Computers & Operations Research 52 (2014) 94–104

Contents lists available at ScienceDirect

Computers & Operations Research

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

Two-phase heuristics for the *k*-club problem

Maria Teresa Almeida [a,b,*], Filipa D. Carvalho [a,b]

[a] ISEG, Universidade de Lisboa, Rua do Quelhas 6, 1200-781 Lisboa, Portugal
[b] CIO, FC, Universidade de Lisboa, Bloco C6, Piso 4, 1749-016 Lisboa, Portugal

ARTICLE INFO

Available online 17 July 2014

Keywords:
Heuristics
Maximum *k*-club
Clique relaxations
Social network analysis
Computational biology

ABSTRACT

Given an undirected graph G and an integer k, a k-club is a subset of nodes that induces a subgraph with diameter at most k. The k-club problem consists of identifying a maximum cardinality k-club in G. It is an NP-hard problem. The problem of checking if a given k-club is maximal or if it is a subset of a larger k-club is also NP-hard, due to the non-hereditary nature of the k-club structure. This non-hereditary nature is adverse for heuristic strategies that rely on single-element add and delete operations. In this work we propose two-phase algorithms which combine simple construction schemes with exact optimization of restricted integer models to generate near optimal solutions for the k-club problem. Numerical experiments on sets of uniform random graphs with edge densities known to be very challenging and test instances available in the literature indicate that the new algorithms are quite effective, both in terms of solution quality and running times.

1. Introduction

During the last decade, the use of network models has been steadily growing in a large number of areas, *e. g.*, social network analysis, computational biology, and financial information data mining, [1–5].

In all these areas it is important to identify groups of elements that form clusters. The characterization of a cluster may be made from several standpoints. In some cases, the most important aspect is the number (or the proportion) of direct links between pairs of elements. In other cases, the most important feature is the number of hops that separate pairs of elements, *i. e.*, the number of intermediate elements that are needed to establish a connection between any two members. The former case can be addressed with density-based network models such as cliques, quasi-cliques, k-cores and k-plexes [6–13]. The latter case can be addressed with diameter-based network models such as k-clubs.

Given an undirected graph $G=(V,E)$ and a pair of nodes $u,v \in V$, the distance $dist_G(u,v)$ is the minimum number of edges needed to link u and v in G. The diameter of G $diam(G)$ is the maximum distance between any pair of nodes. Given $G=(V,E)$ and a positive integer k, a k-club is a subset of nodes $S \subseteq V$ that induces a subgraph $G[S]=(S,E(S))$ with a diameter of at most k and a k-clique is a set of nodes $C \subseteq V$ such that $dist_G(u,v) \leq k$, for all $u,v \in C$. The k-club (resp. k-clique) number of G is the number of nodes in a maximum k-club (resp. k-clique) in G. For $k=1$, k-clubs and k-cliques induce complete subgraphs, *i. e.*, cliques. For any positive integer $k > 1$, a k-club is a k-clique, but the converse is not true – since $dist_G(u,v) \leq dist_{G[C]}(u,v)$, for all $u,v \in C$, the diameter of the subgraph induced by a k-clique C may be greater than k.

The k-club problem consists of finding a maximum cardinality k-club in an undirected graph. It is known to be an NP-hard problem, for any $k > 1$ (Bourjolly et al. [14]). Mahdavi Pajouh and Balasundaram [15] discuss in full length the non-hereditary nature of k-clubs and prove that checking the inclusionwise maximality of a k-club is also an NP-hard problem.

The rest of the paper is organized as follows. Section 2 provides the background for the new heuristics and reviews recently published work on the k-club problem. Sections 3, 4, and 5 contain the detailed description of the two-phase heuristics and the motivation for them. The computational results are presented in Section 6 and the conclusions in Section 7.

2. Background and related work

In this section we briefly review heuristic algorithms and integer models that are used in the two-phase algorithms proposed in Sections 4 and 5. We also survey related work recently published in the literature.

* Correspondence to: Instituto Superior de Economia e Gestão, Departamento de Matemática, Rua do Quelhas 6, 1200-781 Lisboa, Portugal. Tel.: +351 213 925 800; fax: +351 213 922 781.
E-mail addresses: talmeida@iseg.utl.pt (M.T. Almeida), filipadc@iseg.utl.pt (F.D. Carvalho).

http://dx.doi.org/10.1016/j.cor.2014.07.006
0305-0548/

2.1. Heuristic algorithms

Bourjolly et al. [16] proposed three simple heuristics for the k-club problem: Constellation, Drop, and k-Clique and Drop.

Constellation is a constructive heuristic that starts with a star centred at a maximum degree node – which is a k-club for any $k > 1$ – and iteratively adds new nodes while the induced subgraph has a diameter smaller than k. It can be summarized as follows:

Heuristic Constellation

Step 1: Set $t:=2$. The initial 2-club is the set W of nodes of a star graph centred at a maximum degree node.
Step 2: If $t=k$ or $W=V$, stop. Otherwise set $t:=t+1$. Determine the node of W having the largest set S of neighbours in $V\backslash W$ and set $W:=W \cup S$. Repeat step 2.

If heuristic Constellation is halted upon completion of step 1, it returns a star graph. We will call this procedure heuristic Star. Note that, for $k=2$, heuristic Constellation hits the stopping criterion before executing step 2 and reduces to heuristic Star.

Drop is an elimination heuristic that starts with the whole node set V and then removes one node at a time to obtain a subset that induces a k-club. It can be summarized as follows:

Heuristic Drop

Step 1: Compute shortest chain lengths between all node pairs.
Step 2: Compute for each node i of V the number q_i of nodes of V whose shortest chain to i has length at least $k+1$. If $q_i=0$, for all i, stop.
Step 3: Let $W=\{i \in V: q_i \text{ is maximum}\}$ and i^* be a minimum degree node in W. Remove i^* from V and eliminate all edges incident to it.
Step 4: Update shortest chain lengths and go to step 2.

For $k=2$, Carvalho and Almeida [17] proposed an alternative node elimination criterion based on optimal solutions of a strong linear programming model for the 2-club problem.

Given a graph $G=(V,E)$ and an integer $k > 1$, a subset $S \subseteq V$ is a k-clique if $dist_G(u,v) \le k$, for all $u,v \in S$, and it is a k-club if $dist_{G[S]}(u,v) \le k$, for all $u,v \in S$. Since $dist_G(u,v) \le dist_{G[S]}(u,v)$, for all $u,v \in S$, k-cliques may be interpreted as relaxations of k-clubs and large k-cliques are good candidates to contain large k-clubs. This is the basis of heuristic k-Clique and Drop, which can be summarized as follows:

Heuristic k-Clique and Drop

Step 1: Find a largest k-clique.
Step 2: Remove all nodes not in the k-clique and their incident edges and call heuristic Drop.

Shahinpour and Butenko [18] proposed a variable neighbourhood search (VNS) heuristic for the k-club problem that uses Constellation, Drop and Expand solutions as starting points. Expand is a heuristic similar to k-Clique and Drop which finds the k-neighbourhood $N_G^k(v)=\{j \in V : dist_G(j,v) \le k\}$ of a node $v \in V$ instead of a k-Clique. Given a k-club S, they define four neighbourhoods of S, obtained by adding and/or removing nodes, and two procedures, 1-add and 2-add moves, whose goal is to improve a given k-club.

2.2. Integer models

The k-club problem may be formulated as an integer linear programming problem in several different variable spaces. For the sake of brevity, we shall review in detail only the models that are used in Sections 4 and 5.

To simplify the notation, the set $\{\{i,j\} : i,j \in V,\ dist_G(i,j) > k\}$ of all pairs of nodes that cannot belong simultaneously to a k-club is denoted by P^k and the set $\{\{i,j\} : i,j \in V,\ (i,j) \notin E,\ dist_G(i,j) \le k\}$ of all pairs of nonadjacent nodes whose distance in G does not exceed k is denoted by N^k.

Bourjolly et al. [14] proposed a formulation with node and chain variables and showed that, for the case $k=2$, the formulation may be simplified by eliminating all chain variables – since any chain (u, w, v) with length two that links a pair of nonadjacent nodes u and v can be identified by its central node w. The simplified model for the case $k=2$ can be presented as follows. Let x_i be binary variables such that $x_i = 1$ if and only if node i is included in the 2-club. A maximum 2-club is an optimal solution of the following integer linear programming problem:

$$Max \sum_{i \in V} x_i$$

s.t.

$$x_i + x_j \le 1 \quad \{i,j\} \in \mathrm{P}^2 \tag{1}$$

$$x_i + x_j \le 1 + \sum_{r \in (N_i \cap N_j)} x_r \quad \{i,j\} \in \mathrm{N}^2 \tag{2}$$

$$x_i \in \{0,1\} \quad i \in V \tag{3}$$

Constraints (1) impose that if $dist_G(i,j) > 2$, then nodes i and j are not both included in the 2-club. Constraints (2) impose that if $dist_G(i,j) = 2$ then to include these two nodes in the 2-club it is necessary to include also at least one of their common neighbours. Constraints (3) define the variables as binary.

Veremyev and Boginski [19] developed a formulation with recursive path variables and introduced the new concept of robust k-club: a k-club that verifies the stronger condition that every pair of nodes must be linked by at least R node-disjoint paths with at most k edges.

For $k=3$, Almeida and Carvalho [20] proposed two integer formulations denoted by [F_S] and [F_N]. To present them, it is necessary to define for each pair of nodes $\{i,j\} \in \mathrm{N}^3$ the set $E_{ij} = \{(p,q) \in E : p \in (N_i \setminus N_j), q \in (N_j \setminus N_i)\}$ of the inner edges of chains that link i and j, have three edges and whose inner nodes are not in $N_i \cap N_j$.

[F_S] is a node cut set-based model with variables defined as in Bourjolly et al. [14]. For each pair of nodes $\{i,j\} \in \mathrm{N}^3$, let $\mathbf{S}^{ij}$ denote the set of all minimal sets $S_{ij} \subseteq V$ that intersect all chains underlying the definition of edge set E_{ij}. A maximum 3-club is an optimal solution of the following integer linear programming problem:

$$Max \sum_{i \in V} x_i$$

s.t.

$$x_i + x_j \le 1 \quad \{i,j\} \in \mathrm{P}^3 \tag{4}$$

$$x_i + x_j \le 1 + \sum_{r \in (N_i \cap N_j)} x_r + \sum_{s \in S_{ij}} x_s \quad (i,j) \in \mathrm{N}^3, S_{ij} \in \mathbf{S}^{ij} \tag{5}$$

$$x_i \in \{0,1\} \quad i \in V \tag{3}$$

Constraints (4) impose that if $dist_G(i,j) > 3$ then nodes i and j are not both included in the 3-club. Constraints (5) impose that if a pair $\{i,j\} \in \mathrm{N}^3$ is included in the 3-club, then a common neighbour of i and j or one node from each $S_{ij} \in \mathbf{S}^{ij}$ is also included in the

3-club, since otherwise $dist_{G[S]}(i,j) > 3$. Almeida and Carvalho [20] also presented a compact integer relaxation of [F_S], obtained by eliminating for each pair $\{i,j\} \in N^3$ all constraints (5) except one. This relaxation, denoted by [F_S_{IR}], dominates in the theoretical sense the k-clique model (viewed as an integer relaxation of the k-club model). [F_S_{IR}] is one of the components of algorithms 5.2 and 5.3 (see Section 5.2).

[F_N] is a compact formulation, with one binary variable x_i for each node $i \in V$ and one binary variable z_{ij} for each edge $(i,j) \in E$:

$$Max \sum_{i \in V} x_i$$

s.t.

$$x_i + x_j \leq 1 \quad \{i,j\} \in P^3 \tag{4}$$

$$x_i + x_j \leq 1 + \sum_{r \in (N_i \cap N_j)} x_r + \sum_{(p,q) \in E_{ij}} z_{pq} \quad \{i,j\} \in N^3 \tag{6}$$

$$z_{ij} \leq x_i \quad z_{ij} \leq x_j \quad (i,j) \in E \tag{7}$$

$$z_{ij} \geq x_i + x_j - 1 \quad (i,j) \in E \tag{8}$$

$$x_i \in \{0,1\} \quad i \in V \tag{3}$$

$$z_{ij} \in \{0,1\} \quad (i,j) \in E \tag{9}$$

Constraints (6) impose that two nonadjacent nodes i and j cannot be both in a 3-club unless a common neighbour is in the 3-club or a pair of neighbours, p and q, of i and j respectively, linked by an edge, are in the 3-club, since otherwise $dist_{G[S]}(i,j) > 3$. Constraints (7) and (8) impose that an edge (i,j) is used if and only if both its end nodes belong to the 3-club. Constraints (9) define edge variables as binary.

[F_S] is a natural formulation with $|V|$ variables and a non-polynomial number of constraints. [F_N] is an extended formulation with $|V|+|E|$ variables and $(|V|^2 - |V|/2) + 2|E|$ constraints.

Almeida and Carvalho [21] performed an analytical comparison of the linear programming relaxations of all integer formulations in the literature for $k=3$. They showed for $k=3$ that formulation [F_S] dominates all the others from the linear programming (LP) point of view. They also proposed enhanced versions of the chain and the recursive formulations in [14,19] and showed that they are LP-equivalent to [F_S].

All formulations for the k-club problem mentioned above include packing constraints of the form $x_i + x_j \leq 1$, for all $i,j \in V : dist_G(i,j) > k$, to guarantee that if the distance between two nodes is greater than k then at most one of them is selected to be in the k-club. Packing constraints enforce k-clique conditions, which are necessary but not sufficient conditions in the definition of k-clubs. They play a decisive role in the computational performance of Algorithm 4.1 (see Section 4.1) because they force the elimination of all nodes that are at a distance greater than k of at least one node of the initial k-club.

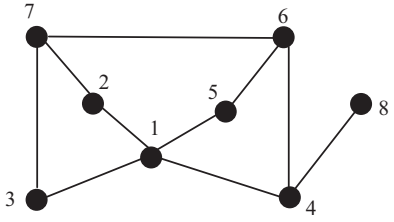

Fig. 1. Example of a graph with 2-club number equal to 7.

3. Motivation for new heuristics

Designing improvement heuristics for the maximum k-club problem is not a straightforward task. A k-club S is a maximal k-club if and only if for every $W \subseteq V \setminus S$ the induced subgraph $G[S \cup W]$ has diameter greater than k. If S is not maximal, then a subset $W \subseteq V \setminus S$ can be added to S to obtain a maximal k-club. While testing cliques for maximality can be performed in polynomial time, testing maximality of k-clubs in general graphs is an NP-hard problem (Mahdavi Pajouh and Balasumdaram [15]). This is a consequence of the non-hereditary nature of k-clubs. Every subset of a clique is a clique, but for k-clubs this property does not hold. If S' and S are k-clubs and $S \subset S'$, then $S \cup W$ is not necessarily a k-club for every subset of nodes $W \subset (S' \setminus S)$. An illustration is provided by the graph depicted in Fig. 1. $S=\{1, 2, 3, 4, 5\}$ is a 2-club and so is set $S'=\{1, 2, 3, 4, 5, 6, 7\}=S \cup \{6, 7\}$. However, neither $S_1=S \cup \{6\}$ nor $S_2=S \cup \{7\}$ are 2-clubs.

Furthermore, it may happen that for a k-club S and two subsets of $V \setminus S$, say W_1 and W_2, $S \cup W_1$ and $S \cup W_2$ are both maximal k-clubs, but with different cardinalities. An illustration is provided by the graph depicted in Fig. 2. Consider the 3-club $S=\{1, \ldots, 10\}$ and sets $W_1=\{a, b\}$ and $W_2=\{c, d, e\}$. $S_1=S \cup W_1$ and $S_2=S \cup W_2$ are both 3-clubs, are both maximal, but S_1 has 12 nodes whereas S_2 has 13 nodes.

The non-hereditary nature of k-clubs is also adverse for traditional greedy heuristics. An enlightening example is given by odd-holes (*i.e.*, cycles with an odd number of nodes and no edges between nonadjacent nodes of the cycle). Let $G=(V,E)$ be a graph and let $U \subseteq V$ be a subset of $2k+1$ nodes ($k > 1$) whose induced graph is an odd-hole. Set U is a k-club, but any constructive algorithm that proceeds by adding one node at a time, while keeping feasibility, will stop after adding $k+1$ nodes and missing another k nodes, *i. e.*, with a relative gap of $(k/2k+1)$.

In light of the discussion above, the heuristics presented in Sections 4 and 5 were devised based on a basic principle: avoid single-element operations when enlarging feasible solutions or removing nodes from non-feasible solutions. To avoid single-element operations, the enlargement of feasible solutions and the removal of nodes from non-feasible solutions are accomplished by solving restricted versions of compact formulations for the k-club problem.

4. New heuristics for the k-club problem

The algorithms proposed in this section are two-phase constructive and elimination heuristics. In the constructive heuristic, phase 1 is devoted to generating a k-club S and phase 2 is devoted to obtaining a largest set $W \subseteq (V \setminus S)$ such that $S \cup W$ is a k-club. In the elimination heuristic, phase 1 addresses the problem of finding

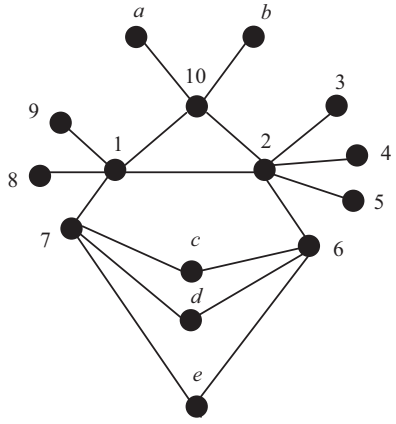

Fig. 2. Example of a graph with 3-club number equal to 13.

a set $C \subseteq V$ likely to contain large k-clubs and phase 2 is devoted to finding a largest k-club $S \subseteq C$.

4.1. Constructive heuristic

The aim of this algorithm is to add nodes to a given non-maximal k-club S in order to obtain a maximum cardinality k-club that includes S. For this purpose, it is necessary to find a largest set $W \subseteq V \setminus S$ such that $G[S \cup W]$ has diameter less than or equal to k. In other words, we need to identify a largest set of nodes that can be added to the current set S while complying with the diameter condition that defines the subgraph induced by a k-club. Let $A(S) = \{W \subseteq V \setminus S : \ diam(G[S \cup W]) \leq k\}$ be the family of all such sets. A set $W^* = \arg\max \{|W| : \ W \in A(S)\}$ is an optimal solution of the restricted integer problem that results from adding to a valid formulation of the k-club problem the condition that all nodes in S must be included in the solution. The general framework of the algorithm can be summarized as follows:

Algorithm 4.1.

begin
  /* Phase 1 */
    Find an initial k-club, S_I;
  /* Phase 2 */
    $A(S_I) := \{W \subseteq V \setminus S_I : \ diam(G[S_I \cup W]) \leq k\}$;
    $W^* := \arg\max \{|W| : \ W \in A(S_I)\}$;
    $S_F := W^* \cup S_I$;
end

The final solution S_F returned by Algorithm 4.1 is a k-club, which is maximal in $G = (V, E)$ and has maximum cardinality among all k-clubs that include S_I.

Phase 1 of Algorithm 4.1 may be implemented with any heuristic algorithm for the k-club problem (see Section 2). Phase 2 may be implemented by adding to an integer programming formulation of the k-club problem constraints $x_v = 1$, for all $v \in S_I$. For every node $v \in S_I$, all variables x_j associated with nodes $j \in V \setminus S_I$ such that $dist_G(v, j) > k$ can be eliminated from the formulation, further reducing its dimension.

4.2. Elimination heuristic

Bearing in mind the discussion in Section 3, it seems intuitively appealing to test another two-phase approach variant: first find a k-clique C in $G = (V, E)$ and then find a largest k-club in the family of all k-clubs included in C.

Let $D(C) = \{S \subseteq C : \ diam(G[S]) \leq k\}$ be the family of all k-clubs that can be obtained by deleting nodes of C. A largest k-club in this family $S^* = \arg\max \{|S| : \ S \in D(C)\}$ can be obtained by solving the restricted model that results from adding constraints $x_v = 0$, for all $v \in V \setminus C$, to any integer programming formulation of the k-club problem.

The algorithm can be summarized as follows:

Algorithm 4.2.

begin
  /* Phase 1 */
    Find a maximum k-clique, C;
    if $diam(G[C]) \leq k$ **then** $S_F := C$;
    else
  /* Phase 2 */
    $D(C) := \{S \subseteq C : diam(G[S]) \leq k\}$;
    $S_F := \arg\max \{|S| : S \in D(C)\}$;
    end if
end

Algorithm 4.2 can be interpreted as an enhanced version of heuristic k-Clique and Drop. For a k-clique C, built in phase 1, it solves to optimality the problem of finding a largest k-club contained in C, rather than applying heuristic Drop. As a consequence, if heuristic k-Clique and Drop and Algorithm 4.2 are initialized with the same maximum k-clique C, then S_F, the k-club returned by Algorithm 4.2, has at least as many nodes as the k-club generated by heuristic k-Clique and Drop. If there is more than one maximum k-clique in G, as it is often the case, and if the algorithms find different sets C in phase 1, then no *a priori* claim can be made about the cardinality of their final k-clubs. To reduce the computing time, a near-optimal solution can be used in phase 1 of Algorithm 4.2, as well as in step 1 of heuristic k-Clique and Drop.

5. New heuristics for the 3-club problem

In this section, we propose new heuristics based on results that hold only for the case $k = 3$.

5.1. Heuristic Backbone

In any graph $G = (V, E)$, the end nodes of an edge and their neighbours are a 3-club. For any edge $(u, v) \in E$, we will refer to this structure as the backbone induced by (u, v) and denote its set of nodes by $bkb(u, v)$.

Heuristic Backbone selects an edge $(u^*, v^*) \in E$ that induces a backbone with a maximum number of nodes and returns its set of nodes, S_B. Denoting by N_i the neighbourhood $\{j \in V : (i, j) \in E\}$ of node $i \in V$, the algorithm can be summarized as follows:

Algorithm 5.1.

begin
  $bkb(u, v) := N_u \cup N_v$ for all $(u, v) \in E$;
  $(u^*, v^*) := \arg\max \{|bkb(u, v)| : (u, v) \in E\}$;
  $S_B := bkb(u^*, v^*)$;
end

For any graph G, the number of nodes in the 3-club S_B, generated by heuristic Backbone, is greater than or equal to the cardinality of the 3-club S_C, built with heuristic Constellation. Let p be the maximum degree node selected in step 1 of heuristic Constellation as the centre of the star and let q be the node in the star whose neighbours are added in step 2. The number of nodes in S_C is the cardinality of $bkb(p, q)$ which is smaller than or equal to the cardinality of $bkb(u^*, v^*)$. That the number of nodes in S_C may be strictly smaller than the number of nodes in S_B is illustrated by the graph depicted in Fig. 3. Heuristic Constellation generates the 3-club $S_C = bkb(1, 6) = \{1, 2, 3, 4, 5, 6, 7\}$ whereas heuristic Backbone generates the 3-club $S_B = bkb(6, 7) = \{1, 4, 5, 6, 7, 8, 9, 10\}$. S_C has 7 nodes whereas S_B has 8 nodes.

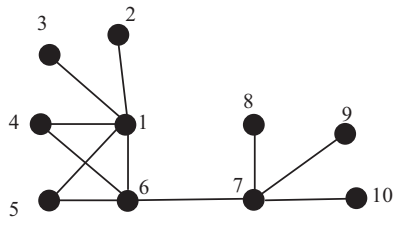

Fig. 3. Illustration of heuristic Backbone's dominance over heuristic Constellation.

5.2. Heuristics based on [F_S_{IR}]

[F_S_{IR}] is a compact integer relaxation of the natural formulation [F_S] for the 3-club problem proposed in Almeida and Carvalho [20] (see Section 2.2). Any feasible solution of [F_S_{IR}] is a 3-clique that satisfies some additional constraints, which are necessary for the 3-club structure. Therefore, an optimal solution of [F_S_{IR}] is likely to contain a large 3-club. Algorithm 5.2 was designed to test this idea.

Algorithm 5.2.

begin
  /* Phase 1 */
  $S_I \leftarrow$ optimal solution of [F_S_{IR}];
  if *diam* $(G[S_I]) \leq 3$ **then** STOP;
  else
  /* Phase 2 */
  run heuristic Drop on subgraph $G[S_I]$;
  end if
end

It should be pointed out that, if heuristic 3-Clique and Drop is initialized with a maximum 3-clique $\overline{C}$ and heuristic S_{IR} and Drop is initialized with set $\overline{S}_I$, optimum for [F_S_{IR}], although the inequality $|\overline{S}_I| \leq |\overline{C}|$ holds, the inclusion $\overline{S}_I \subseteq \overline{C}$ may not hold, since [F_S_{IR}] and the maximum 3-clique problem may have multiple optimal solutions. Furthermore, even if the inclusion $\overline{S}_I \subseteq \overline{C}$ holds, the rules that drive the elimination of nodes performed by the heuristic Drop may lead to very different solutions.

In line with the rationale used for Algorithm 4.2, we also designed a heuristic which solves to optimality the problem of finding the largest 3-club in the universe of all 3-clubs contained in an optimal solution of [F_S_{IR}]. It is aimed at investigating whether or not it is worthwhile replacing heuristic Drop by an exact procedure in phase 2 of Algorithm 5.2. This method can be summarized as follows:

Algorithm 5.3.

begin
  /* Phase 1 */
  $S_I \leftarrow$ optimal solution of [F_S_{IR}];
  if *diam* $(G[S_I]) \leq 3$ **then** STOP;
  else
  /* Phase 2 */
  $D(S_I) := \{S \subseteq S_I : diam(G[S]) \leq 3\}$;
  $S_F := \arg\max \{|S| : S \in D(S_I)\}$;
  end if
end

For some graphs, solving to optimality the integer linear problems embedded in Algorithms 5.2 and 5.3 may be too time-consuming. As in heuristics presented in Section 4, near-optimal solutions may be used to reduce running times.

6. Computational results

All algorithms were coded in C and run on a 2.93 GHz PC Pentium III processor with 3.46GB of RAM. The integer linear programming problems were solved with the integer routine of ILOG/CPLEX 11.1. In algorithms 4.1, 4.2, 5.2, and 5.3 a time limit of 3 min was imposed in each call of the integer routine. In the multi-start versions of algorithm 4.1 (denoted by mS_IP and mB_IP) there are as many integer problems to solve as the number of starting nodes or edges selected. In these versions, the time limit for each integer model solved in one run of the algorithm was set by dividing 10 min by the number of integer problems to solve. When the time limit was hit before obtaining a proven optimal solution, we used the best known integer solution. To assess the performance of the new heuristics, we used the results obtained with the exact algorithms proposed in [17,20] for $k=2$ and $k=3$, respectively. Each gap was computed as

$$\text{GAP} = 100 \times \frac{\overline{\omega}_{best} - |S_H|}{|S_H|}\%$$

where $\overline{\omega}_{best}$ is the k-club number (for the instances solved to proven optimality by the exact algorithms) or the best known upper bound on the k-club number (for the unsolved instances) and $|S_H|$ is the cardinality of the k-club generated by heuristic H.

6.1. Randomly generated uniform graphs

Uniform graphs with 50, 100, 150, and 200 nodes were randomly generated as described in [16]. The generation procedure is controlled by two parameters a and b $(0 \leq a \leq b \leq 1)$. The expected edge density is equal to $(a+b)/2$ and the node degree variance increases with $b-a$. All computational results presented in the literature, both for exact and heuristic algorithms, indicate that, in practice, the difficulty in solving the maximum k-club problem depends to a large extent on the combination of values of k and edge density of the graphs. For the 2-club problem, graphs with edge density around 0.15 seem to be the hardest to solve, regardless of the approach adopted. In this study, for $k=2$, we considered edge density values D $\in \{0.10, 0.15, 0.20\}$. For $k=3$, many authors consider graphs with edge density around 0.05 as the hardest to solve but the results in [20] indicate that graphs with edge density around 0.035 may be even harder to solve to optimality. In this study, for $k=3$, we considered edge density values D $\in \{0.020, 0.035, 0.050\}$. All graphs are connected. For edge density 0.020 (resp. 0.035) we do not present results for graphs with 50 and 100 nodes (resp. with 50 nodes) because the majority of graphs generated was not connected. For each combination of node number and edge density there are 10 graphs in the test set. As in [15,20], 5 graphs (group I) were generated with $a=0$ and $b=2\times$ D and 5 graphs (group II) with $a=b=$D. All these instances can be downloaded from https://aquila2.iseg.ulisboa.pt:443/aquila/homepage/f706/uniform-graphs.

The figures presented in all tables are average values for these groups of 5 graphs.

6.1.1. Results for $k=2$

Table 1 contains the results obtained with Algorithm 4.1. Phase 1 was implemented with heuristic Star (see Section 2). In phase 2 we used the simplified model (1)–(3) (see Section 2.2) with additional constraints $x_v = 1$, for all $v \in S_I$.

Column (1) of Table 1 presents average ratios of the cardinality of the 2-clubs generated by Algorithm 4.1 to the cardinality of the 2-clubs generated by heuristic Star. They show that only for graphs with edge density 0.20 did the new algorithm manage to achieve significant improvements. This poor performance is partially explained by the fact that, for low density graphs, heuristic Star solutions are frequently optimal, as pointed out in [16]. The gap values presented in Table 3 show that in this test set it turned out that heuristic Star generated optimal solutions for a large number of graphs with edge density 0.10. Columns (3) and (4) show that computing times are very small in all cases. We also implemented a multi-start version of the algorithm, denoted by mS_IP, which consists of repeating the algorithm with the initialization centred at different nodes. The results in Columns (2) and (5) show that, for the 0.15 and 0.20 edge density groups, the multi-start version was much more successful, with running times below 40 s. Considering the whole set of 120 graphs, the basic version of

Table 1
Algorithm 4.1.

			Ratio		CPU time (s)		
D	$\|V\|$	Group	S_IP/S (1)	mS_IP/S (2)	S (3)	S_IP (4)	mS_IP (5)
0.10	50	I	1.00	1.00	0.0	0.6	1.2
	50	II	1.00	1.00	0.0	0.8	0.6
	100	I	1.00	1.00	0.0	0.8	1.6
	100	II	1.00	1.00	0.2	0.6	1.8
	150	I	1.00	1.00	0.0	0.8	3.4
	150	II	1.00	1.00	0.0	0.6	3.8
	200	I	1.00	1.00	0.0	0.6	7.4
	200	II	1.00	1.00	0.0	0.8	7.0
0.15	50	I	1.00	1.07	0.0	0.8	1.0
	50	II	1.00	1.03	0.2	1.2	1.2
	100	I	1.01	1.42	0.0	0.6	3.0
	100	II	1.00	1.02	0.2	1.2	1.6
	150	I	1.00	1.72	0.0	0.8	13.0
	150	II	1.00	1.07	0.0	0.6	5.2
	200	I	1.17	2.31	0.0	1.0	37.6
	200	II	1.00	1.08	0.0	0.6	15.6
0.20	50	I	1.03	1.36	0.0	0.8	1.6
	50	II	1.01	1.28	0.0	0.6	1.0
	100	I	1.21	1.80	0.0	0.8	3.8
	100	II	1.37	2.09	0.0	0.4	4.0
	150	I	2.08	2.26	0.0	0.8	11.6
	150	II	3.00	3.18	0.0	1.0	13.2
	200	I	2.56	2.60	0.0	0.6	29.2
	200	II	3.47	3.48	0.0	1.0	33.6

Table 2
Algorithm 4.2.

			Ratio	CPU time (s)	
D	$\|V\|$	Group	2-Cl_IP/2-Cl & D (1)	2-Cl&D (2)	2-Cl_IP (3)
0.10	50	I	1.00	0.6	1.0
	50	II	1.00	1.2	0.6
	100	I	1.10	1.2	1.4
	100	II	1.02	2.0	2.0
	150	I	1.20	2.6	2.6
	150	II	1.19	11.2	11.0
	200	I	1.27	6.4	7.8
	200	II	1.26	97.8	96.4
0.15	50	I	1.01	1.2	1.0
	50	II	1.00	0.6	1.0
	100	I	1.14	0.8	0.6
	100	II	1.17	0.4	1.2
	150	I	1.05	0.6	0.8
	150	II	1.55	1.0	15.2
	200	I	1.04	0.6	0.8
	200	II	1.56	0.8	178.4
0.20	50	I	1.00	0.8	0.8
	50	II	1.04	1.0	1.0
	100	I	1.00	0.8	0.6
	100	II	1.09	1.0	1.0
	150	I	1.01	1.4	1.4
	150	II	1.01	0.8	1.0
	200	I	1.00	1.2	1.4
	200	II	1.00	1.2	1.4

Algorithm 4.1 managed to increase the number of nodes in the Star solution in 28% of the graphs and the multi-start version outperformed the basic version in 48% of the graphs.

The results obtained with Algorithm 4.2 are presented in Table 2. The computational implementation of phase 2 was made using the same integer formulation used to implement Algorithm 4.1.

The ratios presented in Column (1) of Table 2 show that Algorithm 4.2 outperformed heuristic 2-Clique and Drop in seventeen groups of graphs. Considering the whole set of 120 graphs, the cardinality of the 2-clubs generated by Algorithm 4.2 was, on average, 11% higher than the cardinality of the 2-clubs generated by heuristic 2-Clique and Drop. The best relative performance of Algorithm 4.2 was achieved for graphs with edge density 0.15, which are known to be particularly hard to solve to optimality. The overall average computing time of Algorithm 4.2 was less than 14 s. The time limit was hit in heuristic 2-Clique and Drop and Algorithm 4.2, while looking for a maximum 2-clique, for one graph, with 200 nodes and edge density 0.10, in group II. It was also hit in phase 2 of Algorithm 4.2 for four graphs, with 200 nodes and edge density 0.15, also in group II.

The average gaps for Algorithms 4.1 and 4.2 are presented in Table 3, with the best row result printed in bold type.

Columns (1) and (2) present the average gaps obtained with heuristic 2-Clique and Drop and Algorithm 4.2, respectively. Columns (3)–(5) display the average gaps of heuristic Star, the basic version of Algorithm 4.1, and its multi-start version, respectively. For graphs with edge density $D=0.10$, the average gap obtained with heuristic 2-Clique and Drop was 73.04%, while by solving a restricted integer problem, instead of running Drop, the average gap was 50.79%. Heuristic Star and both versions of Algorithm 4.1 yielded an average gap of only 0.34%, with many proven optimal solutions generated by Star. For graphs with edge density $D=0.15$, the average gaps obtained with heuristics 2-Clique and Drop and Star were 47.55% and 59.63%, respectively. With Algorithm 4.2 (2-Cl_IP) and the multi-start version of Algorithm 4.1 (mS_IP), they were only 22.73% and 24.07%, respectively. For graphs with edge density $D=0.20$, the solutions produced by heuristic 2-Clique and Drop had an average gap of 7.37%, whereas by solving a restricted integer model instead of running heuristic Drop, the average gap was only 5.14%. Heuristic Star showed a poor performance, with an average gap of 127.63%. With the basic version of Algorithm 4.1 this gap was reduced to 26.31% and with the multi-start version it was further reduced to only 1.52%.

Table 3
Algorithms for the 2-club problem: Gaps (%).

			2-Cl & D	2-Cl_IP	S	S_IP	mS_IP
D	$\|V\|$	Group	(1)	(2)	(3)	(4)	(5)
0.10	50	I	**1.54**	**1.54**	**1.54**	**1.54**	**1.54**
	50	II	1.67	1.67	**0.00**	**0.00**	**0.00**
	100	I	50.24	36.10	**0.00**	**0.00**	**0.00**
	100	II	83.36	79.36	**0.00**	**0.00**	**0.00**
	150	I	90.57	59.36	**0.63**	**0.63**	**0.63**
	150	II	114.73	78.38	**0.00**	**0.00**	**0.00**
	200	I	83.87	43.30	**0.51**	**0.51**	**0.51**
	200	II	158.32	106.61	**0.00**	**0.00**	**0.00**
0.15	50	I	22.83	21.51	10.67	10.67	**3.72**
	50	II	19.23	19.23	11.27	11.27	**8.19**
	100	I	27.54	11.82	44.74	43.75	**2.01**
	100	II	71.95	46.60	29.15	29.15	**25.98**
	150	I	15.27	10.20	75.03	74.18	**1.70**
	150	II	95.11	**26.03**	45.89	45.89	37.87
	200	I	10.21	6.02	132.50	108.65	**0.67**
	200	II	118.29	**40.41**	127.77	127.77	112.39
0.20	50	I	5.63	5.63	36.55	32.78	**0.83**
	50	II	22.16	16.75	36.11	34.42	**7.13**
	100	I	5.42	5.42	79.85	52.78	**0.00**
	100	II	19.45	9.27	117.00	73.34	**4.19**
	150	I	2.12	1.37	125.62	8.72	**0.00**
	150	II	3.01	1.85	218.43	6.50	**0.00**
	200	I	0.33	0.22	159.89	1.69	**0.00**
	200	II	0.84	0.62	247.56	0.21	**0.00**

To get a more general picture of the algorithms analysed in Tables 1–3, we computed overall average gaps, overall average running times, and number of times each heuristic obtained the best 2-club. They are given in Figs. 4–6, respectively. In Fig. 5, Star's average CPU time is omitted because it is too small to be visible in the chart.

The most successful heuristic was mS_IP, the multi-start version of Algorithm 4.1: it found the best 2-club in 106 graphs and yielded an average gap of 8.7%, with an average running time of only 8.5 s. Its good performance, both in terms of solution quality and running time, can be explained by the fact that, for each choice of the initial node, a large proportion of nodes can be immediately eliminated. This elimination leads to small integer models that are, in general, fast to solve. Since the restricted models are fast to solve, it is possible to try a large number of different initial nodes and get a good insight on the set of large 2-clubs within limited running times. Bringing constructive approaches face to face with elimination approaches, the former were more successful in finding large 2-clubs than the latter and their running times were also more favourable. There are possibly two reasons for that: the dimensions of the integer models embedded in the heuristics and the incumbent solution for the branch-and-bound search, provided by phase 1 of all constructive approaches, but not available in elimination approaches.

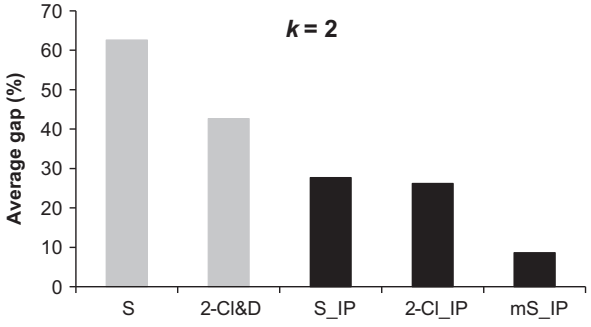

Fig. 4. Average gaps.

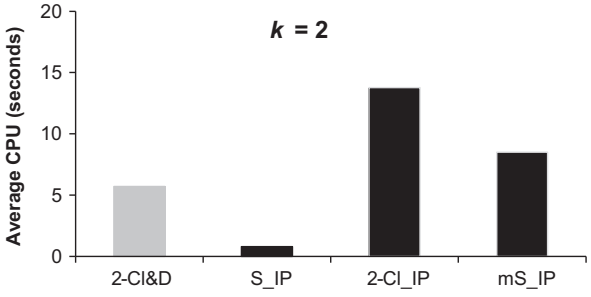

Fig. 5. Average CPU time.

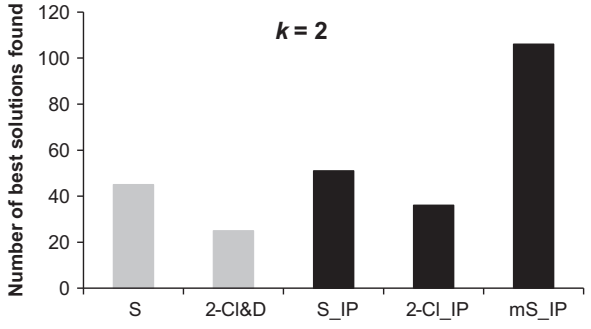

Fig. 6. Number of best solutions found.

6.1.2. Results for $k=3$

In Table 4 we compare the performances of heuristics Backbone and Constellation for the 3-club problem. The figures in Column (1) are average ratios of the cardinality of the 3-clubs generated by Backbone to the cardinality of the 3-clubs generated by Constellation. These ratios show that in 50% of the groups, the 3-clubs built by heuristic Backbone are strictly larger, on average, than the 3-clubs built by heuristic Constellation, but the differences are very small. The corresponding computing times, in Columns (2) and (3), are undistinguishable.

Phase 1 of Algorithm 4.1 was implemented with heuristics Star, Constellation, and Backbone (represented in Table 5 by the initials S, C, and B, respectively) and phase 2 was implemented with the compact formulation [F_N] (see Section 2.2) with the additional constraints $x_v = 1$, for all $v \in S_I$.

Columns (1)–(3) of Table 5 show the average ratios of the number of nodes in the final 3-clubs to the number of nodes in the 3-clubs used to initialize the procedures. Columns (5)–(10) present the corresponding running times. It should be pointed out that for a given graph, since a star is a k-club for any $k > 1$, the number of nodes in S_I is smaller when heuristic Star is used to implement phase 1. As a consequence, the number of constraints $x_v = 1$, $v \in S_I$, added to formulation [F_N] is smaller and the integer model in phase 2 takes longer to solve, but may produce larger 3-clubs (see Table 7). While, on average, phase 2 of Algorithm 4.1 improved Constellation solutions by 41%, it improved Backbone solutions by 44%, suggesting that the backbone structure may be a better seed for large 3-clubs. Columns (4) and (11) refer to a multi-start version mB_IP of the algorithm, implemented running heuristic Backbone in phase 1. This version consists of repeating the algorithm with initial backbones induced by different edges. The selection of these edges was guided by the intuition that edges with high end nodes' degree are more likely to produce good results. After a limited parameter tuning, we decided to select an edge (i, j) to induce an initial backbone if $\min\{\Delta_i, \Delta_j\} \geq 0.5\,\Delta(G)$ and $\max\{\Delta_i, \Delta_j\} \geq 0.7\,\Delta(G)$, where Δ_v and $\Delta(G)$ denote the degree of node $v \in V$ and the maximum node degree in G, respectively. This choice yielded a number of initial backbones that did not lead to unacceptably long computing times and enhanced the search. The ratios in Columns (3) and (4) show that mB_IP outperformed B_IP in all but one group of graphs: graphs with 150 nodes and edge density 0.020 in group I. As shown in

Table 4
Heuristics Backbone and Constellation.

			Ratio	CPU time (s)	
D	|V|	Group	B/C (1)	B (2)	C (3)
0.020	150	I	1.00	0.0	0.0
	150	II	1.00	0.0	0.0
	200	I	1.00	0.0	0.0
	200	II	1.01	0.0	0.0
0.035	100	I	1.01	0.0	0.0
	100	II	1.04	0.0	0.0
	150	I	1.00	0.0	0.0
	150	II	1.02	0.0	0.0
	200	I	1.01	0.0	0.0
	200	II	1.01	0.0	0.0
0.050	50	I	1.00	0.0	0.0
	50	II	1.02	0.0	0.0
	100	I	1.00	0.0	0.0
	100	II	1.00	0.0	0.0
	150	I	1.01	0.0	0.0
	150	II	1.00	0.2	0.2
	200	I	1.00	0.0	0.0
	200	II	1.02	0.0	0.0

Table 5
Algorithm 4.1.

D	|V|	Group	Ratio				CPU time (s)						
			S_IP/S (1)	C_IP/C (2)	B_IP/B (3)	mB_IP/B (4)	S (5)	S_IP (6)	C (7)	C_IP (8)	B (9)	B_IP (10)	mB_IP (11)
0.020	150	I	1.80	1.15	1.19	1.19	0.0	1.8	0.0	1.8	0.0	1.4	2.8
	150	II	1.79	1.11	1.13	1.14	0.0	1.4	0.0	1.0	0.0	1.4	3.0
	200	I	1.81	1.11	1.16	1.21	0.0	1.8	0.0	1.2	0.0	1.4	6.2
	200	II	1.77	1.11	1.11	1.15	0.0	1.6	0.0	1.2	0.0	1.2	6.4
0.035	100	I	2.17	1.30	1.25	1.35	0.0	1.2	0.0	1.2	0.0	1.0	2.6
	100	II	1.87	1.19	1.16	1.20	0.0	1.2	0.0	1.4	0.0	1.4	1.8
	150	I	2.41	1.42	1.40	1.45	0.0	1.6	0.0	1.4	0.0	1.4	5.4
	150	II	2.03	1.20	1.20	1.26	0.0	1.6	0.0	1.0	0.0	1.2	6.6
	200	I	2.56	1.48	1.52	1.54	0.0	5.2	0.0	1.6	0.0	1.8	12.8
	200	II	2.23	1.34	1.30	1.42	0.0	3.0	0.0	2.0	0.0	1.4	17.4
0.050	50	I	1.60	1.05	1.08	1.12	0.0	1.2	0.0	1.6	0.0	1.0	1.4
	50	II	1.64	1.07	1.11	1.13	0.0	1.2	0.0	0.8	0.0	1.0	1.4
	100	I	2.20	1.30	1.33	1.42	0.0	1.2	0.0	1.2	0.0	1.0	3.4
	100	II	2.10	1.31	1.32	1.34	0.0	1.0	0.0	1.2	0.0	1.0	3.4
	150	I	3.22	1.62	1.76	2.24	0.0	6.4	0.0	2.2	0.0	2.0	34.4
	150	II	2.72	1.42	1.42	1.61	0.2	7.2	0.2	2.2	0.2	2.2	15.8
	200	I	5.47	3.04	3.09	3.48	0.0	8.8	0.0	3.4	0.0	3.0	229.2
	200	II	4.34	2.17	2.35	3.05	0.0	182.4	0.0	5.4	0.0	9.2	305.8

Table 6
Algorithm 4.2, Algorithm 5.2, and Algorithm 5.3.

D	|V|	Group	Ratio			CPU time (s)	
			3-Cl_IP/3-Cl & D (1)	S_{IR}_IP/S_{IR} & D (2)	S_{IR}_IP/3-Cl_IP (3)	3-Cl_IP (4)	S_{IR}_IP (5)
0.020	150	I	1.00	1.00	1.00	2.8	4.0
	150	II	1.00	1.00	1.04	4.6	7.2
	200	I	1.01	1.00	1.22	20.6	48.2
	200	II	1.00	1.00	1.01	19.4	48.0
0.035	100	I	1.00	1.00	1.13	1.2	2.2
	100	II	1.00	1.00	1.10	1.0	1.6
	150	I	1.00	1.00	1.08	1.8	26.6
	150	II	1.04	1.00	1.41	3.6	58.8
	200	I	1.15	1.22	1.01	4.6	181.2
	200	II	1.07	1.05	1.14	5.4	181.6
0.050	50	I	1.02	1.00	1.03	0.6	1.0
	50	II	1.06	1.00	1.09	1.2	1.2
	100	I	1.00	1.00	1.17	0.4	4.0
	100	II	1.04	1.13	1.34	1.0	8.6
	150	I	1.17	1.14	1.02	0.4	23.8
	150	II	1.38	1.26	1.03	1.0	116.4
	200	I	1.04	1.04	1.01	46.8	3.0
	200	II	1.07	1.21	1.16	181.2	133.8

Table 7, it turned out that heuristic B_IP built optimal 3-clubs for all graphs in that group, leaving no room for improvement.

Table 6 displays the results obtained with Algorithm 4.2, and also with algorithms 5.2 and 5.3. Figures in Columns (1) and (2) show that solving an integer model in phase 2, rather than using heuristic Drop, yielded an average increase of 6% on the final solution cardinality, regardless of the option taken in phase 1 (a 3-clique or a solution of [F_S_{IR}]). The best results were achieved for graphs with edge density 0.05 – in the group II of instances with 150 nodes 3-Cl_IP and S_{IR}_IP yielded average improvements of 38% or 26%, respectively. Figures in Column (3) show that, in all groups but one, building an optimal solution of [F_S_{IR}] in phase 1 yielded better results than building a maximum 3-clique. As shown in Columns (4) and (5) it was also more time consuming but the overall average running time was below one minute.

The average gaps are presented in Table 7, with the best row value printed in bold type. Considering the whole set of 90 graphs in the test set, the new algorithms proposed in this article produced average gaps between 1.83% (mB_IP) and 19.01% (3-Cl_IP) whereas heuristics Constellation and 3-Clique and Drop produced average gaps of 62.06% and 27.01%, respectively. The multi-start version of Algorithm 4.1 built proven optimal 3-clubs for all graphs in 11 out of the 18 groups. In the whole set of 90 graphs, it produced the largest 3-club for 85 of them. Algorithm 4.1, implemented with the Star solution (phase 1), and algorithm 5.3 were close competitors for the second position in the global ranking.

To get a more general picture of the algorithms for the case $k=3$, we computed overall average gaps, overall average running times, and number of times each heuristic obtained the best 3-club. They are given in Figs. 7–9, respectively.

The most successful heuristic was mB_IP: it found the best 3-club in 85 graphs and yielded an average gap of 1.8%, with an average running time of only 36.7 s. In terms of solution quality

Table 7
Algorithms for the 3-club problem: Gaps (%).

D	|V|	Group	C (1)	3-Cl & D (2)	S_IP (3)	C_IP (4)	B_IP (5)	3-Cl_IP (6)	S_{IR} & D (7)	S_{IR}_IP (8)	mB_IP (9)
0.020	150	I	18.78	**0.00**	**0.00**	3.07	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	150	II	14.39	4.96	**0.00**	3.41	1.05	4.96	1.43	1.43	**0.00**
	200	I	21.06	23.94	6.57	9.34	3.40	21.86	**0.00**	**0.00**	**0.00**
	200	II	16.53	2.06	5.19	5.19	3.81	2.06	1.05	1.05	**0.00**
0.035	100	I	37.06	19.88	1.11	6.16	8.77	9.88	7.14	7.14	**0.00**
	100	II	24.44	10.68	4.71	4.71	3.28	10.68	1.00	1.00	**0.00**
	150	I	45.76	15.18	**0.49**	3.07	4.36	15.18	6.71	6.71	**0.49**
	150	II	29.20	49.00	**0.77**	7.50	5.64	42.96	1.60	1.60	**0.77**
	200	I	59.20	50.40	2.90	8.03	3.30	30.97	58.22	30.51	**2.07**
	200	II	42.78	59.65	6.45	7.08	9.51	49.18	37.06	30.87	**0.00**
0.050	50	I	12.38	5.45	3.08	6.41	3.08	3.33	**0.00**	**0.00**	**0.00**
	50	II	15.09	17.14	5.45	7.27	1.67	8.89	**0.00**	**0.00**	**0.00**
	100	I	42.24	23.79	6.14	9.11	6.96	23.79	6.07	6.07	**0.00**
	100	II	33.59	46.48	2.53	2.53	1.53	41.03	19.33	4.93	**0.00**
	150	I	137.35	34.82	20.24	48.04	34.75	15.10	29.32	13.46	**5.78**
	150	II	86.75	72.44	**15.95**	31.92	31.92	25.02	55.37	22.85	16.45
	200	I	249.72	8.73	9.26	15.03	13.28	4.72	7.67	3.69	**0.57**
	200	II	230.71	41.53	39.15	61.49	49.59	32.49	38.68	14.52	**6.76**

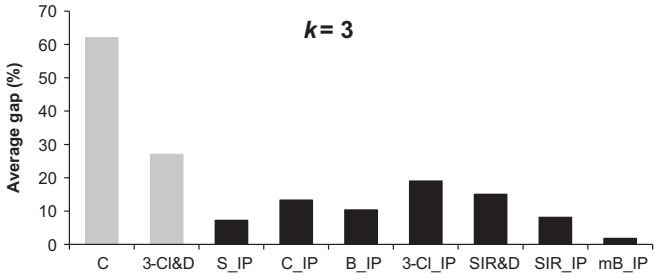

Fig. 7. Average gaps.

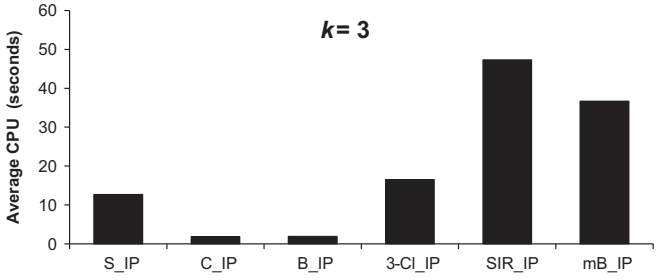

Fig. 8. Average CPU time.

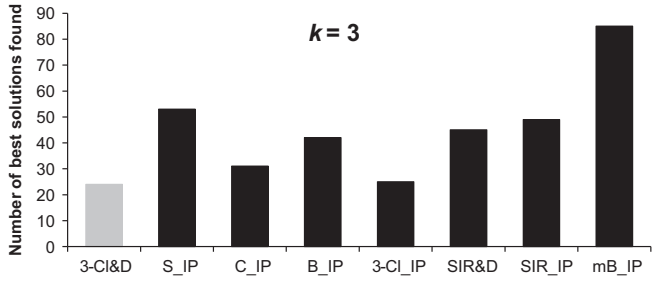

Fig. 9. Number of best solutions found.

the second best is closely disputed by a constructive method (Algorithm 4.1, initialized with the Star solution) and an elimination method initialized with an optimal solution of [F_S_{IR}] (Algorithm 5.3), but in terms of running time the constructive approach is a clear winner. This difference stems from two reasons: in phase 1, solving [F_S_{IR}] takes longer than running heuristics Star, Constellation or Backbone; in phase 2, forcing the nodes in the initial 3-club into the solution excludes a large number of other nodes and provides the integer routine with a feasible solution at the root of the branch-and-bound tree, whereas removing only the nodes not in the solution of [F_S_{IR}] excludes a smaller number of nodes and does not provide an incumbent solution for the branch-and-bound tree search. Since heuristics Constellation and Backbone gave very similar results (see Table 4), in Fig. 7 we omitted the results for the second. Figs. 8 and 9 do not present results for heuristics Star, Constellation, and Backbone because their running times were very close to zero (see Table 5) and they found no best solution. Bringing constructive approaches face to face with elimination approaches, the results were again more favourable for the former, but the difference is less clear than for the case $k=2$.

6.2. Graphs from the DIMACS implementation challenge

To assess their performance on large-scale instances and to allow for a direct comparison with the VNS algorithm (Shahinpour and Butenko [18]), we also tested heuristics mS_IP and mB_IP on a subset of instances from the 10th DIMACS implementation challenge considered in [18]. These are all real-world sparse instances (all edge densities are below 0.15 and more than half of them are below 0.01). Columns (1), (2), (5) and (6) of Table 8 reproduce the figures presented in [18]. The computing times in Columns (2) and (6) were obtained on a Dell workstation with Intel 3.00 Gz Quad-core processor and 8.00 GB RAM (Shahinpour and Butenko [18]).

Columns (1)–(4) of Table 8 present the results for $k=2$: the cardinalities of the solutions found by heuristics VNS and mS_IP are shown in Columns (1) and (3), and the corresponding running times in Columns (2) and (4), respectively. The figures in Columns (1) and (3) show that mS_IP was outperformed by the VNS heuristic only in instances Football and Uk, by just one node in each case. Columns (2) and (4) show that the computing times for mS_IP were all under 10 s while the corresponding running times for the VNS heuristic were in the range [1, 11372]. These results confirmed mS_IP as a very effective heuristic algorithm for the 2-club problem on sparse graphs. For sparse graphs, the number of nodes that are at a distance greater than two of at least one of the nodes in a Star solution represents a very large share of the total number of nodes of the graph. At each step of mS_IP, by forcing the

Table 8
Results on DIMACS instances.

			$k=2$				$k=3$			
Instance	|V|	|E|	VNS (1)	CPU (2)	mS_IP (3)	CPU (4)	VNS (5)	CPU (6)	mB_IP (7)	CPU (8)
Karate	34	78	18	1	18	1	25	4	25	1
Dolphins	62	159	13	2	13	1	29	6	29	1
Polbooks	105	441	28	12	28	1	53	59	53	2
Adjnoun	112	425	50	70	50	<1	82	505	82	<1
Football	115	613	16	6	15	1	58	193	58	79
Jazz	198	2742	103	2013	103	9	174	4224	174	142
Celegans-metabolic	453	2025	238	3604	238	<1	371	3658	371	39
Email	1133	5451	72	274	72	<1	215	3605	211	816
Polblogs	1490	16715	352	3872	352	6	352	3608	532[a]	–
Netscience	1589	2742	35	70	35	<1	54	110	54	2
Add20	2395	7462	124	1957	124	8	671	3659	671	3612
Data	2851	15093	18	261	18	1	32	1919	31	9
Uk	4824	6837	5	392	4	<1	8	5747	7	1
Power	4941	6594	20	480	20	<1	30	1168	30	2
Add32	4960	9462	32	567	32	1	96	1450	99	49
Hep-th	8361	15751	51	1825	51	<1	120	3799	120	529
Whitaker3	9800	28989	9	2321	9	<1	15	3563	15	3
Crack	10240	30380	10	2452	10	<1	17	3498	15	4
PGPgiantcompo	10680	24316	206	5184	206	1	273	9301	400	17065
Cs4	22499	43858	5	11372	5	<1	9	7523	9	3

[a] Backbone heuristic's value.

nodes of a Star into the heuristic solution, that large share of nodes is immediately eliminated. As a consequence, the integer models embedded in mS_IP are very fast to solve to optimality. For these graphs, the search of multiple neighbourhoods, with the search hampered by the non-hereditary nature of the 2-club structure, is naturally more time consuming in most cases.

Columns (5)–(8) of Table 8 present the results for $k=3$: the cardinalities of the solutions found by the heuristics VNS and mB_IP are shown in Columns (5) and (7), and the corresponding running times in Columns (6) and (8), respectively. For the instances with more than 200 nodes, the number of initial backbones considered in mB_IP was limited to a maximum of five. For the Polblogs instance, the figure in Column (7) is the cardinality of the heuristic Backbone's solution since the integer routine of CPLEX reported insufficient memory and gave no result at all. On the other instances, the running time of heuristic mB_IP exceeded ten minutes in only four cases: Email, Add20, Hep-th, and PGPgiantcompo. For the PGPgiantcompo instance, the time limit given to the integer routine of CPLEX was hit, but in the other three cases the integer routine needed only a few seconds to compute the final solution, meaning that a huge share of the time was spent building the integer model. The figures in Columns (7) and (8) corroborate the comments in Section 6.1.2 for sparse instances. The backbone structure was confirmed as a good seed for large cardinality 3-clubs: it plays for the case $k=3$ a role similar to the role played by the star for the case $k=2$. Similarly to the case $k=2$, the search of multiple neighbourhoods, with the search hampered by the non-hereditary nature of the 3-club structure, is naturally more time consuming in most cases.

7. Final remarks

The use of graph concepts to analyse complex systems has been steadily growing in areas such as social network analysis, biology and financial data mining. In all these areas it is important to have fast heuristics to identify the elements relevant for the comprehension of the systems under analysis. In this paper, we have proposed new algorithms for the problem of finding large graph clusters with a small number of hops between any two elements – the maximum k-club problem. The work was motivated by the non-hereditary nature of the k-club structure. Our contribution is a novel approach to tackle the problem. This approach consists of using constructive and elimination algorithms to build restricted integer models that are much smaller in size than the integer models needed to solve the problem to proven optimality. The computational results obtained on a set of real-world DIMACS graphs indicate that the VNS algorithm (Veremyev and Boginski [18]) and constructive-based versions of the new algorithms give in general similar quality results, with the method that produces the largest k-club being dependent on the instance considered. In what computing time is concerned, the VNS algorithm is in general more time-consuming. For sparse graphs, the simplicity of the constructive and elimination procedures embedded in the new algorithms, together with their capability of reducing the dimensions of the integer models to be solved, renders this approach faster to run, when compared with an approach that relies on the search of multiple neighbourhoods, with the search hampered by the non-hereditary nature of the k-club structure. To the best of our knowledge, this is the first time that an integer model-based heuristic approach is proposed for the k-club problem.

Acknowledgements

This work is supported by the National Funding from FCT – Fundação para a Ciência e a Tecnologia, under the project: PEst-OE/MAT/UI0152.

We thank the three anonymous reviewers for their comments and suggestions and Ann Henshall, who was always ready to elucidate our English grammar and vocabulary doubts.

References

[1] Błażewicz J, Formanowicz P, Kasprzak M. Selected combinatorial problems of computational biology. Eur J Oper Res 2005;161:585–97.

[2] Balasundaram B, Butenko S, Trukhanov S. Novel approaches for analyzing biological networks. J Comb Optim 2005;10:23–39.
[3] Bruinsma G, Bernasco W. Criminal groups and transnational illegal markets: a more detailed examination on the basis of social network theory. Crime Law Soc Chang 2004;41:79–94.
[4] Varanda MP, Carvalho FD. Leadership and diffusion of information for policy implementation: a new methodology approach, Working Paper SOCIUS 1/2009. Universidade Técnica de Lisboa; 2009.
[5] Boginski V, Butenko S, Pardalos PM. Mining market data: a network approach. Comput Oper Res 2006;33:3171–84.
[6] Bomze IM, Budinich M, Pardalos PM, Pelillo M. The maximum clique problem. In: Du D-Z, Pardalos PM, editors. Handbook of combinatorial optimization, The Netherlands. Dordrecht: Kluwer Academic Publishers; 1999. p. 1–74.
[7] Butenko S, Wilhelm WE. Clique detection models in computational biochemistry and genomics. Eur J Oper Res 2006;173:1–7.
[8] Martins P. Extended and discretized formulations for the maximum clique problem. Comput Oper Res 2010;37:1348–58.
[9] Benlic U, Hao J-K. Breakout local search for maximum clique problems. Comput Oper Res 2012;40(1):192–206.
[10] McClosky B, Hicks IV. Combinatorial algorithms for the maximum *k*-plex problem. J Comb Optim 2012;23:29–49.
[11] Balasundaram B, Butenko S, Hicks IV, Sachdeva S. Clique relaxations in social network analysis: the maximum *k*-plex problem. Oper Res 2011;59:133–42.
[12] Mokken RJ. Cliques, clubs and clans. Qual Quan 1979;13:161–73.
[13] Wasserman S, Faust K. Social network analysis. New York: Cambridge University Press; 1994.
[14] Bourjolly J-M, Laporte G, Pesant G. An exact algorithm for the maximum *k*-club problem in an undirected graph. Eur J Oper Res 2002;138:21–8.
[15] Mahdavi Pajouh F, Balasundaram B. On inclusionwise maximal and maximum cardinality *k*-clubs in graphs. Discret Optim 2012;9:84–97.
[16] Bourjolly J-M, Laporte G, Pesant G. Heuristics for finding *k*-clubs in an undirected graph. Comput Oper Res 2000;27:559–69.
[17] Carvalho FD, Almeida MT. Upper bounds and heuristics for the 2-club problem. Eur J Oper Res 2011;210:489–94.
[18] Shahinpour S, Butenko S. Algorithms for the maximum *k*-club problem in graphs. J Comb Optim 2013;26(3):520–54.
[19] Veremyev A, Boginsky V. Identifying large robust network clusters via new compact formulations of maximum *k*-club problems. Eur J Oper Res 2012;218:316–26.
[20] Almeida MT, Carvalho FD. Integer models and upper bounds for the 3-club problem. Networks 2012;60(3):155–66.
[21] Almeida MT, Carvalho FD. An analytical comparison of the LP relaxations of integer models for the *k*-club problem. Eur J Oper Res 2014;232:489–98.