Discrete Optimization 9 (2012) 84–97

Contents lists available at SciVerse ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

On inclusionwise maximal and maximum cardinality k-clubs in graphs

F. Mahdavi Pajouh, B. Balasundaram [*]

322 Engineering North, School of Industrial Engineering & Management, Oklahoma State University, Stillwater, OK 74078, United States

ARTICLE INFO

Article history:
Received 14 February 2010
Received in revised form 3 February 2012
Accepted 4 February 2012
Available online 24 February 2012

Keywords:
Clique
k-club
Exact combinatorial algorithms
Graph-based data mining
Social network analysis

ABSTRACT

A k-club is a distance-based graph-theoretic generalization of a clique, originally introduced to model cohesive social subgroups in social network analysis. The k-clubs represent low diameter clusters in graphs and are appropriate for various graph-based data mining applications. Unlike cliques, the k-club model is nonhereditary, meaning every subset of a k-club is not necessarily a k-club. In this article, we settle an open problem establishing the intractability of testing inclusion-wise maximality of k-clubs. This result is in contrast to polynomial-time verifiability of maximal cliques, and is a direct consequence of its nonhereditary nature. We also identify a class of graphs for which this problem is polynomial-time solvable. We propose a distance coloring based upper-bounding scheme and a bounded enumeration based lower-bounding routine and employ them in a combinatorial branch-and-bound algorithm for finding maximum cardinality k-clubs. Computational results from using the proposed algorithms on 200-vertex graphs are also provided.

1. Introduction

Given a simple, undirected graph $G = (V, E)$ of order $n = |V|$ and size $m = |E|$, the subgraph induced by $S \subseteq V$ is denoted by $G[S] = (S, E \bigcap (S \times S))$. A subset $S \subseteq V$ is called a clique if $G[S]$ is complete and it is called an independent set if $G[S]$ is edgeless. The maximum clique problem is to find a clique of maximum cardinality, referred to as the clique number of G, denoted by $\omega(G)$. An early application of cliques was in social network analysis (SNA) where they were used to model "tightly knit" social subgroups referred to as cohesive subgroups [1]. Since, there have been applications of the clique model in diverse fields such as coding theory, pattern recognition, fault diagnosis, bioinformatics, computational finance, and telecommunication in addition to SNA [2–5]. However, cliques also have certain drawbacks that motivated the development of graph-theoretic clique relaxations in SNA. Cliques require all possible edges to exist between a group of vertices for the group to be considered cohesive. This requirement was found to be overly restrictive leading to relaxations based on degree [6], distance [7–9], and edge density [10,11,5]. The clique provides the best possible guarantees for three desirable properties of a tight cluster, namely, high degree, short pairwise distances, high vertex and edge connectivity. In applications where cohesiveness is well described by any one of the three properties alone, the clique model becomes unduly restrictive. This article studies a distance-based generalization of cliques known as *k-clubs* which model low diameter clusters in graphs. The diameter of $G = (V, E)$ is given by $\text{diam}(G) = \max_{u,v \in V} d_G(u, v)$, where $d_G(u, v)$ is the length of a shortest path (in number of edges) between vertices u and v in G. The diameter of G is said to be infinite if there does not exist a path between a pair of vertices.

Definition 1. A k-clique is a subset $S \subseteq V$ for which $d_G(u, v) \leq k$ for all $u, v \in S$.

[*] Corresponding author. Tel.: +1 405 744 6055.
E-mail addresses: mahdavi@okstate.edu (F. Mahdavi Pajouh), baski@okstate.edu (B. Balasundaram).

1572-5286/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2012.02.002

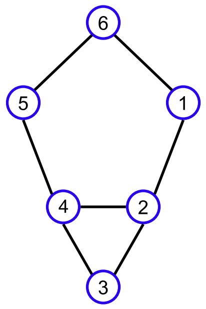

Fig. 1. 2-clique vs. 2-club.

Definition 2. A k-club is a subset $S \subseteq V$ for which $d_{G[S]}(u, v) \leq k$ for all $u, v \in S$. Equivalently, S is a k-club if $\text{diam}(G[S]) \leq k$.

Both k-cliques and k-clubs describe a clique when $k = 1$ and are relaxations when $k \geq 2$. Note that every k-club in G is also a k-clique in G, but the converse is not necessarily true when $k \geq 2$. In a k-clique S, there can exist two vertices $u, v \in S$ such that $d_G(u, v) \leq k$, but $d_{G[S]}(u, v) > k$. For the graph in Fig. 1, vertices $\{1, 2, 3, 4, 5\}$ form a 2-clique which is not a 2-club.

The k-clique model was introduced in SNA by Luce [7] as a distance-based relaxation of cliques. But the definition allows for a k-clique to utilize vertices outside the k-clique in the shortest paths. Since this was not desirable in a cohesive subgroup, the k-club model was introduced to bound pairwise distances in the induced graph instead of the original graph [8,9]. We denote the cardinality of a maximum k-clique by $\widetilde{\omega}_k(G)$, referred to as the k-clique number of G. The k-club number of a graph is the cardinality of a largest k-club in that graph which is denoted by $\overline{\omega}_k(G)$. Clearly, $\omega(G) \leq \overline{\omega}_k(G) \leq \widetilde{\omega}_k(G)$ for every positive integer k. Further, every k-club (k-clique) is also a $k + 1$-club ($k + 1$-clique) by definition. Note that k-cliques are equivalent to cliques through a simple observation concerning power graphs. Given a graph $G = (V, E)$, the k-th power of G is denoted by $G^k = (V, E^k)$ where $E^k = \{(u, v): d_G(u, v) \leq k\}$. So $S \subseteq V$ is a k-clique in G if and only if S is a clique in G^k, and $\widetilde{\omega}_k(G) = \omega(G^k)$. Given the modeling drawback of k-cliques and their equivalence to cliques in power graphs, they have received much less attention in the literature.

1.1. Previous work and our contributions

The problem of finding a maximum clique is NP-hard [12] and it is NP-hard to approximate within any factor $n^{1-\epsilon}$ for any $\epsilon > 0$ [13]. Further, finding a clique parameterized by solution size is not known to be fixed-parameter tractable, and it is in fact a basic $W[1]$-hard problem [14]. Although k-clubs generalize cliques, not all these complexity results extend to k-clubs. Bourjolly et al. [15] established the NP-hardness of the maximum k-clique and k-club problems and developed an exact branch-and-bound algorithm for the latter. Balasundaram et al. [16] independently established the NP-hardness of the problems for every fixed positive integer k, and in addition showed that the problems remain NP-hard even when restricted to graphs of fixed diameter. Hence, the maximum k-clique and k-club problems are known to be NP-hard for every fixed k even on graphs of diameter $k+1$. Bourjolly et al. also studied greedy construction and elimination heuristics for the problem in [17]. Pertinently, Butenko and Prokopyev [18] have shown that recognizing whether there is a gap between $\bar{\omega}_k(G)$ and $\bar{\omega}_l(G)$ for $k \neq l$ is NP-hard. This allowed them to show that for $k \geq 2$, there does not exist a polynomial time algorithm to find a k-club in G of size strictly larger than $\Delta(G) + 1$ on graphs with $\bar{\omega}_k(G) > \Delta(G) + 1$, unless $P = NP$. Note that $\Delta(G)$ is the maximum vertex degree in G, and a vertex of maximum degree along with all its neighbors forms a k-club in G for any $k \geq 2$.

The maximum k-club problem for fixed $k \geq 2$ was shown to be inapproximable within a factor of $n^{\frac{1}{3}-\epsilon}$ for any $\epsilon > 0$, if $NP \neq ZPP$ [19], which has been strengthened to $n^{\frac{1}{2}-\epsilon}$ recently under the assumption $P \neq NP$ [20]. Algorithms for the maximum k-club problem with approximation factors of $n^{\frac{1}{2}}$ for even k and $n^{\frac{2}{3}}$ for odd k have also been proposed recently [20]. The problem is also fixed-parameter tractable when parameterized by solution size as demonstrated in [21] where a $O((s-2)^s s! s^3 n + mn)$ time algorithm is presented to find a k-club of size s in G. The approximability and fixed-parameter tractability of the problem are in contrast to the results obtained for cliques.

As noted in [16], complete bipartite graphs are edge-critical 2-clubs. Therefore, finding a maximum 2-club in a bipartite graph amounts to finding a largest order biclique, which in trees corresponds to the vertex of maximum degree and its neighbors. Recently, it was shown that the maximum 2-club problem can be solved on bipartite graphs in $O(n^5)$, and the maximum k-club problem can be solved on trees and interval graphs in $O(nk^2)$ and $O(n^2)$, respectively [22]. Furthermore, it is also shown to be polynomial-time solvable on graphs with bounded tree-width or clique-width [22].

The 2-club polytope which is the convex hull of incidence vectors of 2-clubs in a graph, is studied in [16] and a family of facet defining inequalities are identified. This line of research is also explored in [23] where additional strong valid inequalities for the 2-club polytope are identified. While the k-club problem has a compact binary integer programming formulation for $k = 2$, the formulations proposed in [15,16] involve exponentially many variables for $k \geq 3$. Alternate formulations for the maximum 3-club problem are explored and valid inequalities are derived in [24]. A compact formulation for general k that uses $O(kn^2)$ variables and constraints has been developed recently in [25].

A fundamental challenge in the development of theory and algorithms for k-clubs is its nonhereditary nature. An important manifestation of this property is the intractability of testing maximality of k-clubs, a key result established in this article. This result is significant in light of the fact that testing whether a clique is maximal by inclusion is solved using a simple polynomial time algorithm. While the nonhereditary nature of k-clubs has been noted in literature (see [9,16]), the computational complexity of this problem has remained open. We settle this question in Section 2. In Section 3, we discuss the nonhereditary property and some of its implications. A class of graphs for which k-club maximality testing is polynomial-time solvable is identified in Section 4. A new upper-bounding technique based on a dual coloring problem and a new lower-bounding strategy based on a series of bounded enumeration searches for the k-club number of a graph are proposed in Section 5. A general framework of a combinatorial branch-and-bound for solving the maximum k-club problem is described in Section 6. Computational performance of the branch-and-bound algorithms using the proposed bounding techniques is studied in Section 7, and we conclude in Section 8.

2. NP-completeness of *k*-club maximality testing

An instance of k-CLUB MAXIMALITY TESTING is given by a simple undirected graph $G = (V, E)$ and a k-club D in G, and we ask if there exists a k-club D' in G such that $D \subset D'$? The following theorem establishes that k-CLUB MAXIMALITY TESTING (k-CMT) is NP-complete.

Theorem 1. *k-club maximality testing is NP-complete for any fixed integer $k \geq 2$.*

Proof. We prove this claim by a polynomial-time reduction from 3-SAT [12]. We assume that the 3-SAT instances satisfy the following restrictions: (a) No clause contains a literal and its negation; (b) There are at least 3 clauses in the 3-SAT formula. Note that the 3-SAT problem is still NP-complete under these restrictions. Before providing the transformation, we present some terminology that we use in the proof. The construction is slightly different depending on whether k is odd or even. A *k-chain* is a path of length k (on $k + 1$ nodes). When k is even, the $(\frac{k}{2} + 1)$th node is called the *midpoint* of the k-chain. When k is odd, the $\frac{k+1}{2}$th and $\frac{k+3}{2}$th nodes are called *midpoints*. Define $q = \frac{1}{2}[k - 2 + (k \bmod 2)]$. A$q$-pendant is a path of length q. One endpoint of a q-pendant is called the *head* and the other is called the *tail*. The node preceding the tail on the path from head to tail is called the *penultimate* node. For a pendant or a chain P containing vertices v_1, v_2, by $P[v_1, v_2]$ we denote the subpath from v_1 to v_2 including both end-points. We use the convention that a single vertex path is a path of length zero. Denote the 3-SAT instance with n boolean variables, m clauses with $3m$ literals as $B = \bigwedge_{i=1}^{m}(p_{i1} \vee p_{i2} \vee p_{i3})$. For any such 3-SAT instance $\langle B\rangle$, the following steps construct in polynomial time, an instance $\langle G, D\rangle$ for k-CMT such that the 3-SAT instance is satisfiable if and only if D is not a maximal k-club in G. □

Construction

1. For each clause $i = 1, \ldots, m$ in B, G contains a clique of size 3, with nodes labeled by the literals p_{ij}. We call these nodes in G, the *literal nodes*.
2. Every pair of literal nodes p_{ij} and p_{uv} that belong to different clauses ($i \neq u$) and are not negations of each other ($p_{ij} \neq \overline{p}_{uv}$) are connected by a k-chain. Such pairs of literal nodes are called *dcnn-pairs*. Each such k-chain consists of $k - 1$ new internal nodes with the dcnn-pair forming the endpoints. The internal nodes created will be called the *chain nodes*.
3. If k is even, we add all possible edges among the midpoints of all the k-chains so that they form a clique. If k is odd, we create a new node called the *nucleus node* and both midpoints of every k-chain are made adjacent to the nucleus node.
4. We always traverse all k-chains going from a lower index clause to a higher index clause, with the exception of k-chains between two clauses 1 and m. We traverse these k-chains going from clause m to clause 1. This imposes an orientation to these paths allowing us to refer to *succeeding* or *preceding* nodes on a path without ambiguity. We follow this convention in the remainder of this proof. Note that G is an undirected graph, and the direction is simply for path traversal. We refer to a k-chain that connects two literal nodes from two consecutive clauses $i, i + 1$ as a *k-bridge* for $i = 1, \ldots, m - 1$. The k-chains connecting literal nodes from clauses m to 1 are also called k-bridges. The set of all nodes thus created (literal nodes, chain nodes and the nucleus in case of odd k) are denoted by U. Note that $|U| = \text{total \# dcnn-pairs} \times (k - 1) + 3m + (k \bmod 2)$.
5. To every node $v \in U$, a q-pendant is attached by making its head adjacent to v. We refer to such a q-pendant as the *connector pendant of* v. We *associate* with every node $v \in U$, a q-pendant called the *opposing pendant of* v (can be taken to mean "indexed by v" as the opposing pendant of v will not be attached to v). Each opposing pendant will be connected to nodes in U once the notion of a *supporting node* is defined in the next step. Note that each such q-pendant of length q from head to tail results in the creation of $q + 1$ new nodes. We denote these q-pendant nodes by D where $|D| = 2(q + 1)|U|$.
6. For two chain nodes i, j, we say *i supports j* if i immediately precedes j on some k-chain. Every literal node supports the first chain node on every k-chain starting from it, and it is supported by the last chain node on every k-bridge ending at

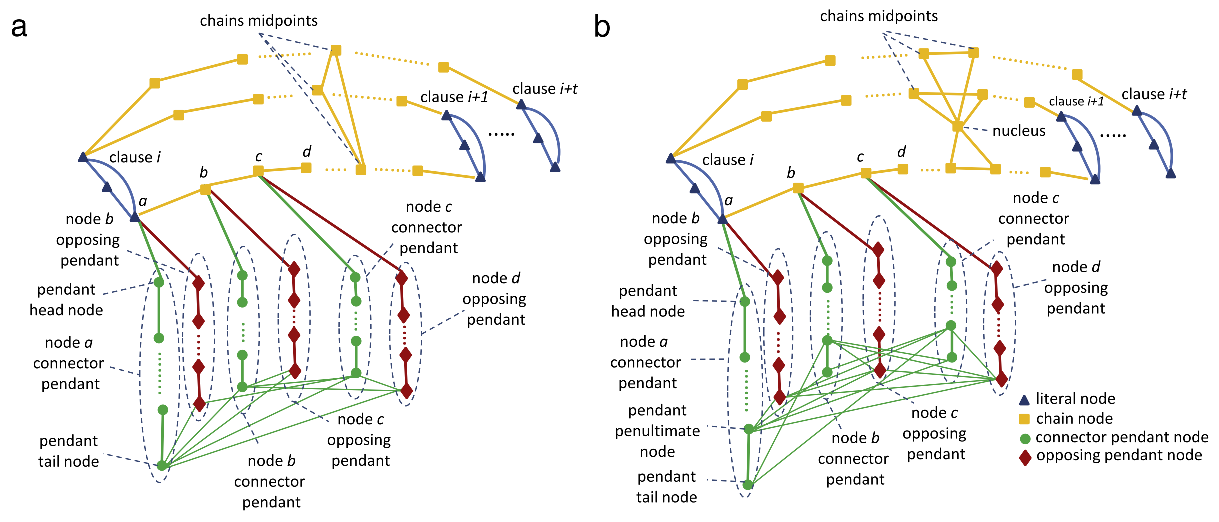
a
chains midpoints
clause *i*+1
clause *i*+*t*
clause *i*
b
c
d
a
node *b* opposing pendant
pendant head node
node *a* connector pendant
pendant tail node
node *b* connector pendant
node *c* opposing pendant
node *c* connector pendant
node *d* opposing pendant
b
chains midpoints
nucleus
clause *i*+1
clause *i*+*t*
clause *i*
node *b* opposing pendant
pendant head node
node *a* connector pendant
pendant penultimate node
pendant tail node
node *b* connector pendant
node *c* opposing pendant
node *c* connector pendant
node *d* opposing pendant
▲ literal node
■ chain node
● connector pendant node
◆ opposing pendant node

Fig. 2. (a) Illustration for even k and (b) illustration for odd k.

it. In case of odd k, we say the nucleus node is supported by all the chain midpoints that are adjacent to it. So in set U, every chain node has exactly one supporting node. Every literal node p_{ij} corresponding to clause i ($i = 2, \ldots, m$), has as many supporters as there are dcnn-pairs of p_{ij} in clause $i-1$. Every literal node p_{1j} has as many supporters as there are dcnn-pairs of p_{1j} in clause m. Every literal node p_{ij} corresponding to clause i ($i = 2, \ldots, m-1$) supports as many chain nodes as there are dcnn-pairs of p_{ij} in clauses $i+1, \ldots, m$. Every literal node p_{1j} supports as many chain nodes as there are dcnn-pairs of p_{1j} in clauses $2, \ldots, m-1$ and every literal node p_{mj} supports as many chain nodes as there are dcnn-pairs of p_{mj} in clause 1.

7. Consider any k-bridge connecting literal node p_{ij} to $p_{i+1,v}$ (including k-bridges from p_{mj} to p_{1v}) with chain nodes $a_1, \ldots, a_{k-1}$. Each node in the subpath from p_{ij} to a_{k-1} is made adjacent to the head of the opposing pendant of the node it supports.
8. Consider any k-chain connecting literal node p_{ij} to p_{uv} (traversed from clause i to clause u) that is not a k-bridge with chain nodes $a_1, \ldots, a_{k-1}$. Each node in the subpath from p_{ij} to a_{k-2} is made adjacent to the head of the opposing pendant of the node it supports. Note that in this case, we stop at the chain node a_{k-2} since a_{k-1} does not support p_{uv}.
9. If k is odd, all midpoints adjacent to the nucleus are also made adjacent to the head of the nucleus' opposing pendant.
10. If k is even, for each $v \in U$, the tail of connector pendant of v is made adjacent to the tail of every other q-pendant except the tail of the opposing pendant of v. For odd k, for each $v \in U$, the penultimate node of connector pendant of v is made adjacent to the tail of every other q-pendant except the tail of the opposing pendant of v. For odd k, we also create a clique among tail nodes of all opposing pendants.

This completes the construction of graph $G = (D \cup U, E)$. Fig. 2 illustrates the construction for even and odd k. Note that by construction, D is a k-club.

Claim 1. *For any $v_1 \in U$:*

a. *If $v_2 \in D$ is the head of v_1's opposing pendant then $d_{G[D\cup\{v_1\}]}(v_1, v_2) > k$.*
b. *If $v_2 \in D$ is not the head of v_1's opposing pendant then $d_{G[D\cup\{v_1\}]}(v_1, v_2) \le k$.*
c. *If $v_2 \in D$ is the head of v_1's opposing pendant then for a set $S \subseteq U$, $d_{G[D\cup S\cup\{v_1\}]}(v_1, v_2) \le k$, if and only if set S contains at least one supporter of v_1.*

Proof. Claims a and b are easily verified from the construction. To prove necessity of Claim c, suppose for a set $S \subseteq U$, $d_{G[D\cup S\cup\{v_1\}]}(v_1, v_2) \le k$ and this set does not contain any supporter of v_1. According to Claim 1a, we know $d_{G[D\cup\{v_1\}]}(v_1, v_2) > k$ so the shortest path between v_1 and v_2 with length less than or equal to k must contain at least one element from $S \setminus \{v_1\}$. Suppose $v_3 \in S \setminus \{v_1\}$ is an internal node on some shortest path from v_1 to v_2 in $G[D \cup S \cup \{v_1\}]$ and let p_3 denote v_3's connector pendant. Let p_1 denote v_1's opposing pendant. Since v_3 is not adjacent to v_2 (as it is not v_1's supporter), the shortest path between v_3 and v_2 for even k is $v_3 - p_3 - p_1[tail(p_1), v_2]$ with length $2q + 2 = k$. For an odd k, this shortest path is $v_3 - p_3[head(p_3) - penultimate(p_3)] - p_1[tail(p_1), v_2]$ of length $2q+1 = k$. Since the shortest path length between v_3 and v_2 is k, the shortest path between v_1 and v_2 that uses v_3 as internal node is at least $k+1$ which contradicts $d_{G[D\cup S\cup\{v_1\}]}(v_1, v_2) \le k$. So S should contain at least one supporter of v_1.

To establish sufficiency of Claim c, let $v_3 \in S$ be a supporter of v_1. So v_3 is adjacent to v_1 and v_2. So $d_{G[D\cup S\cup\{v_1\}]}(v_1, v_2) \le 2 \le k$. □

Claim 2. *For any two literal nodes $v_1, v_2 \in U$, $d_G(v_1, v_2) > k$ if and only if $v_1 = \overline{v}_2$.*

Proof. To show necessity, suppose there exist literal nodes $v_1, v_2 \in U$ such that $d_G(v_1, v_2) > k$ but $v_1 \neq \overline{v_2}$. Nodes v_1 and v_2 cannot belong to the same clause as they would form a clique in this case. Hence, v_1 and v_2 form a dcnn-pair, and by construction are connected by a k-chain contradicting $d_G(v_1, v_2) > k$. Thus, $v_1 = \overline{v_2}$.

To prove sufficiency, suppose there exist literal nodes $v_1, v_2 \in U$ such that $v_1 = \overline{v_2}$. So v_1 and v_2 belong to different clauses and there is no direct k-chain linking them. Then, it can be verified from the construction that the shortest path length between v_1 and v_2 is $k + 1$ (for both odd and even k). □

We now show that the 3-SAT instance is satisfiable if and only if D is not a maximal k-club in G. Suppose the 3-SAT instance has a satisfying assignment. Let S be the set of all literal nodes corresponding to literals with value equal to 1 in this assignment and all chain nodes in all k-chains connecting them. For odd k, S also contains nucleus node. $S \subseteq U$ and contains at least one literal node from each clause. For every literal node in S, there exists at least one k-chain in $G[D \cup S]$ with this literal node as one of its endpoints. We show that $S \cup D$ is a k-club in G.

Consider any two literal nodes $v_1, v_2 \in S$, either they belong to the same clause or they are from two different clauses. In the first case, v_1 and v_2 are adjacent. In the second case, v_1 and v_2 form a dcnn-pair and there is a k-chain connecting them in $G[D \cup S]$.

Now let v_1 be a literal node in S and v_2 be any chain node in this set. Either v_2 belongs to a k-chain that has v_1 as one of its endpoints or not. In the first case, it is easy to see that $d_{G[D\cup S]}(v_1, v_2) \leq k$. In the second case, let c_1 be a k-chain in $G[D \cup S]$ that has v_1 as one of its endpoints and c_2 denote the k-chain which contains v_2. For even k, consider the path $c_1[v_1, \mathit{midpoint}(c_1)] - c_2[\mathit{midpoint}(c_2), v_2]$ in $G[D\cup S]$. The length of this path is less than or equal to $k/2 + 1 + k/2 - 1 = k$. For odd k, without loss of generality, consider the closest midpoint to v_1 and v_2 on c_1 and c_2 respectively, that are both adjacent to the nucleus. This yields a path of length less than or equal to $\frac{k-1}{2} + 2 + (\frac{k-1}{2} - 1) = k$.

Now suppose v_1 and v_2 are two different chain nodes in S. They either belong to the same k-chain or they belong to two different k-chains c_1 and c_2. In the first case, it is easy to see that $d_{G[D\cup S]}(v_1, v_2) \leq k$. For the second case, for an even k, consider the path $c_1[v_1, \mathit{midpoint}(c_1)] - c_2[\mathit{midpoint}(c_2), v_2]$. The length of this path is at most $k/2 - 1 + 1 + k/2 - 1 = k - 1$. For an odd k, consider the closest midpoint to v_1 and v_2 on c_1 and c_2 respectively. The path through these nodes and the nucleus is of length at most $(\frac{k-1}{2} - 1) + 2 + (\frac{k-1}{2} - 1) = k - 1$. So for every two nodes $v_1, v_2 \in S$, $d_{G[D\cup S]}(v_1, v_2) \leq k$.

Consider any literal node v_1 in clause i, $2 \leq i \leq m$. S contains a literal node v_2 from clause $i - 1$ which forms a dcnn-pair with v_1. In case of $i = 1$, it contains such a dcnn-pair literal node from clause m. Because the k-chain connecting v_1 and v_2 is also in S, set S contains at least one supporter of v_1. Now consider any chain node $v_1 \in S$, since the k-chain containing v_1 is also in S, the supporter of v_1 is also available in S. So for any $v_1 \in S$, S contains at least one supporter of v_1. So by Claim 1, $d_{G[D\cup S]}(v_1, v_2) \leq k$ for each $v_1 \in S$, $v_2 \in D$. Since D is a k-club, $d_{G[D\cup S]}(v_1, v_2) \leq k$ for each $v_1, v_2 \in D$. So $G[D \cup S]$ is a k-club containing D and because $S \neq \emptyset$, D is not maximal.

Suppose D is not a maximal k-club and there exists a nonempty $S \subseteq U$ such that $S \cup D$ is a k-club. Select an arbitrary node $v_1 \in S$. Node v_1 is either a literal node or a chain node. If it is a chain node, by Claim 1c, S should contain a supporter of v_1. This supporter is again either a literal node or another chain node. By repeating this argument, we can conclude that S should contain at least one literal node v_1'. Suppose v_1' is located in clause i. Again by Claim 1c, the supporter of v_1' which is a chain node in some k-bridge should also be in S which results in the following conclusion. If $2 \leq i \leq m$, S should contain a literal node v_2 in clause $i - 1$ (in case of $i = 1$, the literal node v_2 is from clause m). Now by repeating this argument, we can conclude that set S should contain at least one literal node from each clause in B. Let $v_1, v_2 \in S$ be any two literal nodes. Since $d_{G[D\cup S]}(v_1, v_2) \leq k$, we can conclude that $d_G(v_1, v_2) \leq k$. Now by Claim 2, we have $v_1 \neq \overline{v_2}$. So by setting the value of the literals corresponding to literal nodes in S to 1 and the rest of them to zero, we will have a satisfying assignment for B.

This establishes that k-club maximality testing is NP-hard. It is easy to see that the problem is in class NP. This completes the proof of Theorem 1.

Contrasting this result with the polynomial-time solvability clique (and k-clique) maximality testing, it is essential to understand the reason for Theorem 1. Despite the conceptual similarity of k-clubs to k-cliques and cliques, there is a fundamental characteristic that distinguishes k-clubs. Cliques and k-cliques are hereditary as every subset of a k-clique is also a k-clique, for every fixed positive integer k. This allows us to verify maximality by inclusion of a k-clique C in polynomial time using the following algorithm. Let $C' = C$, pick any $v \in V \setminus C'$, if $C' \cup \{v\}$ is a k-clique, then add v to C', otherwise delete v from V and repeat until $V \setminus C' = \emptyset$. At termination, C' is a maximal k-clique containing C. However, this approach is not valid for finding maximal k-clubs for $k \geq 2$ as the k-club property is not hereditary. Consider the graph in Fig. 1. $\{1, 5, 6\}$ is 2-club in this graph. The sets $\{1, 5, 6, 4\}$, $\{1, 5, 6, 3\}$, $\{1, 5, 6, 2\}$ however are not 2-clubs. But the maximal 2-club containing $\{1, 5, 6\}$ is $\{1, 5, 6, 2, 4\}$.

3. Nonhereditary properties and maximality by inclusion

A graph property Π is said to be *hereditary on induced subgraphs*, if G is a graph with property Π then every vertex induced subgraph of G also satisfies property Π. Further, property Π is said to be *nontrivial* if it is true for a single vertex graph and is not satisfied by every graph. A property is said to be *interesting* if there are arbitrarily large graphs satisfying Π. Yannakakis [26] showed that the *maximum Π problem* to find the largest order induced subgraph that does not violate property Π is NP-hard for any property Π that is nontrivial, interesting and hereditary on induced subgraphs. This result has a very broad scope as complete subgraphs, edgeless subgraphs, planar subgraphs, bipartite subgraphs, perfect subgraphs are examples of Π that are nontrivial, interesting and hereditary.

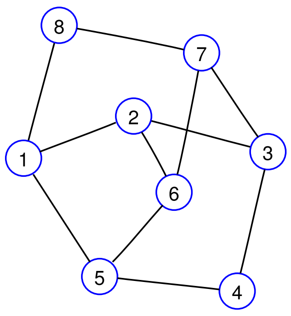

Fig. 3. A graph in which no maximal 2-clique is a 2-club.

If Π is hereditary and $G = (V, E)$ is an arbitrary graph with $V \supseteq C' \supset C$ such that $G[C]$ and $G[C']$ satisfy Π, the elements in $C' \setminus C$ can be added to C in any order with all the intermediate sets inducing subgraphs satisfying Π. Hence, if Π is hereditary, proving maximality by inclusion of $C \subseteq V$ such that $G[C]$ satisfies Π is equivalent to establishing the nonexistence of any single vertex $v \in V \setminus C$ such that $G[C \cup \{v\}]$ satisfies Π. Our result in Section 2 shows that there is no polynomial time algorithm available to test k-club maximality unless $P = NP$. Hence, there could be no generic polynomial time algorithm (as the one described above for hereditary properties) unless $P = NP$, to test maximality by inclusion of subgraphs satisfying any nonhereditary property.

In addition to the complexity theoretic perspective afforded by the result of Yannakakis, we can also view the impact of nonhereditary graph properties from a matroid theory perspective. A combinatorial system described by the pair $M = (S, \mathcal{I})$ where S is a finite ground set and $\mathcal{I}$ is a collection of subsets of S satisfying some specified property Π, is a matroid if the following three axioms hold: (M0) $\emptyset \in \mathcal{F}$; (M1) If $J' \subseteq J \in \mathcal{F}$, then $J' \in \mathcal{F}$; (M2) For every $J \in \mathcal{F}$, every maximal (by inclusion) subset in $\mathcal{F}$ containing J has the same cardinality. A fundamental result in matroid theory is that the maximum Π problem for a matroid can be solved in polynomial time using the greedy algorithm [27]. If M satisfies axioms (M0) and (M1), it is called an *independence system* and the maximum Π problem in general is NP-hard on independence systems [28]. Note that collection of cliques in a graph form an independence system. In contrast, the greedy algorithm can be used to find a maximal by inclusion subset satisfying Π in polynomial time. In light of Theorem 1, combinatorial systems built on nonhereditary properties are much harder to deal with than matroids or independence systems.

Before concluding this section, we outline some algorithmic implications of the nonhereditary nature of k-clubs, especially for combinatorial algorithms. While this feature also poses interesting challenges to identifying facets and valid inequalities for the k-club polytope, our present focus is not on the polyhedral approach.

Since k-clubs have a more restrictive definition than k-cliques, it is no surprise that a maximal k-club need not be a maximal k-clique. For instance, $\{1, 2, 3, 4\}$ is a maximal 2-club in Fig. 1, but not a maximal 2-clique. On the other hand, if a maximal k-clique satisfies the diameter requirement it is also a maximal k-club. This observation has been employed in the past to detect cohesive subgroups in social networks [29]. However, it is possible for $k \geq 2$, that no maximal k-clique in a graph is a k-club. Fig. 3 shows a graph with exactly two maximal 2-cliques $\{1, 2, 3, 4, 5, 6, 7\}$ and $\{1, 2, 3, 5, 6, 7, 8\}$, neither is a 2-club. Hence, even if we enumerate all maximal k-cliques in the graph and choose the ones satisfying the diameter at most k condition, we still may not have all the maximal k-clubs or worse, we may fail to identify a single k-club.

Furthermore, simple extensions of well known exact algorithms for maximum cliques such as the Carraghan–Pardalos algorithm [30], or the Östergård's algorithm [31] are not likely. The Carraghan–Pardalos algorithm is a back-tracking algorithm (a depth-first search order implicit enumeration) which requires that a maximal clique be found prior to back-tracking, which cannot be accomplished in polynomial time for k-clubs unless $P = NP$. Given $G = (V, E)$ with $V = \{1, \dots, n\}$, Östergård's clique algorithm relies on the bounded increase in the size of a maximum clique going from $G[V_{i+1}]$ to $G[V_i]$ where $V_i = \{i, \dots, n\}$. This is true for cliques as the size can at most go up by one vertex. Now consider the graph $G = (V, E)$ where $V = \{1, \dots, n\}$ and $E = \{(1, i) : i = 2, \dots, n\}$, *i.e.*, G is a *star graph* with central vertex 1 and leaves $2, \dots, n$. The 2-club number of $G[V_i] = 1$ for each $i = 2, \dots, n$, but $\bar{\omega}_2(G[V_1]) = n$. Even designing a basic branch-and-bound where the branching is carried out by a selected vertex being included or excluded to create two child nodes, presents interesting challenges. For instance, at some node of the search tree, let F^1 be the set of nodes fixed to be included and F^0 be the set of nodes deleted from consideration. For $k \geq 2$, we cannot fathom by infeasibility if $G[F^1]$ is not a k-club, as addition of vertices from $V \setminus \{F^0 \cup F^1\}$ could turn it into one. Likewise, if $G[F^1]$ is a k-club, we cannot fathom by feasibility unless we know it is indeed a maximal k-club. Hence, in the algorithm developed in Section 6, the pruning of the search tree is accomplished by using bounds in combination with other necessary conditions.

4. Graphs with a polynomial time algorithm for finding maximal k-clubs

We refer to a k-clique which induces a connected subgraph a *connected k-clique*. In a given graph, if every connected k-clique is also a k-club then a k-club's maximality can be checked using a method similar to k-clique maximality testing. Since, by definition every k-club is a connected k-clique and in such graphs every connected k-clique is also a k-club, a

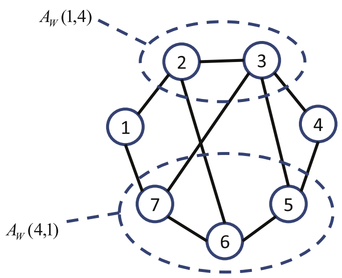

Fig. 4. An asymmetric partitionable cycle with respect to nodes 1 and 4, $W = 1 - 2 - \cdots - 7 - 1$.

maximal connected k-clique is also a maximal k-club. Given a graph $G = (V, E)$, maximality of a connected k-clique D can be checked by testing nodes in $V \setminus D$ that have at least one edge to some node in D, one at a time, to see if they can be added to D.

Definition 3. A graph $G = (V, E)$ is called a partitionable cycle, if it contains a spanning cycle W and a pair of nonadjacent nodes i and j such that every edge in $E \setminus E(W)$ has one endpoint in $A_W(i, j)$ and the other endpoint in $A_W(j, i)$, where $A_W(i, j)$ and $A_W(j, i)$ are the internal nodes on the two paths between i and j in W. A partitionable cycle is said to be asymmetric if $|A_W(i, j)| \neq |A_W(j, i)|$ (see Fig. 4).

Theorem 2. *Given a graph $G = (V, E)$ and fixed integer $k \geq 2$, every connected k-clique is a k-club if G does not contain an asymmetric partitionable cycle on c vertices as an induced subgraph, where $5 \leq c \leq 2k + 1$.*

Proof. Suppose a given graph $G = (V, E)$ contains a connected k-clique D which is not a k-club. Then there exist $i, j \in D$ such that $2 \leq d_G(i, j) \leq k$ but $d_{G[D]}(i, j) = k + l < \infty$ where $1 \leq l < \infty$. Let $i = i_0 - i_1 - \cdots - i_{k+l} = j$ be a shortest path between i and j in $G[D]$. Now $P_1 \colon i = i_0 - i_1 - \cdots - i_{k+1}$ is a shortest path between i and i_{k+1} in $G[D]$ and $d_{G[D]}(i, i_{k+1}) = k + 1$. Since $i, i_{k+1} \in D$, there must be a shortest path P_2 between i and i_{k+1} in G with length $2 \leq l' \leq k$. Since P_1 is a shortest path in $G[D]$ and P_2 is shorter than P_1, there exists node $v \in P_2$ such that $v \notin D$ and therefore $v \notin P_1$.

Now consider $G[V(P_1) \cup V(P_2)]$. Moving along P_1 from i to i_{k+1}, let S denote the sequence of nodes in P_1 which also belong to P_2. So we have $S \colon i = s_0, s_1, \ldots, s_p = i_{k+1}$. If $(s_j, s_{j+1}) \in E$ for a $j \in \{0, \ldots, p - 1\}$ then $(s_j, s_{j+1}) \in P_1$ and $(s_j, s_{j+1}) \in P_2$ because otherwise P_1 and P_2 are not shortest paths between i and i_{k+1} in $G[D]$ and G respectively. Further, if $(s_j, s_{j+1}) \in E$ for all $j \in \{0, \ldots, p - 1\}$ then $S = P_1 = P_2$ which is a contradiction since length of P_2 is strictly less than length of P_1. So there exist $h \in \{0, \ldots, p - 1\}$ such that $(s_h, s_{h+1}) \notin E$. Since $s_h, s_{h+1} \in P_1$, there exists at least one node between s_h and s_{h+1} on P_1. Let I_1 denote the set of all such nodes. Similarly for P_2, there exists at least one node between s_h and s_{h+1} on P_2 and let I_2 denote the set of all such nodes. $I_1 \cap I_2 = \emptyset$ because moving along P_1 from i to i_{k+1}, s_{h+1} is the first node in P_2 after s_h. $G[\{s_h, s_{h+1}\} \cup I_1 \cup I_2]$ is a partitionable cycle with respect to s_h, s_{h+1}, hence, $G[V(P_1) \cup V(P_2)]$ contains at least one partitionable cycle.

Suppose every partitionable cycle in $G[V(P_1) \cup V(P_2)]$ is symmetric. Like before, moving along P_1 from i to i_{k+1}, let S be the sequence of the nodes in P_1 which are also in P_2 and moving along P_2 from i to i_{k+1}, let T be the sequence of the nodes in P_2 which are also in P_1. So $S \colon i = s_0, s_1, \ldots, s_p = i_{k+1}$ and $T \colon i = t_0, t_1, \ldots, t_p = i_{k+1}$. The elements of S and T are the same but their sequence might be different. We know length of $P_1 = \sum_{j=0}^{p-1} d_{G[D]}(s_j, s_{j+1})$ and length of $P_2 = \sum_{j=0}^{p-1} d_G(t_j, t_{j+1})$.

Now suppose the sequences S and T are the same, that is $s_j = t_j$ for each $j \in \{0, \ldots, p\}$. So length of $P_1 = \sum_{j=0}^{p-1} d_{G[D]}(s_j, s_{j+1})$ and length of $P_2 = \sum_{j=0}^{p-1} d_G(s_j, s_{j+1})$. Now if for some $j \in \{0, \ldots, p - 1\}$, $(s_j, s_{j+1}) \in E$ then $d_{G[D]}(s_j, s_{j+1}) = d_G(s_j, s_{j+1}) = 1$ and if for some $j \in \{0, \ldots, p - 1\}$, $(s_j, s_{j+1}) \notin E$, according to the previous discussion, there exists a partitionable cycle W with respect to s_j, s_{j+1}. Let $A_W(s_j, s_{j+1})$ contain the nodes between s_j and s_{j+1} on P_1 and $A_W(s_{j+1}, s_j)$ contain the nodes between s_j and s_{j+1} on P_2. We know $d_{G[D]}(s_j, s_{j+1}) = |A_W(s_j, s_{j+1})| + 1$ and $d_G(s_j, s_{j+1}) = |A_W(s_{j+1}, s_j)| + 1$. Since every partitionable cycle is symmetric, we have $|A_W(s_j, s_{j+1})| = |A_W(s_{j+1}, s_j)|$ which results in $d_{G[D]}(s_j, s_{j+1}) = d_G(s_j, s_{j+1})$. So finally we have length of P_1 is equal to the length of P_2 which is a contradiction.

Now suppose sequences S and T are different. Let s_j be the first node in sequence S which is different from the corresponding node in sequence T (which is t_j). Note that s_j is located after t_j in T and t_j is located after s_j in S. Hence, $d_{G[D]}(s_{j-1}, t_j) = d_{G[D]}(s_{j-1}, s_j) + d_{G[D]}(s_j, t_j)$ and since $d_{G[D]}(s_j, t_j) > 0$ we have,

$$d_{G[D]}(s_{j-1}, t_j) > d_{G[D]}(s_{j-1}, s_j). \tag{1}$$

Considering T, we know t_j is the first node after s_{j-1} and $(s_{j-1}, t_j) \notin E$ as otherwise P_1 is not a shortest path since s_j is located between s_{j-1} and t_j in P_1. So according to the previous discussion, there exists a partitionable cycle W with respect to s_{j-1}, t_j. Let $A_W(s_{j-1}, t_j)$ contain the nodes between s_{j-1} and t_j in P_1 and $A_W(t_j, s_{j-1})$ contain the nodes between s_{j-1} and t_j in P_2. Since every partitionable cycle is symmetric, we have $|A_W(s_{j-1}, t_j)| = |A_W(t_j, s_{j-1})|$. So $d_{G[D]}(s_{j-1}, t_j) = d_G(s_{j-1}, t_j)$. Now using Eq. (1) we have,

$$d_G(s_{j-1}, t_j) > d_{G[D]}(s_{j-1}, s_j). \tag{2}$$

In T, s_j is located after t_j. So,

$$d_G(s_{j-1}, t_j) < d_G(s_{j-1}, s_j). \tag{3}$$

Eqs. (2) and (3) imply $d_G(s_{j-1}, s_j) > d_{G[D]}(s_{j-1}, s_j)$ which is impossible.

So in $G[V(P_1) \cup V(P_2)]$, there must exist at least one asymmetric partitionable cycle. Since the length of P_1 is $k+1$ and maximum length of P_2 is k, the order of such an asymmetric partitionable cycle is at most $(k+1)+k = 2k+1$. On the other hand, it is at least 5 because graphs of smaller order will never form an asymmetric partitionable cycle. □

Corollary 1. *In a bipartite graph $G = (X \cup Y, E)$, every connected 2-clique is a 2-club. Furthermore, every 2-club induces a complete bipartite subgraph.*

Proof. Since G is bipartite, it does not contain an odd cycle and hence, does not contain an asymmetric partitionable cycle of size 5. Any 2-club in G of size at least two, contains vertices $x \in X$ and $y \in Y$. If $(x, y) \notin E$ then $d_G(x, y) \geq 3$. Hence, the 2-club must induce a complete bipartite subgraph. □

5. Bounding strategies for the k-club number of a graph

As discussed in Section 1.1, the only available exact combinatorial algorithm for solving the maximum k-club problem for $k \geq 2$ is the branch-and-bound (BB) algorithm presented in [15]. In this approach, the authors find the k-clique number of the graph associated with the node of the BB tree to obtain an upper-bound. Note that the maximum k-clique problem must be solved to optimality to obtain a valid upper-bound, and it is NP-hard even on graphs of fixed diameter $k+1$. In the next section, we consider a dual to the maximum k-club problem so that any feasible solution of this dual could be used to obtain a valid upper-bound. In Section 5.2, we discuss a lower-bounding technique that builds on existing heuristic approaches for the problem [17].

5.1. Distance k-coloring based upper-bounding technique

Definition 4. Given $G = (V, E)$, $I \subseteq V$ is called a k-independent set if $d_G(u, v) \geq k + 1$ for all $u, v \in V$.

Note that 1-independent sets are the classical independent (stable) sets and the definition is restrictive as k increases. That is, every $k+1$-independent set is a k-independent set but not vice versa. Further, I is a k-independent set in G if and only if I is an independent set in the power graph G^k. Also note that if I is a k-clique in G, I need not be a k-independent set in the complement graph $\bar{G}$ for $k \geq 2$. For instance, {1, 2, 4, 5, 6} is a 2-clique/2-club in the graph in Fig. 1, but it is not a 2-independent set in the complement. In fact, it is still a 2-clique/2-club in the complement graph.

Definition 5. A proper distance k-coloring of $G = (V, E)$ is a map $c: V \longrightarrow \{1, \ldots, n\}$ such that for every pair of vertices $u, v \in V$, if $c(u) = c(v)$ then $d_G(u, v) \geq k + 1$ [32]. For each $v \in V$, $c(v)$ is called the color of v, and the subset of nodes receiving the same color form a color class.

Note that this definition reduces to classical graph coloring when $k = 1$ and is restrictive as k increases. Furthermore, every color class forms a k-independent set and hence, any k-club or k-clique in G can contain at most one vertex from a color class for every proper distance k-coloring of G. Denote the *minimum* number of colors to properly distance k-color G by $\chi_k^d(G)$. We have the following duality relationship for every fixed positive integer k:

$$\overline{\omega}_k(G) \leq \widetilde{\omega}_k(G) \leq \chi_k^d(G). \tag{4}$$

Fig. 5 shows an example of a proper distance 2-coloring of a graph using 5 colors. Finding a minimum distance k-coloring is also an NP-hard problem [32], but for obtaining valid upper-bounds, we only need a proper distance k-coloring. By noting that distance k-coloring is equivalent to classical graph coloring on power graphs, we can employ fast coloring heuristics to yield a proper distance k-coloring. A compromise must be made in the quality of the bound, as k-clique number which is a tighter bound can be hard to obtain, especially at the root node of the BB tree.

5.2. Bounded enumeration based lower-bounding technique

The proposed lower-bounding technique is based on finding an initial k-club S, followed by a tree search that enumerates k-clubs containing S. If left unchecked, this worst-case complete enumeration would return a maximal k-club containing S. However, we employ pruning and termination criteria that would prevent this procedure from consuming an excessive amount of time. The "drop" and "constellation" heuristics described in [17] are used to identify the initial k-club. Both heuristics are run and the larger of the two k-clubs found is used as the initial solution S. The goal of the bounded enumeration search is to improve upon this initial solution if possible by spending a reasonable amount of time.

Given an initial k-club S, consider the graph H initially equal to G. We recursively check H for a vertex v such that $d_H(v, u) > k$ for some $u \in S$ and delete v. Note that v cannot be a part of any k-club in G containing S. When this recursive

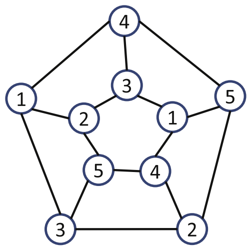

Fig. 5. A proper distance 2-coloring (each number represents a specific color class).

procedure terminates, $S \subseteq V(H)$ and $d_H(v, u) \leq k$ for all $u \in S$ and $v \in V(H) \setminus S$. Every k-club in G that contains S is contained in $V(H)$, and we call $V(H) \setminus S$ the candidate set.

In this bounded enumeration search, at each node of the search tree, the corresponding graph $\mathcal{G} = G[F \cup U]$ consists of two types of vertices. The first is the set of vertices fixed to be included denoted by F, the second is the set of unexplored vertices denoted by U. The root node of the search tree is initialized by setting $F = S$ and $U = V(H) \setminus S$. The search order is depth-first search (DFS), with a vertex in U with the largest number of vertices at distance k or less in $\mathcal{G}$ chosen first to be included. Note that the search order and the greedy selection rule are chosen to encourage detection of better feasible solutions.

If $V(H) \setminus S = \emptyset$ then there is no larger k-club that contains S. Now suppose $V(H) \setminus S \neq \emptyset$. For each integer $1 \leq l \leq |V(H) \setminus S|$, starting from $l = 1$, we initiate a bounded enumeration search with the goal of adding l vertices to the current feasible solution. If at any point in this search we are successful in adding l vertices, we update S, recompute the candidate set as described before, and repeat. If we are unsuccessful in this search for l vertices to add, either because it is not possible, or because we have reached a termination criterion specified in terms of number of tree nodes pruned, we increment l and restart the search with the same S and $V(H)$ as long as $l \leq |V(H) \setminus S|$. The overall procedure stops when either l exceeds $|V(H) \setminus S|$ or if a user-specified time limit is reached.

We maintain the following conditions for the graph $\mathcal{G}$ associated with any active tree node: (C1) F is a k-clique in $\mathcal{G}$; (C2) $d_{\mathcal{G}}(v, u) \leq k, \forall u \in F, v \in U$. Note that at each such node, $S \subseteq F$, $F \setminus S \subseteq V(H) \setminus S$, $U \subseteq V(H) \setminus S$ and vertices in $(V(H) \setminus S) \setminus (F \cup U)$ have been deleted leading to this node of the search tree. While processing a tree node created to include a vertex $i \in U$, if $|F \setminus S| = l$ then the present node is pruned since we are only interested in subsets of $V(H) \setminus S$ of size l. Otherwise, vertex i is added to F and removed from U. Note that $F \cup \{i\}$ will be a k-clique in $\mathcal{G}$. However, (C2) may be violated by some vertices too far from i in the new U. Thus every vertex $v \in U \setminus \{i\}$ violating (C2) is recursively deleted until no such vertex exists. Similarly, while processing a tree node created to delete a vertex $i \in U$, if $|(V(H) \setminus S) \setminus (F \cup U)| = |V(H) \setminus S| - l$ then the present tree node is pruned. Otherwise, after removing vertex i, every vertex $v \in U \setminus \{i\}$ violating (C2) is recursively deleted until no such vertex exists.

In any of these two cases, if removing these vertices results in (C1) being violated, the present node is pruned by infeasibility. If (C1) and (C2) are still satisfied and $U = \emptyset$ or if $d_{\mathcal{G}}(u, v) \leq k, \forall u, v \in U$ then $\mathcal{G}$ is a k-club. In this case, the lower-bounding procedure is terminated and restarted with the initial solution $S = F \cup U$, the associated candidate set $V(H) \setminus S$ and $l = 1$. Otherwise, two new child nodes are created by using the branching strategy and added to the list of active search tree nodes. After processing a tree node, this node will be removed from the set of active nodes. Whenever the set of active nodes is empty, the bounded enumeration for the present value of l terminates. After terminating the search for the subsets of $V(H) \setminus S$ with size l, if $l = |V(H) \setminus S|$ then the lower-bounding algorithm terminates otherwise, a new bounded enumeration starts for the subsets of size $l + 1$. The overall lower-bounding procedure is also limited by a user-specified time limit.

6. Branch-and-bound framework to find the *k*-club number of a graph

In order to study the effectiveness of the bounding techniques proposed in Section 5, we incorporate them in a BB algorithm. Two lower-bounding and two upper-bounding schemes (in total four combinations) are tested. Given a graph $G = (V, E)$ and an integer $k \geq 2$, the first lower-bounding technique is to select the larger of the two solutions found by drop and constellation heuristics (denoted by *DC*). The second technique is the bounded enumeration based lower-bounding technique (denoted by *BE*) proposed in Section 5.2 which will return a solution at least as good as the first. The lower-bounding scheme is only used once at the beginning of the BB algorithm in order to initialize the incumbent solution.

The upper-bounding scheme is used at each node of the BB tree to find an upper-bound on the k-club number of the graph $\mathcal{G}$ associated with that node. The first technique is to use $\widetilde{\omega}_k(\mathcal{G})$ as the upper-bound on $\overline{\omega}_k(\mathcal{G})$ (denoted by *KC*). To find $\widetilde{\omega}_k(\mathcal{G})$, the maximum clique problem is solved on $\mathcal{G}^k$ by using Östergård's algorithm [31]. The second technique is the distance k-coloring based upper-bounding technique (denoted by *CO*) introduced in Section 5.1. To properly color $\mathcal{G}^k$, two heuristics are employed. First, a simple greedy heuristic that repeatedly colors the largest degree uncolored node in the power graph with the color not yet assigned to any of its neighbors. Second, we use the well known DSATUR heuristic [33] for graph coloring. In general, the greedy heuristic is faster than DSATUR, but the quality of the solution found by DSATUR is better. Since finding tighter upper-bounds in top levels of the search tree is much more critical, we use DSATUR while processing higher tree levels and use the greedy heuristic for finding upper-bounds in lower levels. A threshold parameter was used to determine the tree level after which the upper-bounding heuristic should be switched from DSATUR to greedy algorithm.

The structure of each BB tree node is similar to the one used for the proposed *BE* algorithm with the difference that the root node of the BB tree is initialized by setting $F = \emptyset$ and $U = V$. A vertex dichotomy branching rule is also employed in this algorithm. Based on the preliminary computational results, selecting a vertex in U with minimum number of vertices at distance at most k in $\mathcal{G}$ for branching along with the best bound search (BBS) strategy performed better than other branching and search strategies. The reason for this observation is that in this BB algorithm, unlike the proposed *BE* technique which focuses on feasibility, the emphasis is on finding a maximum k-club in G and proving optimality. In case of a tie in choosing the branching vertex, the vertex with minimum degree in $\mathcal{G}$ will be selected to branch upon. If a tie still exists, the branching vertex will be selected randomly from tied vertices. Properties (C1) and (C2), mentioned in Section 5.2, are also maintained during the course of this BB algorithm. Note that at each node of the BB search tree, vertices in $V \setminus (F \cup U)$ have been deleted along the path connecting the root node of the search tree to the current node.

While processing a BB tree node, if the corresponding upper-bound is less than or equal to the size of the incumbent k-club, the node is then pruned by bound. Otherwise, if the BB node is created to delete a vertex $i \in U$, this vertex is removed from U and every vertex $v \in U \setminus \{i\}$ violating (C2) is recursively deleted until no such vertex exists. Similarly, if the BB node is created to include a vertex $i \in U$, this vertex is added to F and removed from U. Note that the new F will satisfy (C1) but (C2) can be violated by some vertices in $U \setminus \{i\}$ which are too far from i. Thus every vertex $v \in U \setminus \{i\}$ violating (C2) is recursively deleted until no such vertex exists.

If deleting these vertices in any of these two cases results in (C1) being violated, the current BB tree node is pruned by infeasibility. Note that it would be incorrect to prune this node by infeasibility if $\text{diam}(G[F]) > k$. We employ the necessary condition that F must be a k-clique in $\mathcal{G}$ for this purpose. If (C1) and (C2) are still satisfied and $U = \emptyset$ or if $d_{\mathcal{G}}(u, v) \le k, \forall u, v \in U$ then $\mathcal{G}$ is a k-club. So the current BB tree node is pruned by feasibility and the incumbent solution is updated if necessary. Otherwise, the upper-bounding technique is used to find an upper-bound on the k-club number of the new $\mathcal{G}$. Again, if the estimated upper-bound is less than or equal to the size of the incumbent solution, the node is fathomed by bound otherwise by using the branching strategy, two new child nodes are created and added to the list of active BB tree nodes. The processed BB node then will be removed from the set of active BB nodes. Whenever the set of active BB nodes is empty, the BB algorithm terminates and the size of the incumbent solution will be equal to the k-club number of G. In addition to this termination criterion, a maximum time allowed is also used to stop the BB algorithm. In case of termination by time limit, the gap between the best upper-bound (the largest upper-bound among all active BB nodes) and the incumbent solution is reported.

Note that for any active node, to compute the upper-bound by distance coloring $\mathcal{G}$, we only need to distance color $G[U]$ as F is a k-clique and every vertex in U is at distance no more than k from every vertex in F. That is, we need $|F|$ colors to color nodes in F and none of these can be used to color nodes in U. The upper-bound for the active node under consideration is equal to the number of colors used to color $G[U]$ plus the cardinality of set F. Another key operation which has a clear effect on the algorithm running time is updating the distance k-neighborhood of each vertex of the graph obtained by deleting vertices. Given a graph $G = (V, E)$ and an integer $k \ge 2$, the distance k-neighborhood of $v \in V$ is the set $N_G^k(v) = \{u \in V : d_G(v, u) \le k\} = N_{G^k}(v)$, that is the neighborhood of v in the power graph G^k. After deleting a vertex i, we only need to update the k-neighborhood of vertices in $N_{\mathcal{G}}^k(i)$. This observation also helps reduce the time taken to update pairwise distances in $\mathcal{G} - i$, after vertex deletion.

7. Implementation details and computational test results

All algorithms were implemented in C++ and all numerical experiments were conducted on a Dell® workstation with Intel® Xeon® W 3550 @ 3.07 GHz processor and 3.00 GB RAM. The test-bed of instances consists of graphs generated randomly by using the algorithm introduced in [15]. The edge density of the graphs produced by this algorithm is controlled by two parameters a and b. The expected edge density d is $(a + b)/2$ and vertex degree variance (VDV) increases with $b - a$. We considered $k = 2$ and 3 and edge densities $d = 0.0125, 0.025, 0.05, 0.1, 0.15, 0.2$ and 0.25 were studied. The experiments were performed on graphs of order $n = 50, 100, 150$ and 200. For each order and density, 10 samples with minimum VDV ($a = b = d$) and 10 samples with maximum VDV ($a = 0, b = 2d$) were generated.

It should be noted that we do not explicitly require the test instances to be connected. Especially the sparse instances could consist of multiple connected components in which case the maximum k-club problem can be decomposed by component. This would be particularly useful to recognize in integer programming approaches, and is implicitly used in our BB algorithm. Also important to note is the fact that the maximum k-club may or may not be in the largest order or the most dense connected component. Table 1 shows the average size of the largest connected component in generated test instances. For a given n and for low edge densities, the average size of the largest connected component in instances with minimum VDV is larger than the one for instances with maximum VDV (except for $n = 50$ and density 0.025) and this difference disappears as the edge density increases.

We also observed that some of the higher order, higher density test instances were trivial since their diameter was less than or equal to the value of k under consideration. While this is expected in general, it was interesting to note that in

Table 1
Average size of the largest connected component in generated test instances.

n	VDV	Edge density						
		0.0125	0.025	0.05	0.1	0.15	0.2	0.25
50	Min	5.6	16.6	43.6	49.5	50	50	50
	Max	4.4	21.6	41.8	48.8	49.1	50	50
100	Min	37.2	91.1	99.7	100	100	100	100
	Max	29.4	87.6	99.4	100	100	100	100
150	Min	112.5	147.6	149.9	150	150	150	150
	Max	109	144	149.6	150	150	150	150
200	Min	178.4	198	200	200	200	200	200
	Max	172.7	197.3	199.9	200	200	200	200

Table 2
Number of trivial test instances in each sample of 10 instances.

k	n	VDV	Edge density			
			0.1	0.15	0.2	0.25
2	150	Min	–	–	–	7
		Max	–	–	–	–
	200	Min	–	–	–	10
		Max	–	–	–	1
3	50	Min	–	2	10	10
		Max	–	1	1	10
	100	Min	–	10	10	10
		Max	–	7	10	10
	150	Min	9	10	10	10
		Max	1	10	10	10
	200	Min	10	10	10	10
		Max	7	10	10	10

Table 3
Challenging densities (among the ones considered) where both *BE* and *DC* took the maximum time among all densities.

k	$n = 50$	$n = 100$	$n = 150$	$n = 200$
2	0.25	0.2	0.15, 0.2	0.15
3	0.1	0.1	0.05	0.05

such cases, for any given k, the minimum VDV scheme appears to have a propensity for producing trivial instances. This is attributable to the fact that when VDV is maximum, there is potential for both large k-clubs (higher degree vertices) as well as isolated vertices. Hence, the entire graph may not have low diameter. We have summarized the number of such trivial instances in Table 2.

7.1. *Experiments with the lower-bounding techniques*

We first consider the numerical results comparing the two lower-bounding techniques *DC* and *BE* discussed in Section 6. The maximum time allowed for *BE* is 600 s and the number of nodes pruned during *BE* is limited to 200 for each subset size. We summarize the key observations here. For a particular k, VDV and n, the average solution size increases as expected with edge density, and more interestingly, the average running time reaches a peak and then drops. This peak occurs at the same or consecutive densities for both lower-bounding techniques indicating that finding good feasible solutions is challenging in these instances. Furthermore for a given k, we find the densities that are challenging to be the same across minimum and maximum VDV instances, and to be decreasing as n increases. This information is summarized in Table 3.

The main result of these experiments is that the *BE* approach is beneficial compared to *DC* when used on such challenging densities (especially on 150 and 200 vertex instances) as opposed to the non-challenging ones where the *DC* approach is preferable. Recall that *BE* approach will take more time as it includes *DC* in it, but has the potential for returning larger k-clubs. Table 4 summarizes the results of these experiments.

7.2. *Experiments with the branch-and-bound framework*

The remainder of our computational experiments are designed to study the performance of four different BB algorithms obtained by using different strategies for lower-bounding and upper-bounding. The four BB algorithms are denoted by *DC*/*CO*, *DC*/*KC*, *BE*/*CO* and *BE*/*KC* (see Section 6) in the numerical results. Tables 5 and 6 report the computational results obtained by solving the maximum k-club problem on 200-vertex graphs for $k = 2$ and $k = 3$, respectively. Average best

Table 4
Minimum and maximum percentage increase in average best objective value found by *BE* over *DC*, and increase in average running time in seconds for the challenging densities (over 10 samples).

k	Metric	$n = 50$	$n = 100$	$n = 150$	$n = 200$
2	Best obj	(1.48, 2.53)	(0.71, 3.92)	(0.25, 5.31)	(2.06, 17.44)
	Time	(0.75, 0.75)	(8.20, 10.20)	(15.51, 36.93)	(98.81, 218.45)
3	Best obj	(0.69, 1.59)	(0.00, 0.11)	(4.22, 7.26)	(3.03, 5.35)
	Time	(0.68, 0.70)	(7.10, 7.64)	(25.24, 30.00)	(143.55, 159.93)

Table 5
Average size of the best 2-club found, average running time (in seconds), and percentage optimality gap for each BB algorithm on 200-vertex instances.

VDV	Metric	Algorithm	Edge density						
			0.0125	0.025	0.05	0.1	0.15	0.2	0.25
Min	Best obj	*DC*/*CO*	8.20	12.70	20.70	33.60	58.50	195.50	200.00
		DC/*KC*	8.20	12.70	20.70	33.60	58.50	195.50	200.00
		BE/*CO*	8.20	12.70	20.70	33.60	68.70	195.50	200.00
		BE/*KC*	8.20	12.70	20.70	33.60	68.70	195.50	200.00
	Time	*DC*/*CO*	2.54	71.62	298.98	3604.30	3613.16	880.00	0.00
		DC/*KC*	0.95	2.12	16.71	4554.49	5276.76	331.06	0.00
		BE/*CO*	2.65	71.93	300.04	3605.00	3618.83	913.00	0.00
		BE/*KC*	1.01	2.53	18.18	4548.06	4542.85	361.98	0.00
	Gap	*DC*/*CO*	0.00	0.00	0.00	37.78	56.14	0.00	0.00
		DC/*KC*	0.00	0.00	0.00	83.20	70.75	0.00	0.00
		BE/*CO*	0.00	0.00	0.00	37.66	48.78	0.00	0.00
		BE/*KC*	0.00	0.00	0.00	83.20	65.65	0.00	0.00
Max	Best obj	*DC*/*CO*	9.20	14.30	23.40	39.10	126.20	178.20	196.40
		DC/*KC*	9.20	14.30	23.40	39.10	126.20	177.20	196.40
		BE/*CO*	9.20	14.30	23.40	39.10	128.80	178.20	196.40
		BE/*KC*	9.20	14.30	23.40	39.10	128.80	177.40	196.40
	Time	*DC*/*CO*	2.61	48.14	299.68	3608.07	3403.16	1664.25	777.07
		DC/*KC*	0.96	2.24	42.41	4884.64	4377.61	5603.51	116.33
		BE/*CO*	2.68	48.07	300.83	3604.49	3411.23	1747.34	813.20
		BE/*KC*	1.03	2.67	44.07	4878.12	4395.17	5606.31	151.69
	Gap	*DC*/*CO*	0.00	0.00	0.00	29.77	8.00	0.06	0.00
		DC/*KC*	0.00	0.00	0.00	80.45	36.90	10.35	0.00
		BE/*CO*	0.00	0.00	0.00	29.34	6.05	0.06	0.00
		BE/*KC*	0.00	0.00	0.00	80.45	35.60	10.25	0.00

objective value, running time and optimality gap across the 10 samples in each category of test instances are reported in these tables.

Although we present the results for 200-vertex instances here, the result are similar across different orders and the subsequent observations are made considering all the instances in our test bed. The complete set of computational results including instances of smaller orders ($n = 50$, 100 and 150) can be obtained from the corresponding author.

The running time limit for all algorithms is 3600 s. This was enforced by checking the elapsed time after each BB node was processed, which in some instances exceeded the time limit by the time it took to process the last BB node before termination. If an instance cannot be solved to optimality within the time limit, relative optimality gap is reported. The percentage gap is calculated as $100 \times (\text{upper} - \text{bound} - \text{bestsolutionsize})/\text{upper} - \text{bound}$. The settings used for *BE* are as stated before. The tree level threshold parameter for switching the upper-bounding heuristic from DSATUR to simple greedy is set at $0.10 \times n$.

For a given k, VDV and n, as edge density increases the average incumbent solution size found by all algorithms increases, while generally the average running time and optimality gap increase up to a peak and then decrease. The peak average running time can be used to determine the challenging densities for these BB algorithms. It can be observed that for a given k, the challenging densities are the same for both minimum and maximum VDV instances, and decrease as n increases. Table 7 identifies these challenging densities for all BB algorithms.

For a given k, n, VDV and for densities that are not challenging, the quality of the solution found by all algorithms are nearly identical. Considering average running time for these non-challenging densities, *DC*/*KC* almost always performs better than the other algorithms. This observation is attributable to the fact that on these densities, *BE* does not significantly improve the quality of the initial solution. Furthermore, it appears that solving the maximum k-clique problem which returns potentially tighter upper-bounds is not as hard on these densities. Likewise for these non-challenging densities, all the algorithms return a zero optimality gap on nearly all the instances.

However, for challenging densities, *BE*/*CO* outperforms all the other algorithms in terms of solution quality. This observation can be explained by the fact that on these densities, *BE* often returns a larger initial solution in a reasonable

Table 6
Average size of the best 3-club found, average running time (in seconds), and percentage optimality gap for each BB algorithm on 200-vertex instances.

VDV	Metric	Algorithm	Edge density						
			0.0125	0.025	0.05	0.1	0.15	0.2	0.25
Min	Best obj	DC/CO	13.60	25.40	89.70	200.00	200.00	200.00	200.00
		DC/KC	13.60	20.70	89.70	200.00	200.00	200.00	200.00
		BE/CO	13.60	25.40	94.50	200.00	200.00	200.00	200.00
		BE/KC	13.60	23.40	94.50	200.00	200.00	200.00	200.00
	Time	DC/CO	87.90	442.32	3616.94	0.00	0.00	0.00	0.00
		DC/KC	17.24	3684.29	4108.80	0.00	0.00	0.00	0.00
		BE/CO	88.28	431.66	3619.46	0.00	0.00	0.00	0.00
		BE/KC	17.98	3702.84	4160.96	0.00	0.00	0.00	0.00
	Gap	DC/CO	0.00	0.00	31.64	0.00	0.00	0.00	0.00
		DC/KC	0.00	89.30	55.15	0.00	0.00	0.00	0.00
		BE/CO	0.00	0.00	28.13	0.00	0.00	0.00	0.00
		BE/KC	0.00	87.92	52.75	0.00	0.00	0.00	0.00
Max	Best obj	DC/CO	14.80	31.90	119.10	199.70	200.00	200.00	200.00
		DC/KC	14.80	24.90	118.70	199.70	200.00	200.00	200.00
		BE/CO	14.80	31.90	122.50	199.70	200.00	200.00	200.00
		BE/KC	14.80	29.40	122.30	199.70	200.00	200.00	200.00
	Time	DC/CO	91.39	477.40	3275.85	262.39	0.00	0.00	0.00
		DC/KC	63.03	3760.07	4523.58	32.50	0.00	0.00	0.00
		BE/CO	91.75	464.38	3304.12	268.81	0.00	0.00	0.00
		BE/KC	62.81	3757.28	4526.92	38.81	0.00	0.00	0.00
	Gap	DC/CO	0.00	0.00	12.29	0.00	0.00	0.00	0.00
		DC/KC	0.00	87.39	40.65	0.00	0.00	0.00	0.00
		BE/CO	0.00	0.00	9.64	0.00	0.00	0.00	0.00
		BE/KC	0.00	85.11	38.85	0.00	0.00	0.00	0.00

Table 7
Challenging densities (among the ones considered) where all four BB algorithms took the maximum time across all densities.

k	$n = 50$	$n = 100$	$n = 150$	$n = 200$
2	0.2, 0.25	0.15, 0.2	0.15, 0.2	0.1, 0.15, 0.2
3	0.1	0.05	0.05	0.05

amount of time. Furthermore, solving the maximum k-clique problem to optimality on these densities is relatively harder than finding a feasible distance k-coloring, requiring longer times to compute the upper-bound at each node. This results in fewer BB nodes being enumerated in a specified time limit, affecting the incumbent solution quality adversely. Due to similar reasons, both DC/CO and BE/CO perform better than the other two algorithms in terms of average running time metric. In terms of optimality gap, DC/CO and BE/CO outperform the other two algorithms while BE/CO is slightly better than DC/CO.

8. Conclusion

The k-clubs can be used to model low diameter clusters in graph-based data mining applications. In this paper, k-club maximality testing is proved to be NP-complete for $k \geq 2$, thus settling a long-open problem. This is due to the nonhereditary nature of k-clubs, and we discuss the implications of this property in detail. A class of graphs for which k-club maximality testing is polynomial-time solvable have also been identified. A new dual coloring based upper-bounding technique and a new bounded enumeration based lower-bounding strategy for the k-club number of a graph have been proposed. To solve the maximum k-club problem, a combinatorial branch-and-bound framework is developed and the computational performance of the proposed branch-and-bound framework with four different combinations of lower- and upper-bounding schemes is studied. It will be beneficial to develop metaheuristic approaches for detecting k-clubs in large-scale real-life graphs, especially power-law graphs from bioinformatics applications. Identifying graph classes on which detecting maximum and maximal k-clubs is polynomial-time solvable is another interesting challenge.

Acknowledgments

This research was supported in part by the US Department of Energy Grant DE-SC0002051. The authors would like to thank the anonymous referees for several comments that improved this article.

References

[1] J. Scott, Social Network Analysis: A Handbook, second ed., Sage Publications, London, 2000.
[2] I.M. Bomze, M. Budinich, P.M. Pardalos, M. Pelillo, The maximum clique problem, in: D.Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp. 1–74.
[3] S. Butenko, W. Wilhelm, Clique-detection models in computational biochemistry and genomics, European Journal of Operational Research 173 (2006) 1–17.
[4] V. Boginski, S. Butenko, P. Pardalos, Mining market data: a network approach, Computers & Operations Research 33 (2006) 3171–3184.
[5] J. Abello, P. Pardalos, M. Resende, On maximum clique problems in very large graphs, in: J. Abello, J. Vitter (Eds.), External Memory Algorithms and Visualization, in: DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 50, American Mathematical Society, 1999, pp. 119–130.
[6] S.B. Seidman, B.L. Foster, A graph theoretic generalization of the clique concept, Journal of Mathematical Sociology 6 (1978) 139–154.
[7] R. Luce, Connectivity and generalized cliques in sociometric group structure, Psychometrika 15 (1950) 169–190.
[8] R. Alba, A graph-theoretic definition of a sociometric clique, Journal of Mathematical Sociology 3 (1973) 113–126.
[9] R. Mokken, Cliques, clubs and clans, Quality and Quantity 13 (1979) 161–173.
[10] D. Corneil, Y. Perl, Clustering and domination in perfect graphs, Discrete Applied Mathematics 9 (1984) 27–39.
[11] S.S. Ravi, D. Rosenkrantz, G.K. Tayi, Heuristics and special case algorithms for dispersion problems, Operations Research 42 (1994) 299–310.
[12] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979.
[13] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, Acta Mathematica 182 (1999) 105–142.
[14] R.G. Downey, M.R. Fellows, Fixed-parameter tractability and completeness II: on completeness for $W[1]$, Theoretical Computer Science 141 (1995) 109–131.
[15] J.M. Bourjolly, G. Laporte, G. Pesant, An exact algorithm for the maximum k-club problem in an undirected graph, European Journal of Operational Research 138 (2002) 21–28.
[16] B. Balasundaram, S. Butenko, S. Trukhanov, Novel approaches for analyzing biological networks, Journal of Combinatorial Optimization 10 (2005) 23–39.
[17] J.M. Bourjolly, G. Laporte, G. Pesant, Heuristics for finding k-clubs in an undirected graph, Computers & Operations Research 27 (2000) 559–569.
[18] S. Butenko, O. Prokopyev, On k-club and k-clique numbers in graphs, Technical Report, Texas A&M University, 2007.
[19] J. Marincek, B. Mohar, On approximating the maximum diameter ratio of graphs, Discrete Mathematics 244 (2002) 323–330.
[20] Y. Asahiro, E. Miyano, K. Samizo, Approximating maximum diameter-bounded subgraphs, in: A. Lpez-Ortiz (Ed.), LATIN 2010: Theoretical Informatics, in: Lecture Notes in Computer Science, vol. 6034, Springer, Berlin/Heidelberg, 2010, pp. 615–626.
[21] A. Schäfer, C. Komusiewicz, H. Moser, R. Niedermeier, Parameterized computational complexity of finding small-diameter subgraphs, Optimization Letters (2011) 1–9. doi:10.1007/s11590-011-0311-5.
[22] A. Schäfer, Exact algorithms for s-club finding and related problems, Master's thesis, Diplomarbeit, Institut für Informatik, Friedrich–Schiller–Universität Jena, 2009.
[23] F.D. Carvalho, M.T. Almeida, Upper bounds and heuristics for the 2-club problem, European Journal of Operational Research 210 (2011) 489–494.
[24] M.T. Almeida, F.D. Carvalho, The k-club problem: new results for $k = 3$, Technical Report CIO Working Paper 3/2008, CIO-Centro de Investigação Operacional, 2008.
[25] A. Veremyev, V. Boginski, Identifying large robust network clusters via new compact formulations of maximum k-club problems, European Journal of Operational Research 218 (2012) 316–326.
[26] M. Yannakakis, Node-and edge-deletion NP-complete problems, in: STOC'78: Proceedings of the 10th Annual ACM Symposium on Theory of Computing, ACM Press, New York, NY, 1978, pp. 253–264.
[27] J.G. Oxley, Matroid Theory, Oxford University Press, Oxford, UK, 1992.
[28] R. Euler, M. Jünger, G. Reinelt, Generalizations of cliques, odd cycles and anticycles and their relation to independence system polyhedra, Mathematics of Operations Research 12 (1987) 451–462.
[29] S.P. Borgatti, M.G. Everett, L.C. Freeman, UCINET 5 for Windows: Software for Social Network Analysis-User's Guide, Analytic Technologies, 1999, Accessed Jan 2010 http://www.analytictech.com/ucinet6/Ucinet_Guide.doc.
[30] R. Carraghan, P. Pardalos, An exact algorithm for the maximum clique problem, Operations Research Letters 9 (1990) 375–382.
[31] P.R.J. Östergård, A fast algorithm for the maximum clique problem, Discrete Applied Mathematics 120 (2002) 197–207.
[32] S.T. McCormick, Optimal approximation of sparse hessians and its equivalence to a graph coloring problem, Mathematical Programming 26 (1983) 153–171.
[33] D. Brélaz, New methods to color the vertices of a graph, Communications of the ACM 22 (1979) 251–256.