Erdős number 2) are included as vertices, and then edges represent collaborations [15]. Namely, the diameter of every Erdős collaboration graph is (at most) four. We use this Erdős collaboration graph [7] as inputs; we prepare 6 graphs named

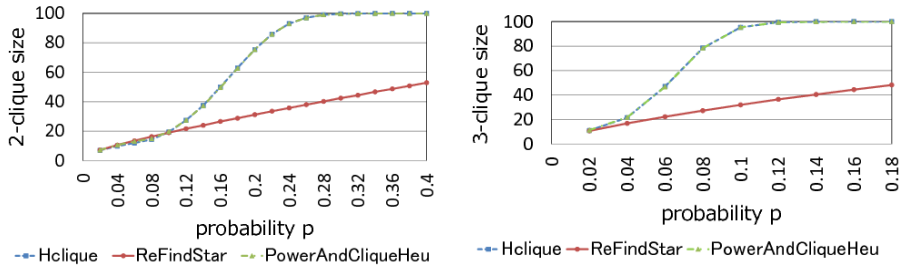

Fig. 6 The sizes of (Left) 2-cliques and (Right) 3-cliques in random graphs with 100 vertices.

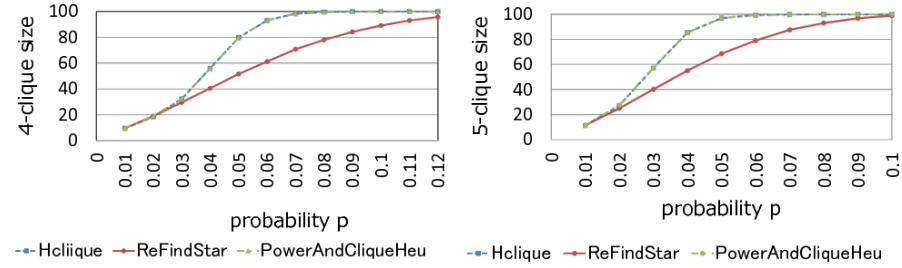

Fig. 7 The sizes of (Left) 4-cliques and (Right) 5-cliques in random graphs with 100 vertices.

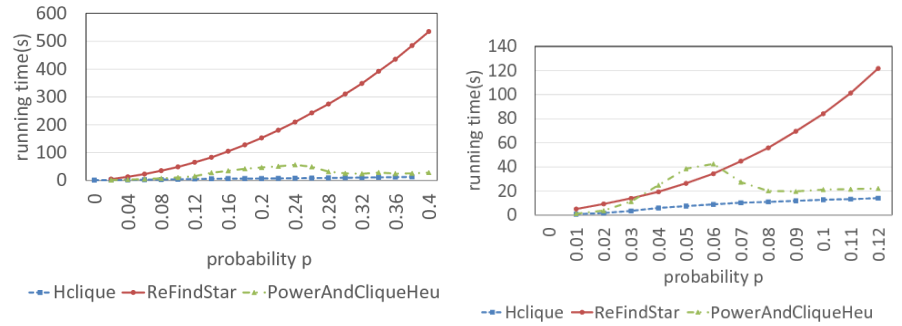

Fig. 8 The running time for $n = 100$. (Left) 2-cliques. (Right) 4-cliques.

erdos-x-y, where $x \in \{97, 98, 99\}$ and $y \in \{1, 2\}$. The graph erdos-x-1 includes the researchers with the Erdős number of at most 1 at year x (excluding Erdős himself). The graph erdos-x-2 includes the researchers with Erdős numbers 1 and 2.

Tables 1 and 2 show the sizes of the d-cliques and the d-clubs obtained by the algorithms for $d = 2$, 3, and 4, respectively. The values with $*$-marks are the largest ones for each Erdős collaboration graph. Generally speaking, `PowerAndCliqueHeu`$_d$ and `DROP`$_d$ respectively obtain larger d-cliques and d-clubs, compared to other two algorithms. However, for most of the sparse

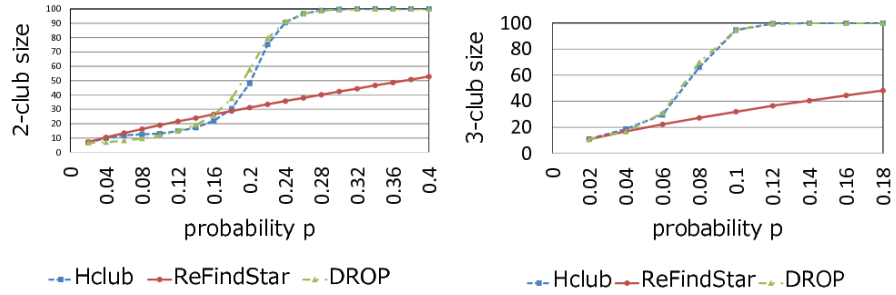

Fig. 9 The sizes of (Left) 2-clubs and (Right) 3-clubs in random graphs with 100 vertices.

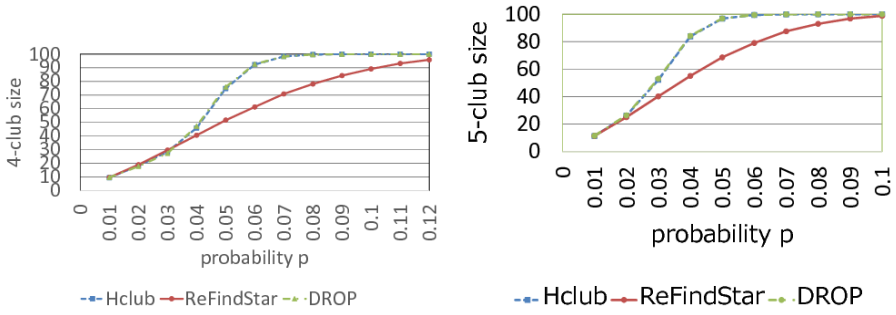

Fig. 10 The sizes of (Left) 4-clubs and (Right) 5-clubs in random graphs with 100 vertices.

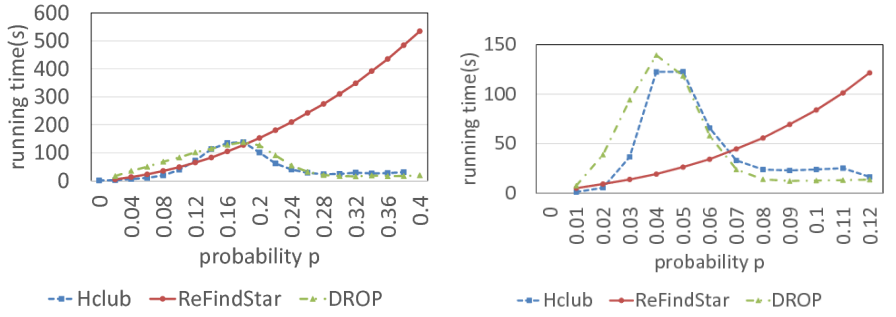

Fig. 11 The running time for $n = 100$. (Left) 2-clubs. (Right) 4-clubs.

graphs erdos-x-2's and $d = 4$, $\texttt{ReFindStar}_d$ obtains the largest 4-cliques and 4-clubs.

Acknowledgements This work was partially supported by JST CREST JPMJR1402 and the Grants-in-Aid for Scientific Research of Japan (KAKENHI) Grant Numbers JP17K00016 and JP17K00024.

Table 1 The sizes of the obtained d-cliques for Erdős collaboration graphs

instance	#vertices	density	d	$\texttt{Hclique}_d$	$\texttt{ReFindStar}_d$	$\texttt{PowerAndCliqueHeu}_d$
erdos-97-1	472	0.012	2	42*	42*	42*
			3	109	73	114*
			4	233*	214	232
erdos-97-2	5488	0.0006	2	144	258*	258*
			3	451	364	473*
			4	1281	1500*	1411
erdos-98-1	485	0.012	2	42	43*	42
			3	122*	75	121
			4	244*	225	241
erdos-98-2	5822	0.0006	2	162	274*	274*
			3	357	381	481*
			4	1403	1593*	1456
erdos-99-1	492	0.012	2	42	43*	42
			3	123*	75	122
			4	246*	227	244
erdos-99-2	6100	0.0005	2	168	277*	277*
			3	364	388	483*
			4	1386	1642*	1503

Table 2 The sizes of the obtained d-clubs for Erdős collaboration graphs

instance	#vertices	density	d	$\texttt{Hclub}_d$	$\texttt{ReFindStar}_d$	$\texttt{DROP}_d$
erdos-97-1	472	0.012	2	42*	42*	35
			3	97	73	117*
			4	232	214	235*
erdos-97-2	5488	0.0006	2	144	258*	258*
			3	427	364	517*
			4	1289	1500	1504*
erdos-98-1	485	0.012	2	42	43*	36
			3	91	75	121*
			4	243*	225	243*
erdos-98-2	5822	0.0006	2	162	274*	274*
			3	348	381	547*
			4	1368	1593*	1581
erdos-99-1	492	0.012	2	42	43*	37
			3	108	75	125*
			4	245*	227	245*
erdos-99-2	6100	0.0005	2	168	277*	277*
			3	456	388	562*
			4	1343	1642*	1631

References

1. Abello, J., Resende, M.G., Sudarsky, S.: Massive quasi-clique detection. In: Proc of LATIN 2002, pp. 598–612. Springer (2002)
2. Alba, R.D.: A graph-theoretic definition of a sociometric clique. Journal of Mathematical Sociology **3**(1), 113–126 (1973)
3. Asahiro, Y., Doi, Y., Miyano, E., Shimizu, H.: Optimal approximation algorithms for maximum distance-bounded subgraph problems. In: Proc of COCOA, pp. 586–600. Springer (2015)
4. Asahiro, Y., Doi, Y., Miyano, E., Shimizu, H.: Optimal approximation algorithms for maximum distance-bounded subgraph problems. Algorithmica **80**(6), 1834–1856 (2018)

5. Asahiro, Y., Kubo, T., Miyano, E.: Experimental evaluation of approximation algorithms for maximum distance-bounded subgraph problems. In: Proc of SCIS & ISIS, pp. 892–897 (2016)
6. Asahiro, Y., Miyano, E., Samizo, K.: Approximating maximum diameter-bounded subgraphs. In: Proc of LATIN 2010, pp. 615–626. Springer (2010)
7. Batagelj, V., Mrvar, A.: Graph files in bajek datasets. `http://vlado.fmf.uni-lj.si/pub/networks/pajek/data/gphs.htm`
8. Bollobás, B.: Random Graphs. Cambridge Univ (2001)
9. boost C++ Libraries – johnson_all_pairs_shortest_paths: `http://www.boost.org/doc/libs/1_60_0/libs/graph/doc/johnson_all_pairs_shortest.html`
10. Bourjolly, J.M., Laporte, G., Pesant, G.: Heuristics for finding k-clubs in an undirected graph. Computers & Operations Research **27**, 559–569 (2000)
11. Carraghan, R., Pardalos, P.: An exact algorithm for the maximum clique problem. Operations Research Letters **9**(6), 375–382 (1990)
12. Erdős, P., Rényi, A.: On Random Graphs I. Publicationes Math **6**, 290–297 (1959)
13. Galil, Z., Margalit, O.: All pairs shortest distances for graphs with small integer length edges. Inf. Comput. **134**, 103–139 (1977)
14. Galil, Z., Margalit, O.: All pairs shortest paths for graphs with small integer length edges. J. Comput. Syst. Sci. **54**, 243–254 (1977)
15. Grossman, J., Ion, P., Castro, R.: Erdős number project. `https://oakland.edu/enp/`
16. Grosso, A., Locatelli, M., Croce, F.: Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem. J. Heuristics **10**(2), 135–152 (2004)
17. Håstad, J.: Clique is hard to approximate within $n^{1-\varepsilon}$. In: Acta Mathematics, vol. 182 (1), pp. 105–142 (1999)
18. Karp, R.: Reducibility among combinatorial problems. Complexity of Computer Computations pp. 85–103 (1972)
19. Katayama, K., Hamamoto, A., Narihisa, H.: An effective local search for the maximum clique problem. Information Processing Letters **95**(5), 503–511 (2005)
20. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proc of ISAAC, pp. 296–303 (2014)
21. Luce, R., Perry, A.: A method of matrix analysis of group structure. Psychometrika **14**, 95–116 (1949)
22. Luce, R.D.: Connectivity and generalized cliques in sociometric group structure. Psychometrika **15**(2), 169–190 (1950)
23. Marinček, J., Mohar, B.: On approximating the maximum diameter ratio of graphs. Discrete Math **244**, 323–330 (2002)
24. Maslov, E., Batsyn, M., Pardalos, P.: Speeding up branch and bound algorithms for solving the maximum clique problem. J. Global Optimization **59**(1), 1–21 (2014)
25. Mokken, R.J.: Cliques, clubs and clans. Quality and Quantity **13**, 161–173 (1979)
26. Moore, C., Mertens, S.: The Nature of Computation. Oxford (2011)
27. Östergård, P.: A fast algorithm for the maximum clique problem. Discrete Applied Mathematics **120**(1), 197–207 (2002)
28. Pajouh, F.M., Balasundaram, B.: On inclusionwise maximal and maximum cardinality k-clubs in graphs. Descrete Optimization **9**, 84–97 (2012)
29. Pattabiraman, B., Patwary, M.M.A., Gebremedhin, A.H., Liao, W.k., Choudhary, A.: Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection. Internet Mathematics **11**(4-5), 421–448 (2015)
30. Seidel, R.: On the all-pairs-shortest-path problem in unweighted undirected graphs. J. Comput. Syst. Sci **51**, 400–403 (1995)
31. Seidman, S.B.: Network structure and minimum degree. Social Networks **5**(3), 269–287 (1983)
32. Seidman, S.B., Foster, B.L.: A graph-theoretic generalization of the clique concept. Journal of Mathematical Sociology **6**(1), 139–154 (1978)
33. Shahinpour, S., Butenko, S.: Algorithms for the maximum k-club problem in graphs. J. Comb Optim **26**, 520–554 (2013)
34. Tomita, E., Seki, T.: An efficient branch-and-bound algorithm for finding a maximum clique. In: Proc of DMTCS, pp. 278–289 (2003)

35. Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., Wakatsuki, M.: A simple and faster branch-and-bound algorithm for finding a maximum clique. In: Proc of WALCOM, pp. 191–203 (2010)
36. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: Theory of Computing, vol. 3, pp. 103–128 (2007)