OPTIMIZATION-BASED NETWORK ANALYSIS WITH APPLICATIONS IN CLUSTERING AND DATA MINING

A Dissertation

by

SHAHRAM SHAHINPOUR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Sergiy Butenko
Committee Members,	Lewis Ntaimo
	Luca Quadrifoglio
	Kiavash Kianfar
Head of Department,	César Malavé

August 2013

Major Subject: Industrial Engineering

ABSTRACT

In this research we develop theoretical foundations and efficient solution methods for two classes of cluster-detection problems from optimization point of view. In particular, the s-club model and the biclique model are considered due to various application areas. An analytical review of the optimization problems is followed by theoretical results and algorithmic solution methods developed in this research.

The maximum s-club problem has applications in graph-based data mining and robust network design where high reachability is often considered a critical property. Massive size of real-life instances makes it necessary to devise a scalable solution method for practical purposes. Moreover, lack of heredity property in s-clubs imposes challenges in the design of optimization algorithms. Motivated by these properties, a sufficient condition for checking maximality, by inclusion, of a given s-club is proposed. The sufficient condition can be employed in the design of optimization algorithms to reduce the computational effort. A variable neighborhood search algorithm is proposed for the maximum s-club problem to facilitate the solution of large instances with reasonable computational effort. In addition, a hybrid exact algorithm has been developed for the problem.

Inspired by wide usability of bipartite graphs in modeling and data mining, we consider three classes of the maximum biclique problem. Specifically, the maximum edge biclique, the maximum vertex biclique and the maximum balanced biclique problems are considered. Asymptotic lower and upper bounds on the size of these structures in uniform random graphs are developed. These bounds are insightful in understanding the evolution and growth rate of bicliques in large-scale graphs. To overcome the computational difficulty of solving large instances, a scale-reduction

technique for the maximum vertex and maximum edge biclique problems, in general graphs, is proposed. The procedure shrinks the underlying network, by confirming and removing edges that cannot be in the optimal solution, thus enabling the exact solution methods to solve large-scale sparse instances to optimality. Also, a combinatorial branch-and-bound algorithm is developed that best suits to solve dense instances where scale-reduction method might be less effective. Proposed algorithms are flexible and, with small modifications, can solve the weighted versions of the problems.

Dedicated to my wife

and

my parents

ACKNOWLEDGEMENTS

I am grateful to many people for their help. First and foremost, I would like to express my gratitude to my advisor, Dr. Sergiy Butenko, for his continuous support, encouragement and especially insights and valuable comments. I sincerely thank Dr. Jorge Leon for providing me the opportunity to teach laboratories and lectures at Engineering Technology & Industrial Distribution Department. Teaching was a wonderful experience for me and a big step towards professional development. I would like to thank Dr. Catherine Yan, Dr. Lewis Ntaimo, Dr. Kiavash Kianfar, and Dr. Luca Quadrifoglio for serving as members of my committee and for their suggestions.

I thank all my colleagues in the research group and fellow students in the department as well as the staff members in Industrial & Systems Engineering Department and Engineering Technology & Industrial Distribution Department for providing friendly environment from which my research has benefited.

I sincerely thank my wife, Shirin, and my parents for their love and support. Without their encouragement and companionship I would have never been able to accomplish this work.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. BACKGROUND AND ANALYTICAL REVIEW

1.1 Definitions and notations

Consider a simple undirected graph $G = (V, E)$ with the set of vertices V and the set of edges E corresponding to pairs of vertices. Two vertices v and v' in G are said to be *adjacent* or *neighbors* if $(v, v') \in E$, in which case the edge (v, v') is said to be *incident* to v and v'. Let $N_G(v) = \{v' \in V : (v, v') \in E\}$ denote the *neighborhood* of a vertex v in G, and let $N_G[v] = \{v\} \cup N_G(v)$ be the *closed neighborhood* of v. The cardinality of the neighborhood, $|N_G(v)|$, is called the degree of v in G and is denoted by $deg_G(v)$. Let $\delta(G)$ and $\Delta(G)$ denote the minimum and the maximum degree of a vertex in G, respectively. We call a graph $G' = (V', E')$ a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. For a subset of vertices $S \subseteq V$, the *subgraph induced by* S, $G[S]$, is given by $G[S] = (S, E \cap (S \times S))$, where "$\times$" denotes the Cartesian product.

A *path* of length r between vertices v and v' in G is a subgraph of G given by an alternating sequence of distinct vertices and edges $v \equiv v_0, e_0, v_1, e_1, \ldots, v_{r-1}, e_{r-1}, v_r \equiv v'$ such that $e_i = (v_i, v_{i+1}) \in E$ for all $1 \leq i \leq r - 1$. A cycle of length r is defined similarly, by assuming that $v \equiv v'$ in the definition of a path. If there is at least one path between two vertices v and v' in G, then we say that v and v' are *connected* in G. A graph is called *connected* if any pair of its vertices is connected. Otherwise, a graph is called *disconnected*. The length of a shortest path between two connected vertices v and v' in G is called the *distance* between v and v' in G and is denoted by $d_G(v, v')$. If v and v' are not connected in G, then $d_G(v, v') = \infty$. The diameter $diam(G)$ of a graph G is given by the maximum distance between any pair of vertices in G, i.e., $diam(G) = \max_{v,v' \in V} d_G(v, v')$.

The *vertex connectivity* $\kappa(G)$ of G is the minimum number of vertices that need to

be deleted from G in order to obtain a disconnected or a trivial graph. The *density* $\rho(G)$ of G is given by $\rho(G) = |E|/\binom{|V|}{2}$. A *complete graph* K_n on n vertices is a graph that contains all possible edges, i.e, $\rho(K_n) = 1$. The *complement* $\bar{G}$ of G is $\bar{G} = (V, \bar{E})$, where $\bar{E}$ is the complement of E, i.e., $E \cap \bar{E} = \emptyset$ and $K_{|V|} = (V, E \cup \bar{E})$. A *clique* C is a subset of vertices such that $G[C]$ is a complete graph. An independent set I is a subset of vertices such that $G[I]$ has no edges. Clearly, $S \subseteq V$ is a clique in G if and only if S is an independent set in $\bar{G}$. A clique (independent set) is called *maximal* if it is not a subset of a larger clique (independent set). A *maximum clique (independent set)* of G is a clique (independent set) of the largest size in G, and the problem of finding a maximum clique (independent set) in a graph is called the *maximum clique (independent set) problem*. The size of a maximum clique in G is called the *clique number* of G and is denoted by $\omega(G)$. The size of a maximum independent set in G is called the *independence* number of G and is denoted by $\alpha(G)$. We have $\omega(G) = \alpha(\bar{G})$.

Some of the well known clique relaxation models are defined next. We assume that s and k are positive integer constants and $\lambda, \gamma \in (0, 1]$ are real constants. Let $S \subseteq V$. S is an *s-plex* if $\delta(G[S]) \geq |S| - s$. S is an *s-defective clique* if $G[S]$ contains at least $\binom{|S|}{2} - s$ edges. S is a *k-core* if $\delta(G[S]) \geq k$. S is a *k-block* if $\kappa(G[S]) \geq k$. S is a *γ-quasi-clique* if $\rho(G[S]) \geq \gamma$. S is a *(λ, γ)-quasi-clique* if $\delta(G[S]) \geq \lambda(|S| - 1)$ and $\rho(G[S]) \geq \gamma$. $D \subseteq V$ is called a *distance s-dominating set* if for any $v \in V \setminus D$ there exists $v' \in D$ such that $d_G(v, v') \leq s$. Distance 1-dominating set is called simply a *dominating set*.

In [84], the motivation behind some of the most popular clique relaxation models was analyzed in a systematic fashion, and a set of simple rules for defining meaningful clique relaxation structures was identified, yielding a methodical taxonomic framework for clique relaxations. The framework is based on the observation that

the clique can be defined using alternative equivalent descriptions via other basic graph concepts, such as distance, diameter, domination, degree, density, and connectivity. The corresponding equivalent definitions are referred to as *elementary clique defining properties*. Then, by applying some simple modifications to the elementary clique defining properties, one can reproduce the known clique relaxation models, as well as define new structures of potential practical interest. We will adhere to this framework in defining the distance-based clique relaxations formally as follows.

First, note that a subset of vertices C is a clique in G if and only if $d_G(v, v') = 1$, for any $v, v' \in C$ or, equivalently, $diam(G[C]) = 1$. These equivalent clique definitions constitute the elementary clique defining properties based on distance and diameter, respectively. In both cases, we have an equivalent characterization of a clique by setting a certain parameter (pairwise distance or diameter) to its minimum possible value. We can define the corresponding clique relaxations by restricting the violation of the respective elementary clique defining property, i.e., by allowing the pairwise distance or diameter to be greater than 1, but no greater than a constant positive integer $s > 1$. As a result, we obtain the following definitions.

Definition 1 (s-clique)**.** *Given a simple undirected graph $G = (V, E)$ and a positive integer constant s, a subset of vertices $S \subseteq V$ is called an s-clique if $d_G(v, v') \leq s$, for any $v, v' \in S$.*

Definition 2 (s-club)**.** *Given a simple undirected graph $G = (V, E)$ and a positive integer constant s, a subset of vertices $S \subseteq V$ is called an s-club if $diam(G[S]) \leq s$.*

An s-clique (s-club) is called maximal in G if it is not a subset of a larger s-clique (s-club) in G, and maximum in G if there is no larger s-clique (s-club) in G. The maximum s-clique (s-club) problem asks to find a maximum s-clique (s-club) in G. The size of the largest s-clique in G is called the *s-clique number* and is denoted by

$\tilde{\omega}_s(G)$. The size of the largest s-club in G is called the s-club number and is denoted by $\overline{\omega}_s(G)$.

According to the taxonomy in [84], clique relaxations based on restricting the violation of an elementary clique defining property can be standard or weak; absolute or relative; and structural or statistical. For a standard relaxation, we require the relaxed clique-defining property to hold in the *induced subgraph*, whereas the corresponding weak relaxation requires the same property to be satisfied within the *original graph* instead of the induced subgraph. Since s-clique is defined by restricting pairwise distances for its members in G, it is a weak relaxation, whereas s-club, which restricts distances in the induced subgraph, is a standard relaxation. Both s-clique and s-club are absolute relaxations, since the value of the constant s refers to the absolute bound on the distance in G or $G[S]$ and does not depend on the size of S. However, their relative version could easily be introduced by replacing the constant s in the definitions of s-clique and s-club with $\gamma|S|$, where $\gamma \in (0, 1)$ is a constant. Finally, both s-clique and s-club are structural clique relaxations and their statistical counterparts could be defined by requiring that the *average* pairwise distance between vertices for S in G or $G[S]$ is at most s. It should be noted that, in contrast to the structural relaxations, statistical relaxations generally impose little in terms of the group structure.

Higher-order clique relaxation models, which relax more than one elementary clique defining properties simultaneously, could also be defined using distance or diameter restrictions in addition to other requirements. Since the graph-theoretic notion of distance relies on paths, in addition to the simple higher order relaxations that combine multiple properties in a straightforward fashion, s-clique and s-club could also be involved in the so-called *k-hereditary* higher-order relaxations, with k-connectivity embedded within their structure. Namely, k-hereditary s-club and

s-clique can be defined as follows. Given $G = (V, E)$ and positive integers s and k, $S \subseteq V$ is called a k-hereditary s-club if $diam(G[S \setminus S']) \leq s$ for any $S' \subset S$ such that $|S'| \leq k$. Similarly, S is a k-hereditary s-clique if $d_G(v, v') \leq s$ for all $v, v' \in S \setminus S'$ for any $S' \subset S$ such that $|S'| \leq k$.

Next we introduce some graph classes for which the problems of interest have been explored in the literature. Consider a graph $G = (V, E)$. Given a cycle in G, its *chord* is an edge between two vertices of the cycle that is not a part of the cycle. A graph is called *chordal* if any cycle on at least 4 vertices has a chord. G is called a *k-partite* graph if V can be partitioned into k non-overlapping independent sets. If $k = 2$, a k-partite graph is *bipartite*. G is a *split graph*, if $V = V_1 \cup V_2$, where V_1 is a clique and V_2 is an independent set such that $V_1 \cap V_2 = \emptyset$. G is an *interval graph* if there exists a set of intervals $\mathcal{I} = \{I_v : v \in V\}$ on the real line such that $I_v \cap I_{v'} \neq \emptyset$ iff $(v, v') \in E$.

While the variations of distance-based relaxations just defined may potentially find interesting applications, we focus on the s-clique and s-club, referred to as *canonical* clique relaxation models for distance and diameter respectively in [84]. In addition, we consider biclique community detection problems with applications in biclustering and genome research.

Definition 3 (Biclique)**.** *Given a simple undirected graph $G = (V, E)$, an induced biclique of G is a pair (X, Y) with $X, Y \subset V$, $X \cap Y = \emptyset$ such that X and Y are stable sets and if $x \in X$ and $y \in Y$ then $(x, y) \in E$.*

In other words, an induced biclique of G is a complete bipartite subgraph of G. Note that if G is a bipartite graph then any biclique in G is induced. In the above definition, if at least one of X or Y is not required to be a stable set, then the pair (X, Y) is called a non-induced biclique of G. A biclique in G is said to be maximal

if it is not a subset of a larger biclique in G and maximum in G if there is no larger biclique in G. Note that in order for a bipartite graph to be a 2-club, it must be a biclique.

The objective of this chapter is to provide an up-to-date survey of known results concerning s-clique, s-club, biclique and the corresponding optimization problems, as well as to identify related open questions to explore. The remainder of the chapter is organized as follows. We start by introducing these models in sections 1.2, 1.3, and discuss the basic structural properties and the complexity results of the associated optimization problems in order to have a better understanding of the computational challenges one has to overcome in order to solve the problems of interest. Integer programming formulations proposed for the optimization problems and known polyhedral combinatorics associated with these formulations are reviewed. An overview on the solution methods that have been proposed for these problems, along with a brief review of the computational results is presented. Selected applications of the problems of interest are discussed in Sec. 1.4 and the chapter concludes with the objectives and open questions we aim to answer in this research.

1.2 Distance-based clique relaxation models

In 1949, Luce and Perry [69] introduced the clique concept to model the notion of a *cohesive subgroup* in social network analysis. Since then, cliques and the associated maximum clique problem have become ubiquitous and have been studied extensively in graph theory [37, 20, 21], theoretical computer science [46, 62] and operations research [15, 27, 22] from different perspectives. In graph-theoretic terms, a clique is a subset of vertices that are pairwise adjacent. The clique definition ensures the perfect reachabiliy between the group's entities, as they are directly linked to each other. Moreover, it also ensures that a clique has the highest possible degree

of each vertex, the highest possible connectivity, and the largest possible number of edges in the induced subgraph among all subsets of vertices of the same cardinality. However, the ideal cohesiveness properties of a clique put limitations on its applicability to situations where enforcing such properties is unnecessary or even prohibitive. For example, in transportation and telecommunication networks easy reachability between the members of a group (or a cluster) is of utmost importance, whereas a large number of edges is either costly to construct and maintain or results in operating inefficiencies, such as excessive interference.

To address particular practical aspects that cannot be suitably modeled by cliques, numerous clique relaxation models have been introduced in the literature that enforce certain elementary properties of cliques to be present, in a relaxed form, in the model of a cluster. The long list of the proposed models includes the distance-based clique relaxations called s-clique [68] and s-club [76], degree-based relaxations called s-plex [94] and k-core [93], and an edge density-based model known as quasi-clique [1] among many others. The focus of this chapter is on distance-based clique relaxations, s-clique and s-club.

Originally proposed by Luce [68] in 1950, s-clique was the historically first clique relaxation concept. This structure relaxes the requirement of having an edge (distance 1) between any pair of vertices from the group by allowing them to be at most distance s apart, thus ensuring that they can communicate via a path of at most $s-1$ intermediate vertices. Note that these intermediate vertices, while guaranteeing the reachability in at most s hops between vertices from an s-clique, do not have to be a part of the s-clique themselves, which may be considered a drawback from the cohesiveness standpoint. This was first pointed out by Alba [4], who proposed a definition of the so-called *sociometric clique of diameter* s, which was later refined by Mokken [76] under the name of *s-club*. Any two members of an s-club are required

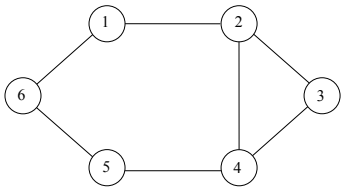

Figure 1.1: A graph illustrating structural differences of 2-cliques and 2-clubs.

to be connected by a path of length at most s, where all intermediate vertices belong to the s-club.

1.2.1 Structural properties and computational complexity

From the definitions, it is clear that an s-club is also an s-clique, however, the converse is not true in general. Even though the s-clique and s-club models appear to be very similar, there are some fundamental differences in their structural properties that have important implications for the associated optimization problems. To highlight these differences, consider a simple example in Fig. 1.1 that originally appeared in [4]. In the graph in this figure, a subset of vertices $\{1, 2, 3, 4, 5\}$ is a 2-clique, but not a 2-club, since the distance between vertices 1 and 5 is 3 in the induced subgraph. Moreover, $\{1, 2, 4\}$ is a 2-clique and a 2-club, $\{1, 2, 4\} \cup \{5\}$ and $\{1, 2, 4\} \cup \{6\}$ are both 2-cliques but not 2-clubs, whereas $\{1, 2, 4\} \cup \{5, 6\}$ is again a 2-clique and a 2-club. This shows the lack of any type of *heredity* for s-clubs, which is formally defined as follows [84]. A graph property Π is called *hereditary on induced subgraphs*, if for any graph G with property Π deleting any subset of vertices does not produce a graph violating Π. A graph property Π is called *weakly hereditary*, if for any graph $G = (V, E)$ with property Π all subsets of V posses the property Π in G. A graph property Π is said to be *nontrivial* if it is true for a single-vertex graph and is not satisfied by every graph. A graph property is said to be *interesting* if there are arbitrarily large graphs satisfying Π.

Unlike s-clubs, s-cliques posses weak heredity, which allows to reduce the problem of finding an s-clique to the problem of finding a clique in an auxiliary power graph defined as follows. Given a graph $G = (V, E)$, the s^{th} power of G, denoted by G^s, is given by $G^s = (V, E^s)$, where $E^s = \{(v, v') : 0 < d_G(v, v') \leq s\}$. Then $S \subseteq V$ is an s-clique in G if and only if S is a clique in G^s. Heredity on induced subgraphs is the core property implicitly exploited by some of the most successful combinatorial algorithms for the maximum clique problem [28, 79], which can also be applied to G^s in order to solve the maximum s-clique problem in G. Because of the presence of weak heredity, s-clique has advantage over s-club in terms of applicability of the variety of existing techniques available for the maximum clique problem to solving the maximum s-clique problem. However, this comes at a price. The fact that the s-clique is defined by restricting the distances in the original graph rather than the induced subgraph leads to the possibility of absence of any cohesiveness in the subgraph induced by an s-clique. For example, the subset of vertices $\{1, 3, 5\}$ in the graph on Fig. 1.1 is a 2-clique that induces an independent set, a structure that can hardly be considered cohesive by any standards. In terms of cohesiveness, the worst-case example of an s-club is a star graph, where one "central" vertex is adjacent to all other vertices, which have no neighbors other than the central vertex. While this structure appears to be quite fragile, as removing the central vertex makes it an independent set, it is still more cohesive than the worst-case example of an s-clique, which is an independent set to begin with. Since s-clique does not have to be connected in general, it makes sense to consider a *connected s-clique*, which is an s-clique that induces a connected subgraph.

Since s-clubs do not have any form of heredity defined above, the maximum clique algorithms cannot be easily adapted for the maximum s-club problem. In fact, the problem of finding a maximal s-club, which is very easy for clique and s-

clique, becomes challenging. Indeed, the problem of checking whether a given clique (s-clique) is maximal reduces to checking whether there is a vertex from outside that can be added to the clique (s-clique). However, the example above clearly shows that this strategy will not work for s-clubs. In fact, Mahdavi and Balasundaram [80] have recently shown that testing whether an s-club is maximal is NP-hard for any fixed integer $s \geq 2$. They have also identified sufficient conditions for every connected 2-clique to be a 2-club based on the concept of a *partitionable cycle*, which can be defined as follows. Consider two nonadjacent vertices v and v' in a cycle C in G. Removing these two vertices breaks the cycle into two paths, $P_1(v, v')$ and $P_2(v, v')$ with the vertex sets $V_1(v, v')$ and $V_2(v, v')$, respectively. If there exist v, v' such that $G[V_1(v, v')] = P_1(v, v')$ and $G[V_2(v, v')] = P_2(v, v')$ then C is called a partitionable cycle. If, in addition, $|V_1(v, v')| \neq |V_2(v, v')|$ then the partitionable cycle C is called asymmetric. Mahdavi and Balasundaram [80] have proved that if no subset of $5 \leq c \leq 2s + 1$ vertices induces an asymmetric partitionable cycle in G, where $s \geq 2$, then every connected s-clique is an s-club. This implies, in particular, that in a bipartite graph every connected 2-clique is a 2-club, which induces a complete bipartite subgraph. In cases where every connected s-clique is an s-club, checking maximality of an s-club reduces to checking maximality of a connected s-clique and hence is easy. Thus, discovering more of such cases is an interesting future research direction, which will provide further insights towards understanding the complexity of the problem.

The maximum clique problem is a classical NP-hard problem [46, 62], which is also hard to approximate. Recall that for a maximization problem with an optimal objective value given by $opt(G)$ on an input graph G, an algorithm $\mathcal{A}$ is called σ-approximation algorithm (or algorithm with approximation ratio σ) if $opt(G)/\mathcal{A}(G) \leq \sigma$ for every input graph G, where $\mathcal{A}(G)$ is the objective value out-

put by $\mathcal{A}$ when applied to G. It is known that the maximum clique size cannot be approximated in polynomial time within a factor of $n^{1-\epsilon}$ for any $\epsilon > 0$ unless $P = NP$ [10, 11, 102]. Since clique is a special case of s-clique and s-club, where $s = 1$, all these results apply to the versions of the maximum s-clique and maximum s-club problems that allow for *arbitrary* (non-fixed, instance-dependent) s. However, these results do not directly extend to the maximum s-clique and maximum s-club problems for the fixed constant parameter $s > 1$, which is given as a part of the problem definition rather than as an instance-dependent parameter. Therefore, in recent years there has been a considerable amount of research towards characterizing these problems in terms of their computational complexity in general and restricted graph classes.

Bourjolly et al. [24] use a reduction from CLIQUE to show that the maximum s-club problem is NP-hard for any fixed s. Balasundaram et al. [16] use an alternative reduction from CLIQUE to prove that both the maximum s-clique and maximum s-club problem are NP-hard, even if restricted to graphs of fixed diameter $s + 1$. Note that both problems are trivial when the graph's diameter is bounded above by s, therefore the transition in complexity is sudden.

Asahiro et al. [12] proved that for any $\epsilon > 0$ and a fixed $s \geq 2$ the maximum s-club problem is NP-hard to approximate within a factor of $n^{1/2-\epsilon}$ in general graphs, improving on the hardness of $n^{1/3-\epsilon}$-approximation result of Marinček and Mohar [72]. They also designed a simple polynomial-time algorithm that approximates the maximum s-club within a factor of $n^{1/2}$ for an even s, and within a factor of $n^{2/3}$ for an odd s. Given a graph $G = (V, E)$, the algorithm finds a maximum degree vertex in the power-$\lfloor s/2 \rfloor$ graph $G^{\lfloor s/2 \rfloor} = (V, E^{\lfloor s/2 \rfloor})$ and outputs its closed neighborhood in $G^{\lfloor s/2 \rfloor}$, which forms an s-club C_s of size $\Delta(G^{\lfloor s/2 \rfloor}) + 1$ in G. To establish the approximation ratio, they consider two cases, $\Delta(G) \geq n^{1/s}$ and $\Delta(G) < n^{1/s}$. Then

in the first case we have:

$$\frac{\overline{\omega}_s(G)}{|C_s|} = \frac{\overline{\omega}_s(G)}{\Delta(G^{\lfloor s/2 \rfloor}) + 1} \leq \frac{\overline{\omega}_s(G)}{\Delta(G) + 1} < n^{1-1/s}.$$

In the second case, noting that $\overline{\omega}_s(G) \leq 1+\Delta(G)+\Delta(G)^2+\ldots+\Delta(G)^s$, the following holds:

$$\frac{\overline{\omega}_s(G)}{|C_s|} \leq \frac{\Delta(G)^s + O(\Delta(G)^{s-1})}{\Delta(G) + 1} = O(\Delta(G)^{s-1}) = O(n^{1-1/s}).$$

Thus, in both cases the approximation ratio of the algorithm is $O(n^{1-1/s})$, which becomes $O(n^{1/2})$ for $s = 2$ and $O(n^{2/3})$ for $s = 3$. To show that the algorithm is, in fact $O(n^{1/2})$-approximate for any even $s \geq 4$, observe that

$$\overline{\omega}_s(G) \leq \overline{\omega}_2(G^{s/2}), \tag{1.1}$$

while the output of the approximation algorithm applied to the maximum s-club problem on G and to the maximum 2-club problem on $G^{s/2}$ is the same. Thus, the approximation ratio of $O(n^{1/2})$ holds for any even s.

It should be noted that [12] uses a stronger claim, $\overline{\omega}_s(G) = \overline{\omega}_2(G^{s/2})$ instead of (1.1) in the proof of the approximation ratio. However, the equality does not hold in general, i.e., we may have $\overline{\omega}_s(G) < \overline{\omega}_2(G^{s/2})$ as in the graph in Fig. 1.2. The proof still holds using the inequality (1.1) instead.

In addition to the above results, Asahiro et al. [12] proved that for any $\epsilon > 0$ the maximum s-club problem is NP-hard to approximate within a factor of $n^{1/3-\epsilon}$ for chordal and split graphs with even s, for bipartite graphs with $s \geq 3$, and for k-partite graphs ($k \geq 3$) with $s \geq 2$. On the other hand, the problem can be solved in polynomial time for chordal and split graphs with odd s, as well as for trees and interval graphs [12, 90]. Unlike the maximum s-club problem with $s \geq 3$,

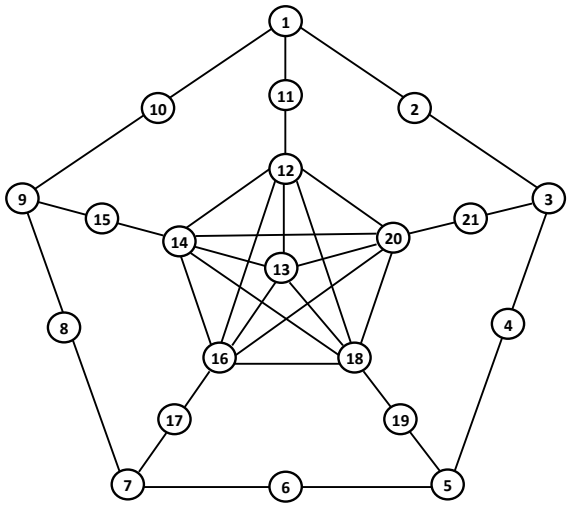

Figure 1.2: A graph with $\overline{\omega}_s(G) = 14$ (a maximum 2-club is given by, e.g., $C = \{5, 6, 7, 8, 9, 12, 13, 14, 16, 17, 18, 19, 20, 21\}$) and $\overline{\omega}_2(G^{s/2}) = 16$ (all vertices excluding 2,4,6,8,10 form the maximum 2-club in G^2), where $s = 4$.

the maximum 2-club problem can be solved in $O(n^5)$ on bipartite graphs [90]. In addition, the maximum 2-club can be approximated within a factor of $n^{1/3}$ for split graphs.

In several recent papers, the maximum s-club problem was approached from the *parameterized complexity* perspective [56, 57, 91, 30]. In this framework, one considers a parameter k (such as the size of a structure sought) in addition to the traditional input size n [40]. A parameterized problem is *fixed-parameter tractable* if there exists an algorithm (referred to as an fpt-algorithm) that solves the parameterized problem in time $f(k) \cdot n^{O(1)}$, where f is a computable (typically exponential) function that depends only on the parameter k.

It is known that deciding if a given graph contains a clique of size k is W[1]-complete [31], meaning that an fpt-algorithm is unlikely to exist. In contrast, Chang et al. [30] have shown that the problem of deciding if a given graph contains an s-club of size k is fixed-parameter tractable for $s > 1$. The proof is as follows. Let $G = (V, E)$ be the given graph. If $G^{\lfloor s \rfloor}$ has a vertex v such that $|N_{G^{\lfloor s/2 \rfloor}}[v]| \geq k$

then $N_{G^{\lfloor s/2 \rfloor}}[v]$ is an s-club of size at least k in G. Otherwise, $|N_{G^{\lfloor s/2 \rfloor}}[v]| < k$ for any $v \in V$, and it can be shown that $|N_{G^s}[v]| < k^2$ when s is even and $|N_{G^s}[v]| < k^3$ when s is odd [30]. Since any s-club C is a subset of $N_{G^s}[v]$ for any $v \in C$, in order to check whether G has an s-club of size k it suffices to check all k-element subsets of $N_{G^s}[v]$ for each $v \in V$. There are at most $\binom{k^3}{k} n$ such subsets, and checking whether a k-element vertex set forms an s-club can be done in k^3 time. Thus, the overall run time is $O(k^{3(k+1)}n)$.

Schäfer et al. [91] have shown that the maximum s-club problem is fixed-parameter tractable not only with the solution size k used as the parameter, but also when parameterized by the so-called dual parameter $d = |V| - k$. The algorithm they propose for this case runs in $O(2^d nm)$, where m is the number of edges in the graph. These ideas are extended to develop a practical algorithm for 2-club in [56], as will be discussed in more detail in Sec. 1.2.4. In addition, Hartung et al. [56] analyzed parameterized complexity of s-club with other parameters, such as the size of a vertex cover, feedback edge set size, size of a cluster editing set, and treewidth of the graph.

1.2.2 Mathematical programming formulations

In this section, mathematical programming formulations for the maximum s-clique and maximum s-club problems are presented. The maximum clique problem is one of the well studied problems in discrete optimization, with a number of known integer, as well as continuous non-convex formulations [22]. Similar formulations can be applied to the maximum s-clique problem on a graph G by reducing it to the maximum clique problem on the s^{th} power of G, $G^s = (V, E^s)$, constructed from the original graph as mentioned above. Let $\overline{E^s}$ denote the complement set of edges in G^s, i.e., $\overline{E^s} = \{(i,j) : i, j \in V, i < j, d_G(i,j) > s\}$. Then the following formulation of the maximum clique problem written for the power-s graph G^s can be used for the

maximum s-clique problem on G:

$$\text{Maximize (max)} \quad \sum_{i \in V} x_i \tag{1.2}$$

$$\text{subject to (s. t.):} \quad x_i + x_j \leq 1 \quad \forall (i,j) \in \overline{E^s} \tag{1.3}$$

$$x_i \in \{0,1\} \quad \forall i \in V. \tag{1.4}$$

The first mathematical program for computing the s-club number of a graph was proposed in [24]; see also [16]. In the following, we explain this general integer programming model first and then describe special cases for s=2,3 that are of highest practical interest and have received more attention in the literature. For $S \subseteq V$ the vector $x \in \{0,1\}^n$ such that $x_i = 1$ if and only if $i \in S$ is called the *characteristic vector* of S. In the general model (1.5)- (1.8) below, which is often referred to as *chain formulation*, the vector of decision variables x is the characteristic vector of the s-club sought. For every pair of vertices $i, j \in V$, let P^s_{ij} be the set of all paths of length at most s between i and j in G. We will denote by $\mathbb{P}$ the set of all such paths in G, i.e., $\mathbb{P} = \cup_{i,j \in V} P^s_{ij}$. Let V_P be the set of vertices included in a path P. Also let y_P be the auxiliary binary variable associated with every path $P \in \mathbb{P}$. If this variable is equal to 1 in a feasible solution, this implies that all the vertices in the path P are included in the corresponding s-club. Then the following finds the

maximum cardinality s-club in G:

$$\max \quad \sum_{i \in V} x_i \tag{1.5}$$

$$\text{s. t.:} \quad x_i + x_j \leq 1 + \sum_{P \in P_{ij}^s} y_P \quad \forall (i,j) \notin E \tag{1.6}$$

$$y_P \leq x_i \quad \forall P \in \mathbb{P}, \quad \forall i \in V_P \tag{1.7}$$

$$x_i, y_P \in \{0,1\} \quad \forall i \in V, \forall P \in \mathbb{P}. \tag{1.8}$$

In this formulation, constraint (1.6) ensures that two vertices i and j such that $d_G(i,j) > s$ cannot both belong to the same s-club (in this case $P_{ij}^s = \emptyset$ and the constraint becomes $x_i + x_j \leq 1$). It also guarantees that if two nonadjacent vertices are included in the s-club sought, then there must be at least one path of length at most s such that all the vertices from this path are also included in the s-club. In addition, constraint (1.7) forces y_P to be 0 whenever a vertex from P is not included in the s-club. For $s = 2$ the above chain formulation becomes:

$$\max \quad \sum_{i \in V} x_i \tag{1.9}$$

$$\text{s. t.:} \quad x_i + x_j \leq 1 + \sum_{k \in N_G(i) \cap N_G(j)} x_k \quad \forall (i,j) \notin E \tag{1.10}$$

$$x_i \in \{0,1\} \quad \forall i \in V. \tag{1.11}$$

This model ensures that any two nonadjacent vertices that are in the same 2-club must have at least one common neighbor inside the 2-club.

Considering the number of possible distinct paths of length at most s between every pair of vertices, the chain formulation may have an excessive number of variables when $s >2$. In general, we may have $|P_{ij}^s| = O(n^{s-1})$ for every pair of vertices,

so $|\mathbb{P}| = O(n^{s+1})$. Therefore, this model does not scale well when s increases, and even solving small instances using this formulation is challenging when $s \geq 3$ [99].

To formulate the maximum 3-club problem using a smaller number of variables, the *neighborhood formulation* (1.13)- (1.16) was proposed in [6] that has $|V| + |E|$ variables. Note that a pair of nonadjacent vertices i and j in G can be a part of the same 3-club S only if they have a common neighbor k in S or there are two adjacent vertices $\{p, q\} \in S$ such that $p \in N_G(i)$ and $q \in N_G(j)$. The first condition holds if and only if $d_{G[S]}(i, j) = 2$. If the first condition does not hold and the second condition holds then $p \in \{N_G(i) \setminus N_G(j)\}$ and $q \in \{N_G(j) \setminus N_G(i)\}$. Let E_{ij} denote the set of edges that connect such intermediate nodes for i and j:

$$E_{ij} = \{(p, q) \in E : p \in \{N_G(i) \setminus N_G(j)\}, q \in \{N_G(j) \setminus N_G(i)\}\} \quad \forall i, j : d_G(i, j) = 3. \tag{1.12}$$

Now associate a binary variable x_i with each vertex $i \in V$ and a binary variable z_{ij} with each edge $(i, j) \in E$. Then the maximum 3-club problem in $G = (V, E)$ can be formulated using the following binary program:

$$\max \quad \sum_{i \in V} x_i \tag{1.13}$$

$$\text{s. t.:} \quad x_i + x_j \leq 1 + \sum_{k \in N_G(i) \cap N_G(j)} x_k + \sum_{(p,q) \in E_{ij}} z_{pq} \quad \forall (i, j) \notin E \tag{1.14}$$

$$z_{ij} \leq x_i, \quad z_{ij} \leq x_j, \quad z_{ij} \geq x_i + x_j - 1 \quad \forall (i, j) \in E \tag{1.15}$$

$$x_i, z_{ij} \in \{0, 1\} \quad \forall i \in V, \forall (i, j) \in E. \tag{1.16}$$

Neighborhood constraints (1.14) ensure that two nonadjacent vertices i and j cannot be both in the solution unless their common neighbor is in the solution or a pair of their neighbors p and q, linked by an edge, are in the solution. The constraints

in (1.15) guarantee that an edge (i, j) is used if and only if both its endpoints belong to the solution. The neighborhood formulation has $|V| + |E|$ variables and $\frac{|V|^2-|V|}{2} + 2|E|$ constraints.

Almeida and Carvalho [6] also proposed another formulation for the maximum 3-club problem that is based on identifying minimal node cut sets for every pair of vertices with $d_G(i, j) = 3$. Consider a pair of nonadjacent vertices $i, j \in G$ and let E_{ij} be defined as in (1.12). Recall that E_{ij} is the set of inner edges of chains with length 3 connecting i and j with no common neighbors. Let V_{ij} represent the set of vertices incident to edges from E_{ij}. We associate with i and j a subgraph $G_{ij} = (V'_{ij}, E'_{ij})$ where $V'_{ij} = V_{ij} \cup \{i, j\}$ and $E'_{ij} = E_{ij} \cup \{(i, v) \in E : v \in V_{ij}\} \cup \{(j, v) \in E : v \in V_{ij}\}$. Figure 1.3 is an example of a subgraph G_{ij} in which $E_{ij} = \{(1, 2), (1, 5), (3, 4)\}$, $V_{ij} = \{1, 2, 3, 4, 5\}$ and $E'_{ij} = \{(1, 2), (1, 5), (3, 4), (i, 1), (i, 3), (2, j), (4, j), (5, j)\}$. Let S_{ij} be an i-j node cut set and define $\mathcal{S}^M_{ij}$ to be the set of all minimal S_{ij}. For our example in the figure, $\mathcal{S}^M_{ij} = \{\{1, 3\}, \{1, 4\}, \{2, 3, 5\}, \{2, 4, 5\}\}$. These sets separate i and j in G_{ij}, therefore, to include nodes i and j with $d_G(i, j) = 3$ in the same 3-club S, it is necessary to include a node of each set $S_{ij} \in \mathcal{S}^M_{ij}$. Thus, the *node cut set formulation* (1.17)- (1.19) for the maximum 3-club problem can be stated as follows:

$$\max \quad \sum_{i \in V} x_i \tag{1.17}$$

$$\text{s. t.:} \quad x_i + x_j \leq 1 + \sum_{k \in N_G(i) \cap N_G(j)} x_k + \sum_{s \in S_{ij}} x_s \quad \forall (i, j) \notin E, S_{ij} \in \mathcal{S}^M_{ij} \tag{1.18}$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \tag{1.19}$$

In this formulation, inequalities (1.18) are the node cut set constraints described above. The formulation has $|V|$ variables, but potentially exponential number of constraints. Note that constraints associated with non-minimal cut sets are domi-

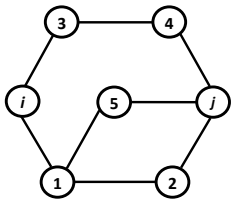

Figure 1.3: Subgraph G_{ij} illustrating the node cut set formulation.

nated by constraints (1.18) and are not necessary.

Next we present an integer programming formulation for the maximum s-club problem recently proposed by Veremyev and Boginski [99]. We first discuss the formulation for $s = 2$ and then extend it to the higher s values. Let $V = \{1, \ldots, n\}$ and let $A = [a_{ij}]_{i,j=1}^{n}$ be the adjacency matrix of $G = (V, E)$. Then the characteristic vector x of a 2-club S must satisfy the following nonlinear constraint:

$$a_{ij} + \sum_{k \in V} a_{ik} a_{kj} x_k \geq x_i x_j \quad \forall i, j \in V. \tag{1.20}$$

Linearizing this constraint, we formulate the maximum 2-club problem as follows:

$$\max \quad \sum_{i \in V} x_i \tag{1.21}$$

$$\text{s. t.:} \quad a_{ij} + \sum_{k=1}^{n} a_{ik} a_{kj} x_k \geq x_i + x_j - 1 \quad \forall i, j \in V \tag{1.22}$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \tag{1.23}$$

The above formulation can be simplified as follows:

$$\max \quad \sum_{i \in V} x_i \tag{1.24}$$

$$\text{s. t.:} \quad \sum_{k \in N_G(i) \cap N_G(j)} x_k \geq x_i + x_j - 1 \quad \forall (i,j) \notin E \tag{1.25}$$

$$x_i \in \{0,1\} \quad \forall i \in V. \tag{1.26}$$

Similarly, the characteristic vector x of a 3-club must satisfy the following nonlinear constraints:

$$\sum_{k=1}^{n} a_{ik} a_{kj} x_k + \sum_{k=1}^{n} \sum_{m=1}^{n} a_{ik} a_{km} a_{mj} x_k x_m \geq x_i x_j \quad \forall (i,j) \notin E. \tag{1.27}$$

Letting $w_{ij} = x_i x_j$ for all $i, j \in V$ and linearizing the constraints, we obtain the following formulation for the maximum 3-club problem:

$$\max \quad \sum_{i=1}^{n} x_i \tag{1.28}$$

$$\text{s. t.:} \quad \sum_{k=1}^{n} a_{ik} a_{kj} x_k + \sum_{k=1}^{n} \sum_{m=1}^{n} a_{ik} a_{km} a_{mj} w_{km} \geq x_i + x_j - 1 \ \ \forall (i,j) \notin E \tag{1.29}$$

$$w_{ij} \leq x_i, w_{ij} \leq x_j, w_{ij} \geq x_i + x_j - 1 \quad \forall i, j \in V \tag{1.30}$$

$$x_i, w_{ij} \in \{0,1\} \quad \forall i, j \in V. \tag{1.31}$$

This formulation contains $O(n^2)$ binary variables and $O(n^2)$ constraints. Similarly, one can develop the model and linearize it using the standard approaches for the general case of the maximum s-club problem. However, the resulting formulation will have $O(n^{s-1})$ variables. Veremyev and Boginski [99] exploit the special structure

of s-club and propose an efficient linearization technique that reduces the number of variables substantially. We discuss their compact binary formulation next.

Consider a subset of vertices S and its characteristic vector x. Let $v_{ij}^{(l)}, (i, j = 1, \ldots, n; l = 2, \ldots, s)$ be a binary variable taking the value 1 if there exists at least one path of length l from i to j in $G[S]$ and 0 otherwise. Note that for $l = 2$ we have $v_{ij}^{(2)} = min\{x_i x_j \sum_{k=1}^{n} a_{ik} a_{kj} x_k, 1\}$, which can be linearized using the following set of constraints:

$$\begin{cases} v_{ij}^{(2)} \leq x_i, \quad v_{ij}^{(2)} \leq x_j \\ v_{ij}^{(2)} \leq \sum_{k=1}^{n} a_{ik} a_{kj} x_k, \quad v_{ij}^{(2)} \geq \frac{1}{n} \left(\sum_{k=1}^{n} a_{ik} a_{kj} x_k \right) + (x_i + x_j - 2). \end{cases}$$

Other variables for higher values of $l = 3, \ldots, s$ can be found recursively using $v_{ij}^{(l)} = \min\{x_i \sum_{k=1}^{n} v_{kj}^{(l-1)} a_{ik}, 1\}$ and linearized by applying the following set of inequalities:

$$\begin{cases} v_{ij}^{(l)} \leq x_i, \quad v_{ij}^{(l)} \leq \sum_{k=1}^{n} a_{ik} v_{kj}^{(l-1)} \\ v_{ij}^{(l)} \geq \frac{1}{n} \left(\sum_{k=1}^{n} a_{ik} v_{kj}^{(l-1)} \right) + (x_i - 1). \end{cases}$$

Therefore the maximum s-club problem can be formulated as the following binary

linear program:

$$\max \quad \sum_{i=1}^{n} x_i \tag{1.32}$$

s. t.:

$$\sum_{l=2}^{k} v_{ij}^{(l)} \geq x_i + x_j - 1 \quad \forall (i,j) \notin E \tag{1.33}$$

$$v_{ij}^{(2)} \leq x_i, \quad v_{ij}^{(2)} \leq x_j, \quad v_{ij}^{(2)} \leq \sum_{k=1}^{n} a_{ik} a_{kj} x_k \quad \forall i,j \in V, i < j \tag{1.34}$$

$$v_{ij}^{(2)} \geq \frac{1}{n} \left(\sum_{k=1}^{n} a_{ik} a_{kj} x_k \right) + (x_i + x_j - 2) \quad \forall i,j \in V, i < j \tag{1.35}$$

$$v_{ij}^{(l)} \leq x_i, \quad v_{ij}^{(l)} \leq \sum_{k=1}^{n} a_{ik} v_{kj}^{(l-1)} \quad \forall i,j \in V, i < j, l = 3, \ldots, s \tag{1.36}$$

$$v_{ij}^{(l)} \geq \frac{1}{n} \left(\sum_{k=1}^{n} a_{ik} v_{kj}^{(l-1)} \right) + (x_i - 1) \quad \forall i,j \in V, i < j, l = 3, \ldots, s \tag{1.37}$$

$$x_i, v_{ij}^{(l)} \in \{0,1\} \quad \forall i,j \in V, i < j, l = 2, \ldots, s. \tag{1.38}$$

The above model is the most compact known formulation for the maximum s-club problem with $O(sn^2)$ variables and constraints. For more information about compact formulation and its properties, we refer the reader to [99].

1.2.3 Polyhedral results

Due to the structure of the problem and its dependence on the value of parameter s, most of the research in this area has been focused on the 2-club polytope and, partially, 3-club polytope and not on the s-club polytope in general. In this section, we review the polyhedral results available for the 2-club polytope.

Consider a nontrivial simple undirected connected graph $G = (V, E)$. A subset of vertices $I \subseteq V$ is a *2-independent set* in G if $d_G(i,j) > 2 \quad \forall i,j \in I$. Let M_1 be the edge-vertex incidence matrix of the complement graph $\bar{G}$. The rows of M_1

correspond to edges $(i,j) \in \bar{E}$ and the columns correspond to vertices $i \in V$. The entries in the row corresponding to an edge (i,j) are 1 in columns i and j and are 0 otherwise. Let M_2 be the matrix representing the common neighborhood of i,j for every $(i,j) \in \bar{E}$. The rows of M_2 correspond to edges $(i,j) \in \bar{E}$ and the columns correspond to vertices $i \in V$. The entries in the row corresponding to an edge (i,j) are 1 in columns $k \in N_G(i) \cap N_G(j)$ and are 0 otherwise. Let $A = M_1 - M_2$, then the maximum 2-club problem formulation (1.9)-(1.11) can be written as [16]:

$$\bar{\omega}_2(G) = max\{\mathbf{1}^T x : Ax \leq \mathbf{1}, x \in \{0,1\}^{|V|}\},$$

where $\mathbf{1}$ is the vector of all ones of appropriate dimension and $\bar{\omega}_2(G)$ is the 2-club number of G. Let Q be the set of feasible binary vectors defined as $Q = \{x \in \{0,1\}^{|V|} : Ax \leq \mathbf{1}\}$, then the 2-club polytope P_{2C} is given by the convex hull of Q: $P_{2C} = conv(Q)$. The following results were established in [16].

1. $dim(P_{2C}) = |V|$.
2. $x_i \geq 0$ induces a facet of P_{2C} for every $i \in V$.
3. For any $i \in V$, $x_i \leq 1$ induces a facet of P_{2C} if and only if $d_G(i,j) \leq 2 \quad \forall j \in V$.
4. Let I be a maximal 2-independent set in G. Then $\sum_{i \in I} x_i \leq 1$ induces a facet of P_{2C}.

Note that each neighborhood constraint is associated with two vertices v and v' such that $d_G(v,v') = 2$. For any node $i \in V \backslash \{v,v'\}$ such that $\min\{d_G(i,v), d_G(i,v')\} > 2$, the inequality $x_v + x_{v'} + x_i - \sum_{j \in \{N_G(v) \cap N_G(v')\}} x_j \leq 1$ is valid for P_{2C} because neither v nor v' can be included in a 2-club that includes node i, and to include nodes v and v', at least one of their common neighbors must also be included in the 2-club.

This inequality is a lifted version of the neighborhood constraint (1.10). Carvalho and Almeida [29] used this observation to establish the following valid inequality for P_{2C}:

$$\sum_{i \in I \cup \{v,v'\}} x_i - \sum_{j \in N_G(v) \cap N_G(v')} x_j \leq 1, \tag{1.39}$$

where $I \subseteq V \setminus \{v, v'\}$ is such that $I \cup \{v\}$ and $I \cup \{v'\}$ are 2-independent sets in G. They also extended this result to triples of vertices as follows. Let v, v', v'' be given. For a vertex j denote by $a_j = (|N_G(j) \cap \{v, v', v''\}| - 1)^+$, where $a^+ = \max\{a, 0\}$, i.e., $a_j = 2$ if j neighbors all three vertices; $a_j = 1$ if j neighbors two of the three vertices; and $a_j = 0$, otherwise. Let $R = \{v, v', v''\}$ be an independent set in G. Let $I \subseteq V \setminus \{v, v', v''\}$ be such that $I \cup \{v\}$, $I \cup \{v'\}$ and $I \cup \{v''\}$ are 2-independent sets in G. Then the inequality

$$\sum_{i \in I \cup \{v,v',v''\}} x_i - \sum_{j \in V} (|N_G(j) \cap \{v, v', v''\}| - 1)^+ x_j \leq 1 \tag{1.40}$$

is valid for P_{2C}.

More recently, Mahdavi [71] developed a family of valid inequalities that subsume both (1.39) and (1.40).

Theorem 1 ([71])**.** *Let I be an independent set in G. Then the inequality*

$$\sum_{i \in I} x_i - \sum_{j \in V \setminus I} (|N_G(j) \cap I| - 1)^+ x_j \leq 1 \tag{1.41}$$

is valid for P_{2C}. If, in addition, I is a distance 2-dominating set, i.e., I is an independent distance 2-dominating set (I2DS), then this inequality defines a facet for P_{2C} referred to as I2DS facet.

Mahdavi [71] also proved that given a noninteger feasible point $\tilde{x}$ in P_{2C} deciding

whether this point violates an I2DS inequality is NP-complete, i.e., the I2DS inequalities separation problem is NP-complete. On a positive note, I2DS inequalities are sufficient to derive the complete description of the 2-club polytope of trees. See [71] for a more detailed discussion on the 2-club polytope.

As for the maximum 3-club problem, Almeida and Carvalho [6] developed some non-trivial valid inequalities based on ideas similar to those used to develop (1.39) and (1.40) above. An interesting future research question is whether the valid inequalities they developed for 3-club can be generalized to develop a class of facets similar to I2DS above.

1.2.4 Exact and heuristic algorithms

The correspondence between the maximum clique and maximum s-clique problems implies that the heuristic and exact algorithms for maximum clique problem can be applied to the s^{th} power of the graph to solve the maximum s-clique problem. In such cases the performance of these algorithms may be poor as the edge density is higher in G^s. Unlike the maximum clique problem, the maximum s-clique problem has not been the subject of extensive research and we are not aware of any computational results for this problem to date. This may be due to the above-mentioned correspondence between the two problems which facilitates the use of proposed algorithms for clique detection to solve the later case. Therefore, in this section our focus will primarily be on the existing algorithms for the maximum s-club problem.

Due to computational intractability of the maximum s-club problem, heuristics become the method of choice for solving the maximum s-club problem in practice. Several construction heuristics have been proposed in the literature. In 2000, Bourjolly et. al [23] proposed three simple heuristic algorithms for the maximum s-club problem called DROP, CONSTELLATION and s-CLIQUE-DROP. Among these,

DROP, which runs in $O(|V|^3|E|)$ time, was reported to produce the best result in terms of solution quality specially in graphs with higher density. DROP works as follows: we start with the whole graph G and at each iteration the vertex i with most infeasibility is deleted where infeasibility is defined as the number q_i of vertices of G whose shortest distance to i has a length of at least s+1. If there is a tie, a vertex of minimum degree is then selected for elimination and the graph is updated. The procedure continues until no infeasible vertex can be found. CONSTELLATION is based on identifying the largest star graph in the first step. In the next iteration the vertex having the largest number of neighbors in the remaining graph is selected and added to the s-club provided that the total number of iterations does not exceed s-1. CONSTELLATION runs in $O(s(|V| + |E|))$ time and was reported to perform well solving the maximum 2-club problem on low density graphs. The third algorithm, s-CLIQUE-DROP, proceeds by identifying the largest s-clique in G and removing all vertices not belonging to s-clique from G along with their incident edges. Then DROP is called to find a feasible solution. To obtain the largest s-clique, the maximum clique problem is solved on G^s using one of the existing algorithms. Another simple heuristic, named IDROP, was recently proposed in [30].

The first exact algorithm for the maximum s-club problem was proposed by Bourjolly et al. [24]. The proposed branch-and-bound (B&B) algorithm employs DROP heuristic to direct its branching process. For the bounding process, algorithm relies on the solutions to the maximum stable set problem solved on an auxiliary graph. Two branches are generated at the root node of the search tree that correspond to removing or keeping the vertex selected by a single iteration of DROP. The algorithm first explores the branch that removes the vertex. The process is then recursively applied until a terminal node is reached, yielding a depth first search. Note that deciding to remove or keep a vertex during the branching process may increase the

shortest chains length and this affects the whole subtree rooted at the node in which this decision has been made. As a result, a pair of vertices in the current solution at some node of the subtree may appear too far away from each other. This leads to an infeasible solution and thus the corresponding branch is pruned. For the upper bounding procedure, let $G' = (V', E')$ be the graph induced by the current solution at a given node of the B&B tree and let $H = (V', F)$ be an auxiliary graph associated with G', where there is an edge between any two vertices in H only if the shortest path connecting these two vertices in G' has length greater than s. Obviously, if there is an edge between two vertices in H, they cannot both belong to the same s-club in G'. Therefore the largest independent set in H provides an upper bound on the size of the largest s-club in G'. In their computational results, authors report the average solution size and CPU time for s=2,3,4 on randomly generated instances with different edge densities. Instances were generated using the method proposed in [49].

Recently, Chang et al. [30] have shown that the B&B algorithm of Bourjolly et al. [24] runs in $O(1.62^n)$ time and proposed a variation of this algorithm that uses IDROP procedure to find the initial feasible solution and computes the s-coloring number of the graph associated with each node of the B&B tree to obtain an upper bound on the size of the s-club for that node. Observe that the *chromatic number* $\chi(G)$ of G is the minimum number of colors required to color the vertices of G *properly*, i.e., so that no two neighbors are assigned the same color and the s-coloring number of G is the minimum number of colors required to color all vertices such that no two vertices of distance at most s are assigned the same color. Note that the s-coloring number of G provides an upper bound on the s-club number of G and one can compute the chromatic number $\chi(G^s)$ of the s^{th} power graph G^s to obtain the s-coloring number of G. The authors report the results of computational experiments

with a set of randomly generated instances, Erdös collaboration networks [52, 17], and some benchmark graphs from the second DIMACS implementation challenge [38].

More recently Mahdavi and Balasundaram [80] proposed another B&B algorithm to compute the s-club number of a graph. Their algorithm employs two methods for computing a lower bound. The first method selects the larger of the two solutions found using DROP and CONSTELLATION heuristics, and the second method is a bounded enumeration-based technique. This lower-bounding scheme proceeds by finding an initial s-club S followed by a bounded search that enumerates s-clubs containing S. The idea behind this bounded search is to improve the initial solution in a reasonable amount of time. The best solution found by these two methods initializes the incumbent. Two methods are used to derive an upper bound for the B&B algorithm. The first one, proposed independently of [30], is based on obtaining the s-coloring number of graph associated with every node in the B&B tree. To obtain this upper bound a combination of greedy heuristic and DSATUR heuristic, proposed in [25], is used. The second method computes the maximum s-clique to serve as an upper bound for the s-club number of a given graph G. To obtain this upper bound the algorithm proposed by [79] is employed to find the maximum clique on s^{th} power graph G^s. For branching, a vertex dichotomy is used, where a vertex is selected and fixed to be included or deleted from the solution. To traverse the search tree, best bound search (BBS) strategy has been considered. Authors report extensive computational results, for s=2,3, using four different combinations of lower-bounding and upper-bounding techniques on a set of randomly generated instances of order n=50, 100, 150 and 200 with seven different densities ranging from 0.0125 up to 0.25. They report on the effectiveness of the bounding techniques used in the B&B algorithm and their relation with the topological structure of the randomly generated instances.

Veremyev and Boginski [99] solved the maximum s-club problem for $s = 2, \ldots, 7$, using the compact formulation (1.32)-(1.38) on a set of randomly generated instances of order n=100, 200, 300 with different edge densities. For every combination 10 instances are generated and the average maximum s-club size, average CPU time and the average tightness for each group of problem instances have been reported. The advantage of the compact formulation is that it contains a reasonable number of entities that grows linearly as s increases, thus providing an opportunity to solve the problem for higher values of s. Computational results show that the compact formulation is rather tight and the relative gap between the exact and the LP relaxation objective values decreases for larger values of s. The results of experiments with IP-based approaches for $s = 2, 3$ have also been reported in [6, 29].

Hartung et al. [56] used their theoretical findings concerning parameterized algorithms for 2-clubs based on the dual parameter $d = |V| - k$ to develop a B&B strategy in conjunction with a kernelization proposed in [91]. The results of experiments with the proposed algorithm for the maximum 2-club problem that they report are very encouraging. In particular, their implementation significantly outperforms other known exact approaches on small to medium-size random graphs and large-scale real-life networks from the tenth DIMACS implementation challenge [39].

1.3 The biclique model

Networks provide a convenient modeling tool for representation and analysis of the interaction between elements of a complex system. Biological networks are examples of such systems. In protein-protein interaction networks, proteins are represented as vertices and physical interaction between two proteins is represented by an edge [43, 61, 96]. In genome research, the relationship between genes and diseases, or other experimental conditions like treatments, can be modeled using graphs in which

genes and diseases are represented by vertices and edges represent a significant relationship between a gene and a disease [70, 78]. Biclique community detection has attracted a lot of attention in recent years due to its various applications in automata and language theory, biology and genome research, clustering and data mining, artificial intelligence and graph compression [2, 9, 32, 34, 64, 67, 70, 89]. These applications are motivated by different variants of the biclique community detection problems in the literature which will be defined next.

The *maximum edge biclique problem* (MEB) is concerned with finding the maximum edge cardinality biclique in G. MEB is a special case of the *maximum edge weight biclique problem* (MEWB) and has been applied successfully for biclustering and formal concept analysis [44, 70, 78]. The *maximum vertex biclique problem* (MVB) is to find the maximum vertex cardinality biclique in G and is a special case of the *maximum vertex weight biclique problem* (MVWB). A biclique is said to be balanced if the two bipartitions have the same cardinality. The *maximum balanced biclique problem* (MBB) is to find the maximum vertex cardinality biclique that is balanced. Next we review the known results about the complexity of these problems.

1.3.1 Computational complexity

From the definition, it is obvious that every biclique is a 2-club but the converse is not true. Bicliques preserve heredity property which allows the design of effective combinatorial algorithms. One such method has been proposed in Chap. 4.

To discuss the complexity for variants of the maximum biclique problem we consider two graph classes, the general simple graphs and bipartite graphs. Peeters [85] proved that the *maximum edge biclique problem* is NP-complete in bipartite and general graphs using a reduction from CLIQUE problem. Nussbaum et al. [78] proved the polynomial solvability of the problem in convex bipartite and biconvex graphs.

Under some plausible assumptions, MEB is hard to approximate within a factor of $O(n^{\epsilon})$ [8, 41, 42]. The weighted version of the problem (MEWB) was shown to be NP-hard in [34] and hard to approximate [97]. Hochbaum [58] considered a related problem in which the objective is to remove the minimum number of edges or vertices such that the remaining graph is a biclique. For the edge deletion version of the problem and based on the solution from LP-relaxation, a 2-approximation algorithm for bipartite and general graphs is proposed, and for the node deletion version in general graphs a 2-approximation algorithm is provided. Independently, Haemers [53] developed an upper bound on the size of maximum edge biclique using eigenvalues of the matrix representation of the underlying graph.

The *maximum vertex biclique problem* and its weighted version (MVWB) are polynomial time solvable in bipartite graphs [34, 46] and NP-complete in general simple graphs. The following result from Yannakakis [101] can be used to determine the computational complexity for a class of optimization problems in graph theory. Given a graph property Π, the *maximum* Π problem is to find the largest order induced subgraph that does not violate property Π. Yannakakis proved a general complexity result for such properties Π that can be stated as follows.

Theorem 2 ([101])**.** *The maximum* Π *problem for nontrivial, interesting graph properties that are hereditary on induced subgraphs is NP-hard.*

Observe that the complete bipartite subgraph is an example of Π with nontrivial, interesting and hereditary properties. Therefore finding the largest induced subgraph that is biclique, MVB, is NP-complete. Using the above theorem, the same conclusion can be drawn for the optimization problems of finding edgeless, planar, complete, perfect and bipartite subgraphs. Also note that the MVB is a special case of the MVWB, implying that the latter is also NP-complete in general graphs. As another

special case, the *maximum balanced biclique problem* has been proved to be NP-complete [46].

1.3.2 Mathematical programming formulations and algorithms

In this section we review some of the mathematical formulations and algorithms proposed for variants of the maximum biclique problem. Our focus here is mainly on algorithms that employ mathematical programming formulations of the MBP variants in the solution procedure. Consider a bipartite graph $B = (V_1 \cup V_2, E)$ with a weight w_v associated with each vertex. The following formulation was proposed for the maximum vertex weight biclique problem in bipartite graphs [34] where x_v is the binary variable with value one if vertex v is in the biclique and zero otherwise.

$$\text{Maximize} \quad \sum_{u \in V_1} w_u x_u + \sum_{v \in V_2} w_v x_v \tag{1.42}$$

$$\text{s. t.:} \quad x_u + x_v \leq 1 \quad u \in V_1, v \in V_2, (u,v) \notin E \tag{1.43}$$

$$x_v \in \{0,1\} \quad \forall v \in V_1 \cup V_2. \tag{1.44}$$

Note that the constraint set matrix of the above binary program is node-edge incidence matrix of a bipartite graph and is unimodular. Therefore the solution to the LP-relaxation of the problem is integer. As a result, MVWB is polynomially solvable in bipartite graphs. The same result holds for MVB in bipartite graphs.

Next we introduce a class of integer programs with an interesting property. Integer programs with up to three variables per inequality, called IP2, were considered in [58, 59]. There is no limitation in the number of times two of these variables can appear in other constraints but the third one may appear only once. As an example

consider inequalities of type

$$a_i x_{j_i} - b_i x_{k_i} \leq c_i + d_i z_i \quad i = 1, ..., m \tag{1.45}$$

in which all variables are binary, and the only limitation is that z_i can appear only once. It is assumed that all the constraint coefficients may take arbitrary rational values except d_i's that need to be integer.

Definition 4. *An inequality of type* (1.45) *is monotone if* $a_i, b_i \geq 0$ *and* $d_i = 1$.

Theorem 3 ([59])**.** *An IP2 problem on monotone constraints is solvable in integers in the time required to solve a minimum cut, or maximum flow, problem on a graph with* $O(n)$ *nodes and* $O(m)$ *edges.*

The above property has been used to design efficient algorithm for non-induced MVWB in general graphs [58]. In non-induced version, bipartitions are not required to be independent sets. In this formulation, the assumption is that the biclique contains two adjacent vertices s and t each in one bipartition to avoid having an empty bipartition. Therefore the formulation is given for each possible choice of

such edge.

$$\text{Maximize} \quad \sum_{j \in V} w_j y_j^{(1)} + \sum_{j \in V} w_j y_j^{(2)} \tag{1.46}$$

$$\text{s. t.:} \quad 1 - x_j \geq 2y_j^{(1)} \quad \forall j \in V \tag{1.47}$$

$$1 + x_j \geq 2y_j^{(2)} \quad \forall j \in V \tag{1.48}$$

$$x_i - x_j \leq 1 \quad \forall (i,j) \notin E \tag{1.49}$$

$$x_j - x_i \leq 1 \quad \forall (i,j) \notin E \tag{1.50}$$

$$x_s = 1, x_t = -1 \tag{1.51}$$

$$x_j \in \{-1, 0, 1\}, y_j^{(1)}, y_j^{(2)} \in \{0, 1\} \quad \forall j \in V. \tag{1.52}$$

The variables x_j are associated with the vertices and can have three possible states. If vertex j belongs to either of the bipartitions, variable x_j will take values 1 or -1 depending on which side of the bipartition it belongs to and, otherwise, zero. Binary variable $y_j^{(1)}$ will take value 1 if $x_j = -1$ and zero otherwise. Likewise $y_j^{(2)}$ is equal to 1 only if $x_j = 1$. Therefore the binary variables determine the contribution of a vertex to the objective function. Using the arguments defined earlier, Hochbaum [58] showed that the above formulation is monotone and is thus solvable in integers in $O(mT(n, \binom{n}{2} - m))$ where n and m are the number of nodes and edges in G, respectively. An alternative formulation, node deletion version, is also provided. This model minimizes the total weight of vertices that should be removed such that the remaining subgraph induces a non-induced biclique. Formulation is given for each possible choice of an edge relying on the fact that the optimal biclique contains at least one edge $(s,t) \in E$ and its incident vertices. The alternative formulation provides the basis for the proposed approximation algorithm. The method proceeds with solving the formulation for each edge $(s,t) \in E$ and selecting the solution with

the smallest objective function. Note that for each edge in G, the IP formulation, or the network model, must be set up and solved once but the monotonicity of the formulation allows the use of network flow algorithms, as noted in Theorem 3, and having an integer solution.

For the induced version of the problem, induced MVWB, a node deletion model is proposed. The objective is to minimize the total weight of the vertices that should be removed such that the remaining subset of vertices in G forms an induced biclique. A binary variable x_j is equal to 1 only if vertex j is deleted and zero otherwise. The common neighborhood of vertices s and t is defined as $N_G(s,t) = N_G(s) \cap N_G(t)$, whereas $N_G^{'}(s) = N_G(s) \setminus N_G(s,t)$, $N_G^{'}(t) = N_G(t) \setminus N_G(s,t)$, and $V(s,t) = N_G^{'}(s) \cup N_G^{'}(t)$. A bipartite graph is formed based on $N(s), N(t)$ for all $(s,t) \in E$ and since each bipartition must be an independent set, all vertices in $N(s,t)$ must be removed from the graph. We have the following formulation:

$$\text{Minimize} \quad \sum_{j \in V_{s,t}} w_j x_j \tag{1.53}$$

$$\text{s. t.:} \quad x_i + x_j \geq 1 \quad \forall (i,j) \notin E, i \in N_G^{'}(t), j \in N_G^{'}(s) \tag{1.54}$$

$$x_i + x_j \geq 1 \quad \forall (i,j) \in E, \quad i,j \in N_G^{'}(t) \tag{1.55}$$

$$x_i + x_j \geq 1 \quad \forall (i,j) \in E, \quad i,j \in N_G^{'}(s) \tag{1.56}$$

$$x_j \in \{0,1\} \quad \forall j \in V_{s,t} \tag{1.57}$$

The first constraint ensures that if there is any missing edge in the bipartition, at least one of its two end points cannot be in the solution. The other two constraints make sure that vertices in each bipartition form an independent set. In the current format, the above formulation is not monotone but it is equivalent to the vertex cover on a graph including edges induced by $N_G^{'}(s)$ and $N_G^{'}(t)$ and the complement of the

edge set in the bipartition. Therefore it is enough to solve m vertex cover problems, which is 2-approximable in polynomial time, and select the one with minimum objective function. Using the same techniques, approximation algorithms are provided for the maximum edge biclique problem [58].

In addition to approximation algorithms, other methods have been proposed for biclique community detection ranging from enumeration of all maximal bicliques of a graph to exact exponential time methods and mining quasi-bicliques [5, 19, 48, 58, 67]. Liu et al. [67] propose a divide-and-conquer approach for finding large maximal bicliques. Their algorithm uses the size constraints on both sides of the biclique to iteratively prune the search space, mainly the non-maximal and duplicate bicliques. Authors report brief computational experiments using instances from second DIMACS challenge and compare the results with other algorithms. Alexe et al. [5] propose a *consensus* algorithm for finding all maximal bicliques of a graph. This method starts with a collection C of bicliques that cover the edge set of a given graph G and proceeds with a sequence of transformations, named *absorption* and *consensus adjunction*, to find maximal bicliques. The algorithm stops when no more transformation on C is possible.

1.4 Applications and extensions

1.4.1 s-clique and s-club

The introduction of the concepts of s-clique and s-club was originally motivated by applications in social networks analysis, where these distance-based clique relaxations are used to model cohesive subgroups [4, 76]. A social network can be formalized by a simple undirected graph $G = (V, E)$. The vertex set V can represent people, or actors, in a social network and the mutual relationships between pairs of actors can be naturally modeled using edges. For example, in a collaboration network

the edges could represent collaborations between researchers. For mathematicians and computational geometers [52, 17] such collaboration networks are used to determine the collaborative distance between researchers which was first popularized by the concept of Erdös numbers [51].

Studying cohesive or "tightly knit" subgroups, which describe groups of actors that tend to share certain features of interest [92, 100], finds applications in different branches of sociology, including epidemiology of sexually transmitted diseases [86], organizational management [36], and crime detection/prevention and terrorist network analysis [88, 87, 18] among many others. For example, in [73], s-cliques and s-clubs are used to analyze 9/11 terrorist network.

Even though the distance-based clique relaxation structures may not be characterized by a very high overall degree of interaction between their members that is typical for some other models, the low distances between all group members make them appropriate models of cohesive subgroups in situations where easy reachability is most crucial. This is the case, in particular, when one deals with various types of *flows* in the network, such as flows of information, spread of diseases, or transportation of commodities. It is therefore not surprising that s-cliques and s-clubs appear naturally in many real-life complex systems, including biological and social systems, as well as telecommunication, transportation, and energy infrastructure systems.

In social networks, the proliferation of low-diameter structures manifests itself in catch-phrases "small world phenomenon" and "six degrees of separation" that made their way to the mainstream popular culture. A low diameter is a key characteristic of many other massive-scale complex networks that tend to have *power-law* degree distribution, or the so-called *scale-free* property [77]. Such networks typically have a small number of high-degree nodes, which are most likely to be "central vertices" in the largest s-clubs. In biology, it has been observed that groups of proteins where in-

teractions occur via a central protein often represent similar biological processes [14]. This phenomenon makes computing 2-clubs, especially those that induce star graphs, particularly interesting [16, 82].

In transportation, hub-and-spoke model is the most popular network architecture used by major airlines [3, 60]. One of the main advantages of this model is that it is optimal in the sense that it ensures a 2-hop reachability while using the minimum possible total number of direct connections. Under this model, most of the flights are routed through several hub airports. This provides passengers a convenient access (via hubs) to numerous destinations that would not be able to support many direct connections, as well as allows to facilitate a wide variety of services, thus attracting more customers.

Another application is in computer and communication networks security [35]. A *bot* is a malicious program carrying out tasks for other programs or users and a *botnet* is a network of *bots*. Botnets are usually controlled by members of organized crime groups, called botmasters, for many different purposes. Almost all computers can host malicious programs that belong to a particular botnet and only a few of them might be immune to becoming a host. Distribution of spam and Distributed Denial-of-Service Attacks (DDoS) are among the malicious tasks performed by botnets. Naturally, the botmaster would like to maximize the effect of attack, damage, to the network and is therefore interested in selecting the densest subnetwork to initiate the attack. Identifying the densest subnetwork would help the botmaster to pick the minimum number of nodes to attack. This strategy leads to the greatest possible damage to the network and, at the same time, minimizes the chance of detection and regulation. Therefore to protect the network and minimize the damage and, at the same time, reduce the cost of taking defensive steps, it is essential to locate cohesive subgraphs and nodes through which such malicious programs can propagate all over

the network very quickly.

In internet research, 2-clubs have been used for clustering web sites to facilitate text mining in hyper-linked documents [74], as well as search and retrieval of topically related information [98]. In wireless networks, small-diameter dominating sets, or the so-called *dominating s-clubs* offer an attractive alternative to usual connected dominating sets as a tool to model virtual backbones used for routing [26, 63]. Since wireless networks are often modeled using geometric graphs known as unit disk and unit ball graphs, studying the distance-based relaxations restricted to such graphs is of special interest. It is well-known that a maximum clique in unit disk graphs can be found in polynomial time [15, 33]. However, the complexity of maximum 2-clique and 2-club problems restricted to unit disk graphs remains open. A 0.5-approximation algorithm for the maximum 2-clique problem in unit-disk graphs that is based on geometric arguments is given in [83].

1.4.2 Biclique

Many of the real world interactions between classes of entities form a bipartite graph in nature [66]. Scientific collaboration networks, where authors and papers are the two partitions, song-listener or movie-recommendation networks that connect users to movies that have been watched are examples from social network analysis. Word-document graphs, where documents can be webpages, emails or dictionary entries and gene-disease relationships are examples of information and biological networks. Therefore bicliques provide a convenient method for modeling and analysis in such networks.

Among many application areas, biologists and genetic researchers have used these models in phylogenetic dataset mining. The evolutionary relationships between variety of species or entities that are considered to have a common ancestor is represented

using phylogenetic trees. Accuracy of these reconstructed phylogenetic trees is crucial and phylogeneticists extract large multigene data sets from gene sequence databases to determine whether there are at least k genes sampled from a pre-determined number of species. Therefore the problem reduces to finding bicliques with minimum number of vertices in each bipartition representing genes and species. Enumerating all maximal bicliques that satisfy the size constraints leads to discovery of complete and accurate phylogenetic trees [89].

DNA microarray data, often presented as a two-dimensional matrix, is used to study the interaction between genes and conditions in biology. Rows of the matrix correspond to genes or clones and columns correspond to test conditions that can be samples, diseases or treatments, etc. Entries then represent the expression level of a given gene i measured under a certain condition j. It is of interest to understand the relationship between subsets of genes and subsets of conditions as it directly captures the essence of biological processes at the cell and molecular level. This includes finding a subset of conditions with same effect on the expression level of a subset of genes or finding a group of genes that are up(down)-regulated in a systematic way under a subset of conditions. Biclustering techniques have become popular in data analysis and bioinformatics due to the fact that simple clustering methods can only provide one dimensional analysis and cannot capture the interrelationship between two or more entities. DNA microarray data can be modeled using a bipartite graph where genes and conditions are represented as vertices in the two partitions, and if there is significant increase or decrease in the expression level of a gene with respect to a specific condition, the two vertices are connected with an edge. In this graph a bicluster corresponds to a biclique and search for a maximum edge biclique is equivalent to finding a maximum bicluster [32].

Bicliques have been widely used in web community discovery, manufacturing

planning, language theory and formal concept analysis [34, 64, 67, 70]. Recently, bicliques were used to model and solve a marketing problem named product bundling. This marketing strategy is mainly considered for consumer retail products and is reportedly helpful in lowering the supply chain costs by directly delivering the goods from producer to the retail store. The concept is to sell two or more different products in a single package. Therefore given the demand, the objective is to select an optimal set of k product bundles that maximizes the total number of products sold [2].

1.5 Research objectives and concluding remarks

This chapter presented an up-to-date survey of the literature on some of the cluster-detection models and the corresponding optimization problems. Applications in large-scale social, information and telecommunication, biological, and transportation networks, where easy accessibility between the system's entities is of utmost importance, stimulated a significant activity in studying distance-based clique relaxation models, s-clique and s-club. Due to its stronger cohesiveness properties and non-hereditary nature, which results in interesting research challenges, the maximum s-club problem has attracted much more attention. The lack of results for the maximum s-clique problem can also be explained by the fact that this problem is equivalent to the maximum clique problem in the corresponding power-s graph, and the maximum clique problem has been studied extensively in the last several decades. While solving the maximum s-clique problem by reducing it to the maximum clique problem is, perhaps, most natural and straightforward approach, it is not clear whether it is most effective. The power-s graph G^s typically has a much higher edge density than the original graph G, and the maximum clique problem is known to be particularly difficult to solve on dense graphs in practice. Moreover, clique is W[1]-hard, and, given that s-club is fixed-parameter tractable for $s > 1$,

reducing s-clique to clique does not appear to be appealing from the parameterized complexity viewpoint. Therefore, investigating the maximum s-clique problem from alternative perspectives may be of interest. In this research we are specifically interested to answer the following questions on the s-club model:

- Given the non-heredity property of the s-clubs, develop an scalable algorithm that can provide a good solution in a reasonable computational time for the large-scale instances of the MsCP problem regardless of the density of the input graph.

- Develop a hybrid solution method to investigate the effect of high quality starting solution on the performance of the exact algorithms for MsCP.

Algorithms provided for solving variations of the maximum biclique problem are mainly heuristic and do not guarantee optimality of the solution. Also considering the size of the underlying network in the application areas, scalability of the solution method is important. Therefore this research attempts to find the answer to the following questions on the biclique community detection:

- Characterizing the structure of the optimal solution for the variations of the maximum biclique problem in uniform random graphs and providing an analytical comparision between the size of these structures in large-scale networks

- Design and implementation of algorithms for solving large-scale instances of the maximum biclique problem in general graphs

Answers to these questions will provide better insights in the analysis of biological networks and their interactions. Moreover it facilitates the solution to clustering and biclustering problems arising in different applications.

2. ALGORITHMS FOR THE MAXIMUM s-CLUB PROBLEM[1]

2.1 Introduction

Let $G = (V, E)$ be a simple undirected graph, where $V = \{1, \ldots, n\}$ is the vertex set, and $E \subseteq \{(i, j) : i, j \in V\}$ is the edge set. Unlike cliques, s-clubs do not posses heredity, meaning that a subset of a s-club may not be a s-club [76]. In other words, s-clubs are not closed under set inclusion. This property of s-clubs exacerbates the development of exact and heuristic algorithms for MsCP and leads to intractability of testing the maximality of s-clubs [80], which is in contrast to trivial verifiability of maximal cliques.

In this chapter, we propose a method to test the maximality of a given s-club. In addition, we develop a new construction heuristic based on s-neighborhood of a given initial solution, which proves to be very effective for sparse graphs. Several combinatorial neighborhood structures are introduced for MsCP and a VNS metaheuristic is proposed that utilizes the suggested neighborhood structures. The developed VNS approach is then incorporated into a branch-and-bound framework proposed by [80] to obtain a hybrid exact algorithm for MsCP.

2.2 Checking maximality of a s-club

Due to the NP-hardness of checking whether an s-club is maximal, developing sufficient conditions for an s-club to be maximal is of importance for designing effective algorithmic procedures for the maximum s-club problem.

Let $G = (V, E)$ be a simple undirected graph. Let $N(i)$ and $N[i]$ denote the *neighborhood* and *closed neighborhood* of a vertex i respectively as defined in chapter 1.

[1]Parts of this chapter are reprinted with permission from S. Shahinpour and S. Butenko: Algorithms for the maximum k-club problem in graphs. Journal of Combinatorial Optimization, 2012, DOI:10.1007/s10878-012-9473-z ©Springer.

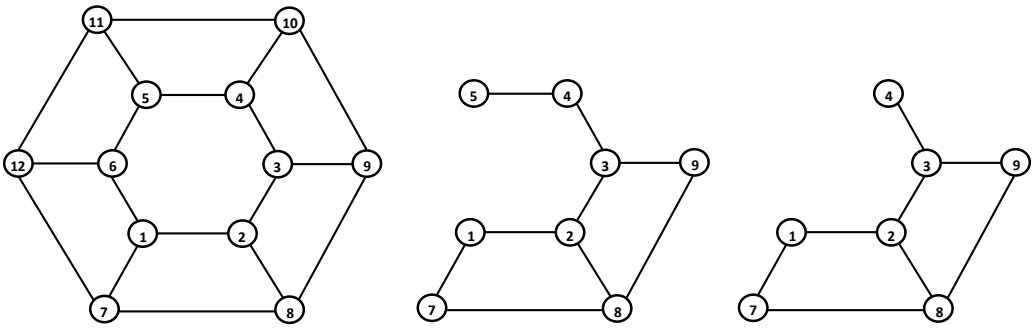

Figure 2.1: Graph G (left) and the induced subgraphs $G[N^3(G, S, 1)]$ (middle) and $G[N^3(G, S, 2)]$ (right) for $S = \{1, 2, 3, 9\}$.

For a given positive integer s, the *s-neighborhood* $N^s_G(i)$ of a given vertex $i \in V$ is the set of vertices having a distance of at most s from i in G, $N^s_G(i) = \{j : d_G(i, j) \leq s\}$. Note that since $d_G(i, i) = 0$, we have $i \in N^s_G(i)$ for any $s \geq 1$ and $N^1_G(i) = N[i]$. Similarly, we can define the s-neighborhood $N^s_G(S)$ of a given subset of vertices S as follows: $N^s_G(S) = \bigcap_{i \in S} N^s_G(i)$. For a s-club S and a positive integer p, denote by $N^s(G, S, p)$ the following recursively defined set:

$$N^s(G, S, p) = \begin{cases} N^s_G(S), & \text{if } p = 1, \\ N^s(G[N^s(G, S, p-1)], S, 1), & \text{if } p \geq 2. \end{cases}$$

Then, according to the above definition, for $p \geq 1$, $N^s(G, S, p)$ is the s-neighborhood of S in $G[N^s(G, S, p-1)]$. Figure 2.1 illustrates the definition for s=3 on a 12-vertex graph G with $S = \{1, 2, 3, 9\}$. According to the definition, $N^3(G, S, 1) = \{1, 2, 3, 4, 5, 7, 8, 9\}$ and $N^3(G, S, 2) = N^3(G[N^3(G, S, 1)], S, 1) = \{1, 2, 3, 4, 7, 8, 9\}$. Moreover, for any $p \geq 2$ the 3-neighborhood of S is the same in the induced subgraph $G[N^3(G, S, 2)]$, thus, $N^3(G, S, p) = \{1, 2, 3, 4, 7, 8, 9\}$.

The following lemma describes some basic properties of $N^s(G, S, p)$ and explains why this structure may be useful for designing algorithms for computing a maximal s-club in a graph.

Lemma 1. *The following properties hold.*

1. *If $S \subseteq S^* \subseteq V$ then we have $N^s(G[S], S, p) \subseteq N^s(G[S^*], S, p) \subseteq N^s(G, S, p)$ for any $p \geq 1$.*

2. *If S is a s-club then we have $S = N^s(G[S], S, p) \subseteq N^s(G, S, p+1) \subseteq N^s(G, S, p)$ for any $p \geq 1$.*

3. *Let S be a s-club that is not maximal. Then for any maximal s-club S^* containing S we have: $S^* \subseteq N^s(G, S, p)$ for any $p \geq 1$.*

Proof. The first property follows from the observation that for any $i, j \in S$, if $S \subseteq S^* \subseteq V$ then we have $d_{G[S]}(i, j) \geq d_{G[S^*]}(i, j) \geq d_G(i, j)$. The second property can be easily established using induction and definition of $N^s(G, S, p)$. To prove the third property, note that since S^* is a s-club containing S, we have: $S^* = N^s(G[S^*], S, p) \subseteq N^s(G, S, p)$ for any $p \geq 1$. □

Based on the third property in the above lemma, local search algorithms for the MsCP can concentrate on searching the set $N^s(G, S, p)$, which may significantly reduce the search space, especially in low-density graphs. The following property provides a sufficient condition for maximality of a s-club that will later be used in the local search phase of the proposed VNS algorithm.

Theorem 4. *Given a s-club S, if there exists a positive integer p such that*

$$N^s(G, S, p) = S$$

then S is a maximal s-club.

Proof. The proof follows from Lemma 1. Indeed, if we assume that $N^s(G, S, p) = S$ holds but S is not a maximal s-club, then there exists a maximal s-club S^* containing

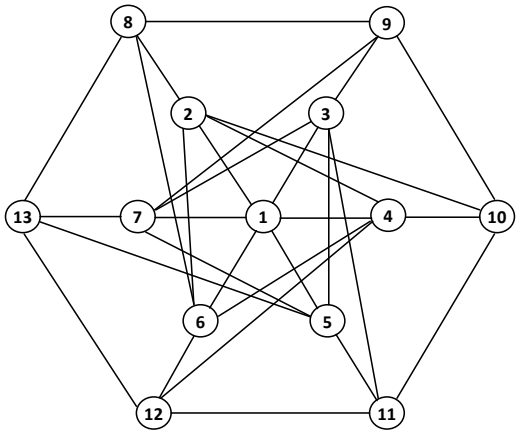

Figure 2.2: A graph $G = (V, E)$ with a maximal 2-club $S = \{1, 2, 3, 4, 5, 6, 7\}$ such that $N^2(G, S, p) = V$ for any $p \geq 1$.

S, and, according to the third property of Lemma 1, $S \subset S^* \subseteq N^s(G, S, p)$ for any $p \geq 1$. This contradicts to the condition that there exists a number p for which $N^s(G, S, p) = S$ and the proof is complete. □

Note that Theorem 4 only provides a sufficient condition for s-club maximality and the reverse statement is not necessarily true. In other words, maximality of a s-club S does not imply the existence of p such that $N^s(G, S, p) = S$. For example, consider the graph $G = (V, E)$ shown in Fig. 2.2. In this graph, the subset of vertices $S = \{1, 2, 3, 4, 5, 6, 7\}$ is a maximal 2-club, however, for any $p > 1$, $N^2(G, S, p) = N^2(G, S, 1) = V \supset S$. Note that V is not a 2-club since, for example, the distance between vertices 8 and 11 is 3.

2.2.1 Maximality test

Recall that not only computing a maximum s-club is NP-hard, but even finding a *maximal* s-club is a computationally intractable problem. Therefore, guaranteeing a maximal s-club in the output is a reasonably ambitious goal for a metaheuristic approach. To verify whether a given s-club is maximal, we will use the result from Theorem 4 as follows. Given a s-club S, we compute $N^s(G, S, p)$ starting with p=1

and keep increasing p until we reach the point where $N^s(G, S, p) = N^s(G, S, p-1)$. If for such value of p we have $N^s(G, S, p) = S$, then the subgraph induced by S is a maximal s-club and cannot be improved to a larger s-club containing S.

2.3 Description of the VNS method

2.3.1 Background and the proposed method

A general combinatorial optimization problem can be defined by the set $\mathcal{S}$ of its feasible solutions, an objective function $f(s)$ used to compute the value of each feasible solution $s \in \mathcal{S}$, and a minimization or maximization objective. Due to the computational intractability of many combinatorial optimization problems, (meta)heuristic methods are often used in practice, most of which are based on some type of local search. Local search methods typically rely on a certain *neighborhood structure* $\mathcal{N}$, which, given a feasible solution $s \in \mathcal{S}$ defines a set $\mathcal{N}(s) \subseteq \mathcal{S}$ of its *neighbors*. Such combinatorial neighborhood structures arising in design of heuristic approaches should not be confused with vertex neighborhoods in graphs discussed above. The difference in these concepts is sufficient for the reader to easily figure out which type of neighborhood is meant each time the term is used in the remainder of this chapter.

Variable neighborhood search (VNS) is a metaheuristic framework developed by [75] (see also [54]) for solving hard combinatorial optimization problems. VNS explores the solution space through a systematic change of neighborhood structures. The procedure is based on the fact that a local optimum for one neighborhood structure is not necessarily a local optimum for another neighborhood structure. Hence, to avoid being trapped in a poor-quality local optimum, the VNS explores multiple neighborhood structures in a systematic fashion. Different variations of VNS have been successfully applied to diverse combinatorial optimization problems. For example, [13] applied VNS to the graph coloring problem and [55] used VNS to

solve the maximum clique problem. Other problems successfully approached using VNS include location [45], routing [65], sequencing [47] and container loading [81] problems.

Let us denote by $\mathcal{N}_k, k = 1, ..., k_{max}$, a finite set of predefined neighborhood structures, where $\mathcal{N}_k(s)$ is the set of solutions in the k^{th} neighborhood of a given solution s. Note that the standard local search heuristics use only one neighborhood structure, i.e., $k_{max} = 1$. The basic VNS consists of the following steps.

1. *Initialization.* Find an initial solution s; choose a stopping condition; set $k = 1$.
2. *Main loop.* Repeat the following sequence until the stopping condition is met.
   - 2.a *Shaking.* Generate a point s' at random from the k^{th} neighborhood of s ($s' \in \mathcal{N}_k(s)$).
   - 2.b *Local search.* Apply some local search method starting with s' to obtain a local optimum s''.
   - 2.c *Move or not.* If s'' is better than the incumbent s, set $s = s''$, and continue the search with $\mathcal{N}_1$ (i.e., set $k = 1$); otherwise, set $k = k \mod k_{max} + 1$.

The stopping condition for VNS may be, for example, maximum CPU time allowed, maximum number of iterations, or maximum number of iterations between two improvements. Observe that a solution s' generated in step 2.a is obtained by randomly choosing it in the k^{th} neighborhood. This helps to avoid cycling that may occur if any deterministic rule was used. Since a local optimum with respect to some neighborhood is not necessarily a local optimum with respect to another, change of neighborhoods can be performed during the local search phase. This local search is then called *variable neighborhood descent (VND).* It is also possible to use a simple

descent method or a more powerful technique such as *tabu search* or *simulated annealing* [50] within this framework. The above basic VNS can be viewed as a descent algorithm since we update incumbent only if the local optimum s'' obtained in step 2.b is better than s. It is also possible to derive other variants of the VNS without much additional effort. For example, it can be transformed into a descent-ascent method if in step 2.c a solution s'' could be accepted to substitute for the incumbent s with some probability even if it is worse than the incumbent. In step 2.a, it is also possible to choose the best solution s' as a result of applying several neighborhood structures to s selected at random. In some applications, the local search step is dropped to save time, resulting in the so-called *reduced VNS*. In another variation called *variable neighborhood decomposition search (VNDS)*, VNS is applied to a partial set of variables at each iteration.

To solve the M*s*CP, a variation of VND is adopted as the *local search* procedure of choice in step 2.b of the VNS framework. The proposed method is summarized in Alg. 1, which also contains pointers to specific sections of the chapter providing more detail on the particular steps. Given the set of neighborhood structures and initial solution X, the local search proceeds with generating a neighboring solution X_i using the k^{th} neighborhood structure starting from $k = 1$. In the next step, maximality of this neighbor solution is checked using the method described in Sec. 2.2.1. If this solution is not maximal, *Local-Improvement* procedure is executed to improve the solution if possible, as described in Sec. 2.3.3. If the solution thus obtained is better than the initial solution X, the initial solution is updated and the local search starts over again with $k = 1$. On the other hand, if no improvement is possible after complete search of the k^{th} neighborhood, the algorithm explores the next neighborhood structure by setting $k = k + 1$. The above procedure continues until no improvement in the solution is observed in one complete iteration of VND.

Algorithm 1 Steps of the basic VND.

Initialization: Select the set of neighborhood structures $\mathcal{N}_k^{'}, k = 1, ..., k_{max}^{'}$ and let X be the initial solution
repeat
    Set $k \leftarrow 1$;
    repeat
        procedure Neighborhood-Exploration(k, X)
        Set $X^{'} \leftarrow X$;
        for $i \leftarrow 1$ to $|\mathcal{N}_k^{'}(X)|$ **do**
            generate a neighbor X_i using k^{th} neighborhood structure; ▷ see Sec. 2.3.4
            Maximality-Test(X_i); ▷ see Sec. 2.2.1
            if X_i is not maximal **then**
                Local-Improvement(X_i); ▷ see Sec. 2.3.3
            end if
            if $|X^{'}| < |X_i|$ **then**
                Set $X^{'} \leftarrow X_i$;
            end if
        end for
        end procedure
        if $|X| < |X^{'}|$ **then**
            Set $X \leftarrow X^{'}$ and $k \leftarrow 1$;
        else
            Set $k \leftarrow k + 1$;
        end if
    until $k = k_{max}^{'}$
until no improvement is obtained
return the best solution X found;

2.3.2 Initial feasible solution

To obtain an initial solution, we employ three algorithms in the construction phase. The first two are called CONSTELLATION and DROP and were developed by [23]. The third one, which we call EXPAND, is based on vertex expansion. The best initial solution provides a lower bound, which we will refer to as LB-1. This solution is selected to start the VNS iterations. The first step of CONSTELLATION procedure consists of identifying the vertex of highest degree, including this vertex together with its neighbors in the s-club S being constructed. The next step consists of $s - 2$ recursive iterations, in each of which a vertex from S having the largest number of neighbors in $V \setminus S$ is selected, and all its neighbors are added to the

Algorithm 2 EXPAND algorithm for construction phase.

```
1: Initialization: Sort vertices based on non-increasing order of their degree in G; C* = ∅;
2: for i ← 1 to |V| do
3:       C_i ← N_G^s(N[i]);
4:       if |C_i| > |C*| then
5:             C_i ←DROP(C_i);
6:             if |C_i| > |C*| then
7:                   C* ← C_i;
8:             end if
9:       end if
10: end for
11: return C*;
```

s-club S. In DROP construction, we start with the whole graph G and at each iteration a vertex i with the highest infeasibility is deleted and the graph is updated. The infeasibility of a vertex i is defined as the number of vertices in G whose distance to i is at least $s+1$. If there is a tie, a vertex with minimum degree is then selected for elimination, with further ties broken at random. The procedure continues until no infeasible vertex can be found.

The proposed EXPAND construction proceeds by taking the closed neighborhood of each vertex, computing its s-neighborhood and applying the DROP procedure to ensure feasibility. Note that the closed neighborhood of a vertex provides a 2-club, which is used to obtain the s-neighborhood in our algorithm for $s \geq 2$. The steps of EXPAND are outlined in Alg. 2. Note that the vertices are sorted based on a non-increasing order of their degrees in the initialization step. This is done to increase the chance of obtaining larger s-clubs in the initial steps of EXPAND. At each iteration i, the number of vertices in $C_i = N_G^s(N[i])$ is compared against the cardinality of the best s-club C^* found so far, and DROP is carried out only if $|N_G^s(N[i])| > |C^*|$.

2.3.3 Local improvement procedure

Before we describe the proposed neighborhood structures, we first discuss the basic local improvement procedures that are natural to use in the neighborhood

definitions. We refer to these procedures as *1-add move* and *2-add move*, respectively, as they attempt to add one or two new vertices to the existing solution. The 1-add and 2-add moves applied in sequence constitute the local improvement procedure described next.

Let $\mathcal{C}_s(G)$ denote the set of all s-clubs in G and let $S \in \mathcal{C}_s(G)$ be a given s-club. Denote by A_1 a subset of vertices outside of S, each of which forms a s-club together with S, i.e., $A_1 = \{u \in N_G^s(S) \setminus S : S \cup \{u\} \in \mathcal{C}_s(G)\}$. Set A_1 contains the vertices that can be used to perform a 1-add move. Furthermore, let A_2 be a subset of edges whose endpoints are not in $S \cup A_1$, but form a s-club together with S, i.e., $A_2 = \{(u, v) \in E : u, v \in N_G^s(S) \setminus (S \cup A_1)$ and $S \cup \{u, v\} \in \mathcal{C}_s(G)\}$. A_2 consists of pairs of vertices that can be used for a 2-add move. The definition of A_2 is motivated by the lack of the heredity property in s-clubs and is illustrated for s=2 in Fig. 2.3, where $S = \{1, 2, 3\}$ is a 2-club and $(4, 5) \in A_2$. In this case, both vertices 4 and 5 can be added to S simultaneously and we obtain a 2-club, while adding only one of these vertices will result in an infeasible solution. Similarly, it may be necessary to

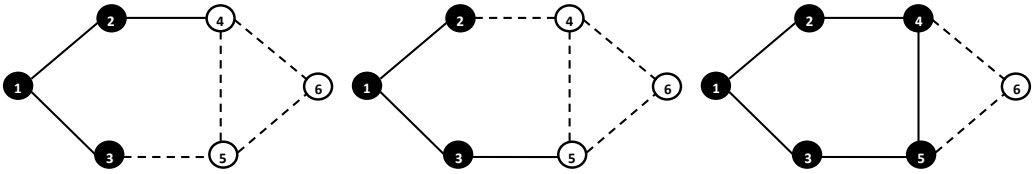

Figure 2.3: Graphical representation of 2-add move for 2-club.

add $p \geq 3$ vertices at a time to maintain the feasibility of the solution, while adding any other subset of these vertices would result in an infeasible solution. Thus, we can define a *p-add move* to address this issue. However, for $p \geq 3$ the p-add move becomes inefficient due to the combinatorial nature of computing the possible moves.

In our implementation of the 1-add procedure, the vertices in A_1 are sorted based on non-increasing order of their degree in S since adding vertices in this order is

likely to result in larger s-clubs early on. After the first vertex from A_1 is added, the sets S, $N_G^s(S)$ and A_1 are updated and the procedure continues recursively until $A_1 = \emptyset$. While computing A_1 at each iteration of the 1-add procedure, for every vertex $v_i \in N_G^s(S) \setminus S$ we maintain a list $T(v_i)$ of all vertices in S whose distance from v_i is at least $s+1$. This list is later used for 2-add move to facilitate computing the set A_2 of candidate vertex pairs. Then two new vertices corresponding to an edge in A_2 are introduced to the solution at the same time without violating the feasibility, the set A_2 is updated, and the 2-add procedure continues recursively until A_2 is empty.

The result of the local improvement procedure, which consists in applying the 2-add moves on top of the 1-add procedure, is a s-club S that cannot be further increased by adding any one or two vertices at a time. Next we discuss the proposed neighborhood structures that can be divided into four different groups. Several other variations of these neighborhood structures were considered in preliminary numerical experiments and are not discussed here due to their inferior performance.

2.3.4 Neighborhood structures

This subsection discusses the proposed neighborhood structures. They can be viewed as variations of the *exchange neighborhoods* commonly used in combinatorial optimization, which, essentially, modify the current solution by removing some of its elements (e.g., vertices) and adding other elements that are not included in the current solution.

$\mathcal{N}_1$: DROP-based neighborhood

Given a s-club S, we compute $N_G^s(S)$ and consider the graph induced by this s-neighborhood. Then sequentially remove the most infeasible vertex, where the infeasibility is measured as described in Sec. 2.3.2, until a new feasible solution is

obtained.

$\mathcal{N}_2$: Adding a new vertex to the current solution

For a new vertex i to be added to a given s-club S, i must belong to the s-neighborhood $N_G^s(S)$ of S. Therefore, one of the vertices in $N_G^s(S) \setminus S$, say v, is selected at random and added to S. There are two possibilities at this point. If the subgraph induced by $S \cup \{v\}$ is still a s-club, we obtain a neighbor of S that we denote by $\mathcal{N}_2^v(S)$, otherwise, if $S \cup \{v\}$ is not a s-club, it needs to be refined in order to obtain a feasible s-club, i.e., some vertices have to be removed in order for v to be added. While computing the feasibility of the subgraph induced by $S \cup \{v\}$, we keep track of the vertices whose distance from v is at least $s + 1$. These are the vertices that need to be removed. We start by removing a vertex that has the minimum number of neighbors in S first. Next, the pairwise distances in $G[S \cup \{v\}]$ are computed, and, like in the DROP procedure, a vertex that is different from v and has the maximum infeasibility is removed. The procedure outputs a feasible solution containing v. We will denote this feasible solution by $\mathcal{N}_2^v(S)$. Then, given a s-club S, its neighborhood $\mathcal{N}_2$ can be defined as $\mathcal{N}_2(S) = \{\mathcal{N}_2^v(S) : v \in N_G^s(S) \setminus S\}$.

Note that in the $\mathcal{N}_2$ neighborhood we are essentially looking for a s-club in $S \cup \{v\}$ with the additional constraint that the s-club must contain v. Based on our computational experiments, this neighborhood structure changes the size of a given s-club drastically in sparse graphs, while for dense graphs this change is usually moderate or small. The idea behind using this neighborhood is that introducing a new vertex to the current solution helps to diversify the search.

$\mathcal{N}_3$: Removing a vertex from the solution

In this neighborhood structure, vertices in S are sorted in a certain order $v_1, v_2, \ldots, v_{n_S}$, where n_S be the number of vertices in S. We build n_S neighbors of the solution S as

follows. To form the i^{th} neighbor, $i = 1, \ldots, n_S$, vertex v_i is temporarily eliminated from the graph until the corresponding neighbor is constructed. It is obvious that we need to check the feasibility of the remaining set. In case it is infeasible, a feasible solution is extracted from the remaining set using DROP. Given the resulting feasible solution, we apply the local improvement procedure described in Sec. 2.3.3 to check if any new vertices can be added to improve the solution.

We considered the following five different strategies for ordering vertices in S:

1. Non-decreasing order of vertex degrees in the subgraph induced by S;
2. Non-decreasing order of vertex degrees in the subgraph induced by $V \setminus S$;
3. Non-decreasing order of the cardinality of s-neighborhoods $N_G^s(v), v \in S$;
4. Non-decreasing order of cardinality of the sets

$$\{u \in N_G^s(S) \setminus S : d_{G[S \cup \{v\}]}(u, v) \leq s\}, v \in S,$$

   which characterize the reachability of v from the vertices in the s-neighborhood of S.
5. Random ordering.

$\mathcal{N}_4$: Removing a group of vertices from the solution

This neighborhood structure is similar to the previous one, but instead of removing only one vertex, a group of vertices is removed to form a neighbor of S. In this case, the number of neighbors of a given s-club S may not be the same as the cardinality of S. We sort vertices in S in either non-increasing order of their degrees in the subgraph induced by $V \setminus S$ or non-decreasing order $v_1, v_2, \ldots, v_{n_S}$ of their

degrees in the subgraph induced by S. For each $i = 1, \ldots n_S$ we remove a subset S_i of vertices in S, where S_i is determined using one of the following rules:

1. $S_i = N[v_i] \cap S$;
2. $S_i = \{v \in S : |N(v) \cap S| = |N(v_i) \cap S|\}$;
3. $S_i = \{v_i, v_j\} \subset S$, where $(v_j, v_i) \notin E$ and $j > i$;
4. $S_i = \{v_i, v_j\} \subset S$, where $j > i$.

2.4 A hybrid exact algorithm

Inspired by a promising performance of the proposed VNS method, we develop an exact algorithm for the MsCP incorporating the VNS into a combinatorial branch-and-bound algorithm (B&B) proposed by [80]. Recall from section 2.3.2 that LB-1 refers to the lower bound used to initialize the VNS algorithm. The B&B algorithm employs two lower bounding schemes. The first one, which will be denoted by LB-2 considers the best solution among DROP and CONSTELLATION heuristics, both originally proposed by [23], to identify an initial feasible solution and initializes the incumbent. The second scheme considers the best solution among DROP and CONSTELLATION heuristics followed by a bounded enumeration to possibly grow the cardinality of the starting solution. We will denote by LB-3 the lower bound corresponding to this initial solution.

Two methods are used to derive an upper bound for the B&B algorithm. The first one is based on the fact that the chromatic number $\chi(G^s)$ of the s^{th} power graph G^s provides an upper bound on the s-club number of G. Given a graph $G = (V, E)$, the s^{th} *power graph* $G^s = (V, E^s)$ is a graph defined by the set of vertices V and the set

of edges $E^s = \{(i,j) : i,j \in V; d_G(i,j) \leq s\}$. The *chromatic number* $\chi(G)$ of G is the minimum number of colors required to color the vertices of G *properly*, i.e., so that no two neighbors are assigned the same color. To obtain this upper bound, which will be denoted by UB-1, a combination of greedy heuristic and DSATUR heuristic, proposed by [25], is used. The second method computes the maximum s-clique to serve as an upper bound for the s-club number of a given graph G. To obtain this upper bound the algorithm proposed by [79] is employed to find the maximum clique on s^{th} power graph G^s. We will denote by UB-2 the upper bound corresponding to this method. For branching, a vertex dichotomy is used, where a vertex is selected and fixed to be included or deleted from the solution. To traverse the search tree, best bound search (BBS) strategy has been considered. For more details on the B&B algorithm please refer to [80].

The idea behind the proposed hybrid algorithm is to use the solution obtained from VNS as a starting solution for B&B algorithm. This is due to the fact that the quality of the solution obtained from the VNS algorithm is in most cases higher than the lower bounding techniques used to initiate the B&B algorithm. Therefore, the hybrid algorithm employs VNS to obtain a good initial solution, denoted by LB-4, that is considered as incumbent to start the branch and bound search.

2.5 Results of computational experiments

The developed algorithms were implemented in C++ and all numerical experiments were conducted on Dell workstation with Intel 3.00 GHz quad-core processor and 8.00 GB RAM. To diversify the experiments and fully explore the potential of the proposed algorithms, two different set of instances were considered. The first set includes some of the test instances from the tenth DIMACS implementation challenge [39] in order to test the algorithms on some large-scale real networks. Instances

in the second set were generated randomly using the algorithm introduced by [49], which is a generalization of the classical uniform random graph generator and is controlled by two density parameters a and b ($0 \leq a \leq b \leq 1$). The expected edge density of instances produced using this method is equal to $(a+b)/2$, and the vertex degree variance increases with the increase in $b-a$. For this set of instances the experiments for MsCP with s=2 and 3 were performed on graphs with n=50, 100, 150 and 200 vertices with edge densities d=0.0125, 0.025, 0.05, 0.1, 0.15, 0.2, and 0.25, respectively. Among these, according to [23, 24], the edge densities 0.15 and 0.025 were observed to produce the hardest instances for their respective s values. For each size and density considered, we generated 10 sample graphs with $a = b = d$ (we refer to this case as having the minimum vertex degree variance (VDV)) and 10 samples with $a = 0, b = 2d$ (referred to as the maximum VDV cases). The first ten instances in each category are those with the minimum VDV and the second ten are instances with the maximum VDV. For the VND algorithm used within the VNS framework, a set of five neighborhood structures among those proposed in Sec. 2.3.4 have been used, i.e., $k'_{max} = 5$. The choice of these neighborhoods is based on their effectiveness measured by examining different sets of neighborhoods in the initial experiments and observing the improvement in the objective function after a specific amount of time. These neighborhoods are the first three structures defined for $\mathcal{N}_3$ and the last two structures defined for $\mathcal{N}_4$. Note that the neighborhoods not used in the local search step are not necessarily ineffective in all cases attempted in the experiments. Although some of them had reasonably good performance in terms of the solution quality, the computational time required for exploring those neighborhoods was rather high. In particular a truncated version of $\mathcal{N}_4$ neighborhood has been used for instances with higher densities in order to reduce the search space. For the VNS framework, all of the proposed neighborhood structures, except for four

of them, have been used. These neighborhoods were excluded due to their inferior performance with respect to either solution quality or computational time requirements, and include the last two structures defined for $\mathcal{N}_3$ and the first two structures for $\mathcal{N}_4$. The running time limit for all three algorithms was set to 3600 seconds and results are compared. For the VNS algorithm the time limit was enforced by keeping track of the elapsed time after each call to the local search algorithm, VND, which for some instances exceeded the time limit mainly because of the time needed to finish the neighborhood search using all corresponding neighborhood structures. For the B&B algorithm in [80], the elapsed time is monitored after processing each B&B tree node to enforce the one-hour time limit. For some of the larger instances from DIMACS the time required to process a tree node was significant and as a result the CPU time exceeded the time limit by a large margin. For the hybrid algorithm, the time limit considered for the lower bounding technique was set to 1200 seconds, although in many cases the time required for the VNS was only a fraction of this amount. Only those DIMACS instances that were not solved to optimality and for which VNS obtained a better solution than the B&B algorithm were attempted using the hybrid method. On the remaining DIMACS instances, the hybrid algorithm was not able to obtain any improvement over the lower bound obtained using VNS or to prove optimality of of the best s-clubs found. Likewise, for the randomly generated instances the hybrid algorithm was used to solve some of the harder instances in the sense that the B&B algorithm was not able to solve them to optimality in one-hour time limit or required a rather high computational time. Therefore, for s=2 we considered moderate size instances of density 0.15 to solve using the hybrid algorithm, and for s=3 instances with density 0.025 and 0.05 were attempted.

Tables 2.1-2.2 show the best objective value, optimality gap and the running time for DIMACS instances solved by VNS and B&B algorithms. The optimality

gap for B&B is computed as 100×*(upper bound-best solution size)/(upper bound)*. For s=2, most of the small to medium size instances were solved to optimality using both methods in the given time limit. In two of the larger instances the B&B algorithm could not reach any feasible solution mainly because the time required for the construction procedure was greater than the time limit and the process was terminated. In these two cases even extending the time limit beyond three hours was not enough to obtain a feasible solution using B&B. In seven other cases the optimality of the solution could not be verified. Among these, in one instance the size of the solution found by VNS was larger. Although the running time grows for both methods with increase in size of the instances, the growth rate is much slower for the VNS algorithm. Figure 2.4 illustrates this by comparing the CPU time for the two methods obtained by removing the time limit restriction in order to solve the large instances to optimality. Note that except for two cases, all other instances shown in the figure were solved before the time limit by VNS. For s=3 both methods solve the small instances and the running time grows with the size and density of the problems. There are six instances for which the B&B returned a feasible solution and then the execution was terminated beyond the time limit. This was a result of significant time needed to process a tree node and as a result no optimality gap could be reported. In four cases the solution obtained by VNS is better, and in three instances the B&B algorithm did not return any solution. The same figure illustrates the CPU time only for instances that were solved by VNS within the time limit and by relaxing this requirement for B&B in order to solve M3CP to optimality. Except for one case, all other DIMACS instances are non-trivial for both s=2 and s=3, and solving M3CP appears to be harder than solving M2CP for both methods.

Tables 2.3-2.10 show the average best objective value, optimality gap and the CPU time obtained from solving MsCP using randomly generated instances, while

detailed numerical results for VNS heuristic, the B&B and the proposed hybrid algorithm are reported in Tables B.1–B.14. The average optimality gap for the VNS algorithm is calculated based on those instances for which the optimal solution was obtained using the B&B algorithm within one hour. Only the first five instances of minimum VDV and the first five instances of maximum VDV are reported. Recall that there are four different possible combinations to set up the B&B algorithm, considering two methods for each of the lower and upper bounding techniques as mentioned before. For each particular density and size, the best result among all four combinations of the B&B has been compared with VNS in the tables in appendix. In these tables, LB* denotes the lower bound (LB-2 or LB-3) associated with the best among the four solutions obtained for the corresponding instance. Here the best solution is the one with the smallest CPU time, if the instance was solved to optimality; otherwise, it is the solution with the best objective obtained within the one-hour limit. For some particular n, s and d values, e.g., the randomly generated instances with higher order and higher density, the MsCP was a trivial problem since the diameter of the instances were less than the value considered for s. These are mainly instances of density 0.15, 0.2 and 0.25 for the M3CP and their detailed numerical results are not included in the appendix.

For s=2 and densities d =0.0125, 0.025 and 0.05 the VNS algorithm finds an optimal solution in all cases. For small size instances the time required by VNS is reasonably close to that of B&B and for instances of moderate size VNS outperforms B&B on average. For instances with density d =0.1, VNS obtains optimal solutions except for instances of size $n = 200$, for which the optimality gap could not be closed when solved by B&B method. Also in all instances tried, except those of size 50, VNS requires considerably smaller amount of computational time. The rate of increase of the CPU time as a function of instance size is also smaller for VNS than for the exact

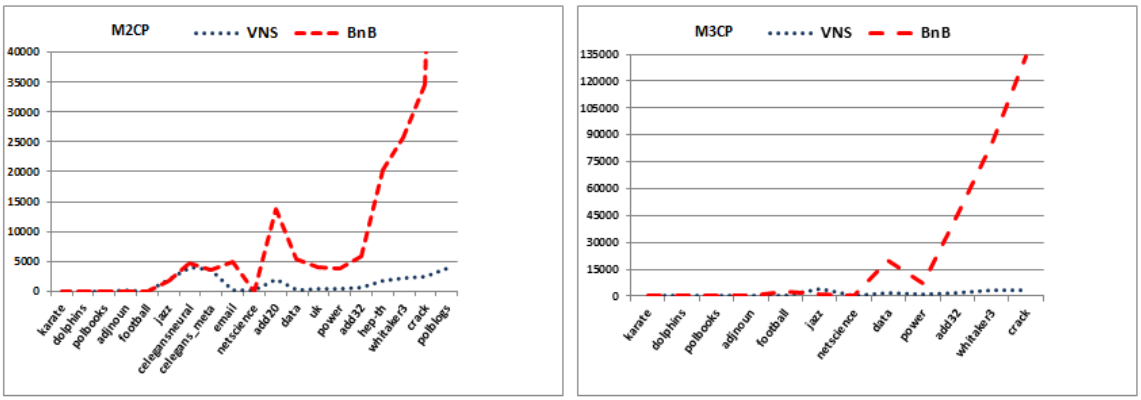

Figure 2.4: Illustration of CPU time required for VNS and B&B to solve M2CP and M3CP on DIMACS instances.

algorithm. For instances of density 0.15, which are reportedly the hardest ones in the literature, the VNS algorithm finds better solutions for moderate-size instances than the B&B algorithm and for higher densities the VNS algorithm obtains optimal or near optimal solutions. For d=0.2, there are some instances on which B&B has a better performance, but in some other cases VNS generates better solutions. In the case of d=0.25, the running time of the exact algorithm is better than that of VNS mainly because of the use of the upper bound based on the maximum clique on s^{th} power graph G^s, which appears to be quite tight and helps reducing the search tree for dense instances. Among instances that have not been solved to optimality, those with the maximum VDV seem to be harder for the VNS algorithm, as the average optimality gap is larger in these cases. Using hybrid algorithm for instances with density d=0.15, the average solution size increases for graphs of order n =100, 150 and 200 compared to the B&B method. This is mostly because of using VNS as a lower bounding technique, which has a better performance than the heuristics used in B&B. For some instances of size $n = 100$, the effect of using a better lower bound appears in reducing the computing time compared to the B&B case. In all other instances of higher order, except for two cases that have been solved to optimality

Table 2.1: Computational results of solving the MsCP using VNS and B&B algorithms for s=2 on DIMACS instances.

Instance	**n**	**m**	**VNS**		**B&B**		
			s-club size	CPU (sec)	s-club size	Gap (%)	CPU (sec)
karate	34	78	18	1.108	18	0	0.484
dolphins	62	159	13	1.544	13	0	14.711
polbooks	105	441	28	12.324	28	0	10.904
adjnoun	112	425	50	70.37	50	0	20.857
football	115	613	16	5.71	16	0	120.884
jazz	198	2742	103	2012.92	103	0	1824.829
celegansneural	297	2148	286	4100	286	0	4589.566
celegans-metabolic	453	2025	238	3603.615	238	0	3598.077
email	1133	5451	72	273.983	72	0	4821.446
polblogs	1490	16715	352	3872.459	352	76.37	4295.725
netscience	1589	2742	35	70.45	35	0	215.576
add20	2395	7462	124	1957.428	124	94.82	4489.353
data	2851	15093	18	260.785	18	21.73	5440.516
uk	4824	6837	**5**	392.106	4	50	3975.644
power	4941	6594	20	480.024	20	0	3731.988
add32	4960	9462	32	567.45	32	0	5694.063
hep-th	8361	15751	51	1825.316	51	99.39	9403.884
whitaker3	9800	28989	9	2320.6	9	35.71	12631.52
crack	10240	30380	10	2452.326	10	99.9	3924.711
PGPgiantcompo	10680	24316	206	5183.705	-	-	>3600
cs4	22499	43858	5	11371.81	-	-	>3600

by all methods, the hybrid algorithm was not able to close the optimality gap within the time limit, although the difference between the size of the lower bound for the two algorithms is significant in some cases. This suggests that the use of a better lower bounding technique speeds up the branch-and-bound process but is not sufficient to close the optimality gap even for moderate size problems.

For s=3, in all instances tried with density d =0.0125 the VNS algorithm obtained the optimal solution in a time competitive to that of the exact B&B algorithm. For instances of density 0.025, VNS exhibited similar performance, except for some

Table 2.2: Computational results of solving the MsCP using VNS and B&B algorithms for s=3 on DIMACS instances.

Instance	**n**	**m**	**VNS**		**B&B**		
			s-club size	CPU (sec)	s-club size	Gap (%)	CPU (sec)
karate	34	78	25	3.572	25	0	1.123
dolphins	62	159	29	5.808	29	0	29.858
polbooks	105	441	53	58.773	53	0	183
adjnoun	112	425	82	505.05	82	0	313.711
football	115	613	58	192.729	58	0	2280.202
jazz	198	2742	174	4223.505	174	0	1053.395
celegansneural	297	2148	297	0	297	0	0
celegans-metabolic	453	2025	371	3657.981	371	18.1	3698.452
email	1133	5451	**215**	3605.378	201	-	>3600
polblogs	1490	16715	352	3608.22	-	-	>3600
netscience	1589	2742	54	110.073	54	0	441.289
add20	2395	7462	671	3659.34	671	-	>3600
data	2851	15093	**32**	1919.02	28	99	3717.729
uk	4824	6837	8	5746.64	8	38.46	5386.57
power	4941	6594	30	1167.6	30	0	7056.33
add32	4960	9462	96	1450.16	99	-	>3600
hep-th	8361	15751	120	3799.29	120	-	>3600
whitaker3	9800	28989	**15**	3563.2	13	-	>3600
crack	10240	30380	**17**	3497.75	15	-	>3600
PGPgiantcompo	10680	24316	273	9301.1	-	-	>3600
cs4	22499	43858	9	7522.83	-	-	>3600

instances of size 200 for which there was a minor optimality gap. The time needed to solve the problem increased slightly as a function of the instance size, but the amount of increase is moderate. This can be attributed to the use of s-neighborhood function for finding the initial solution, as well as during the local search, which results in a significant reduction of the search space, especially in low-density graphs. Another observation is that the average computational time increases with density and reaches its peak for d=0.05 and then declines for higher densities. This can be explained by the fact that the sparse problems are easier to solve and in many cases are composed

Table 2.3: Average output s-club size, average optimality gap and average running time for the VNS and B&B algorithms (d=0.0125).

s	n	VDV	VNS			B&B		
			s-club size	Gap (%)	CPU (sec)	s-club size	Gap (%)	CPU (sec)
2	50	min	3.7	0	0.09	3.7	0	0.07
		max	4.3	0	0.09	4.3	0	0.06
	100	min	6.1	0	0.28	6.1	0	0.29
		max	5.9	0	0.29	5.9	0	0.28
	150	min	6.9	0	0.62	6.9	0	0.74
		max	8.3	0	0.67	8.3	0	0.76
	200	min	8.2	0	1.15	8.2	0	1.14
		max	9.2	0	1.21	9.2	0	1.20
3	50	min	4.6	0	0.05	4.6	0	0.08
		max	4.4	0	0.05	4.4	0	0.06
	100	min	8.4	0	0.34	8.4	0	0.34
		max	7.9	0	0.28	7.9	0	0.31
	150	min	11.7	0	1.19	11.7	0	14.66
		max	12	0	1.25	12	0	1.13
	200	min	13.6	0	3.94	13.6	0	51.29
		max	14.8	0	3.77	14.8	0	175.38

of separate connected components, whereas for higher densities many of the instances are trivial for s=3. For all considered instances of density 0.05, the VNS algorithm performed well. Specifically, for the moderate-size instances in this category, VNS generated better solutions. For all higher densities VNS found optimal solution in most cases.

Using hybrid algorithm, problem instances with density d=0.025 and 0.05 were attempted. In the former case, all instances were solved to optimality requiring less computational time compared to the B&B algorithm. In the latter case, d=0.05, for instances of order $n = 100$ the computation time decreased and for $n = 150$ only two instances were solved, and in all other cases the optimality gap could not be closed. Likewise, for $n = 200$ two of the maximum VDV instances were solved to optimality, and in all other cases the optimality gap could not be closed, although the average

Table 2.4: Average output s-club size, average optimality gap and average running time for the VNS and B&B algorithms (d=0.025).

s	n	VDV	VNS			B&B		
			s-club size	Gap (%)	CPU (sec)	s-club size	Gap (%)	CPU (sec)
2	50	min	4.9	0	0.14	4.9	0	0.07
		max	5.2	0	0.16	5.2	0	0.07
	100	min	8.8	0	0.65	8.8	0	0.34
		max	10.1	0	0.69	10.1	0	0.36
	150	min	11	0	1.4	11	0	1.08
		max	11.8	0	1.52	11.8	0	1.19
	200	min	12.7	0	2.64	12.7	0	2.97
		max	14.3	0	2.94	14.3	0	3.08
3	50	min	6.6	0	0.1	6.6	0	0.08
		max	7.8	0	0.13	7.8	0	0.1
	100	min	14.6	0	1.55	14.6	0	19.2
		max	15.8	0	2.03	15.8	0	22.08
	150	min	20.5	0	7.16	20.5	0	252.03
		max	22.9	0	9.36	22.9	0	277
	200	min	25.3	0.34	31.58	25.4	0	1007.4
		max	31.6	0.87	44.2	31.9	0	1106.97

optimality gap was decreased.

2.6 Conclusion

In this chapter, a sufficient condition for maximality of a given s-club is developed and is used in the design of a variable neighborhood search heuristic to increase the effectiveness of the local search step. For the construction phase, we proposed a new algorithm that is very effective, especially for sparse graphs. Several methods have been proposed to create neighborhood solutions, which is another challenging issue for this problem that can be attributed to its nonhereditary nature. The emphasis has been placed on the simplicity and effectiveness of the neighborhood structures developed. The computational experiments show that the proposed VNS method is competitive with other algorithms that are available for MsCP to this date, especially on instances with modest density and high order that are reported to be the hardest

Table 2.5: Average output s-club size, average optimality gap and average running time for the VNS and B&B algorithms (d=0.05).

s	n	VDV	VNS			B&B		
			s-club size	Gap (%)	CPU (sec)	s-club size	Gap (%)	CPU (sec)
2	50	min	6.7	0	0.3	6.7	0	0.1
		max	7.4	0	0.31	7.4	0	0.1
	100	min	11.6	0	1.65	11.6	0	1.02
		max	12.7	0	1.71	12.7	0	1.05
	150	min	16.8	0	4.62	16.8	0	5.38
		max	19.5	0	6.05	19.5	0	5.66
	200	min	20.7	0	9.89	20.7	0	21.33
		max	23.4	0	12.62	23.4	0	51.33
3	50	min	11.1	0	0.45	11.1	0	2.44
		max	12	0	0.45	12	0	1.76
	100	min	27.7	1.02	19.57	28	0	222.16
		max	30.9	1.27	25.61	31.3	0	235.37
	150	min	**47.6**	-	310.81	41.3	34.75	3608.88
		max	**65**	-	405.26	60.6	17.74	3439.31
	200	min	**97**	-	2053.37	89.7	33.1	3650.25
		max	**126.3**	-	2269.47	118.7	13.93	3520.72

in the literature. Moreover, it can be used to enhance exact branch-and-bound algorithms. From the practical point of view, it would be of interest to develop more neighborhood structures for this problem and apply them in a metaheuristic and hybrid algorithms. Specifically a scale-reduction algorithm would be very helpful in increasing the potential of the existing algorithms to solve the large sparse networks. Another possible direction of future research is to perform a polyhedral study of this problem and develop facets and valid inequalities that can be applied in a branch-and-cut algorithm to solve MsCP.

Table 2.6: Average output s-club size, average optimality gap and average running time for the VNS and B&B algorithms (d=0.1).

s	n	VDV	VNS			B&B		
			s-club size	Gap (%)	CPU (sec)	s-club size	Gap (%)	CPU (sec)
2	50	min	11	0	2.93	11	0	1.36
		max	11.6	0	3.6	11.6	0	2.48
	100	min	18.9	0	13.95	18.9	0	127.57
		max	22	0.43	20.16	22.1	0	151.85
	150	min	26.8	0	50.16	26.8	0	2252.09
		max	31.7	-	66.63	31.7	2.03	2297.13
	200	min	33.6	-	155.45	33.6	41.96	3614.62
		max	39.1	-	212.75	39.1	33.7	3616.16
3	50	min	30	0.78	13.9	30.2	0	15.47
		max	32	0	9.8	32	0	9.91
	100	min	94	0	16.59	94	0	78.55
		max	93	0.21	22.03	93.2	0	58.06
	150	min	149.9	0	8.01	149.9	0	4.76
		max	148.2	0	74.89	148.2	0	43.42
	200	min	200	0	0	200	0	0
		max	199.7	0	58.57	199.7	0	37.06

Table 2.7: Average output s-club size, average optimality gap and average running time for the VNS and B&B algorithms (d=0.15).

s	n	VDV	VNS			B&B		
			s-club size	Gap (%)	CPU (sec)	s-club size	Gap (%)	CPU (sec)
2	50	min	15.3	1.13	23.59	15.5	0	11.31
		max	16	2.00	22.44	16.3	0	11.07
	100	min	**28.5**	-	214.3	28.2	6.27	2710.33
		max	38.9	4.55	501.17	40.7	0	1689.09
	150	min	**46.2**	-	1582.43	37.7	51.18	3618.04
		max	**79.9**	-	3209.95	75.1	12.44	3444.55
	200	min	**79.2**	-	3770.21	58.5	57.02	3639.17
		max	**131.9**	-	3919.40	126.2	8.88	3523.3
3	50	min	47.4	0	1.01	47.4	0	2.71
		max	43.6	0	0.95	43.6	0	1.69
	100	min	100	0	0	100	0	0
		max	99.7	0	1.7	99.7	0	4.05
	150	min	150	0	0	150	0	0
		max	150	0	0	150	0	0
	200	min	200	0	0	200	0	0
		max	200	0	0	200	0	0

Table 2.8: Average output s-club size, average optimality gap and average running time for the VNS and B&B algorithms (d=0.2).

s	n	VDV	VNS			B&B		
			s-club size	Gap (%)	CPU (sec)	s-club size	Gap (%)	CPU (sec)
2	50	min	23.2	1.29	51.48	23.5	0	20.17
		max	23.8	1.13	57.76	24.1	0	15.74
	100	min	66.5	-	91.87	69.5	3.31	1776.39
		max	71.1	2.05	68.15	72.5	0	416.59
	150	min	135.3	-	368.73	136.1	0.6	1381.29
		max	122.8	-	223.02	123.5	0.09	1013.28
	200	min	195.2	0.15	769.94	195.5	0	1063.19
		max	**177.9**	-	961.32	177.6	0.46	2358.11
3	50	min	50	0	0	50	0	0
		max	48.3	0	0.78	48.3	0	1.62
	100	min	100	0	0	100	0	0
		max	100	0	0	100	0	0
	150	min	150	0	0	150	0	0
		max	150	0	0	150	0	0
	200	min	200	0	0	200	0	0
		max	200	0	0	200	0	0

Table 2.9: Average output s-club size, average optimality gap and average running time for the VNS and B&B algorithms (d=0.25).

s	n	VDV	VNS			B&B		
			s-club size	Gap (%)	CPU (sec)	s-club size	Gap (%)	CPU (sec)
2	50	min	33.5	3.29	38.39	34.6	0	28.57
		max	34.3	0.97	29.09	34.6	0	11.03
	100	min	95.8	0.1	125.93	95.9	0	96.74
		max	88.1	0.12	105.82	88.2	0	138.8
	150	min	149.4	0	86.15	149.4	0	99.53
		max	143.7	0.07	222.98	143.8	0	342.84
	200	min	200	0	0	200	0	0
		max	196.4	0	578.65	196.4	0	131.16
3	50	min	50	0	0	50	0	0
		max	50	0	0	50	0	0
	100	min	100	0	0	100	0	0
		max	100	0	0	100	0	0
	150	min	150	0	0	150	0	0
		max	150	0	0	150	0	0
	200	min	200	0	0	200	0	0
		max	200	0	0	200	0	0

Table 2.10: Average output s-club size, average optimality gap and average running time for the hybrid algorithm.

s	d	n	min VDV			max VDV		
			s-club size	Gap (%)	CPU (sec)	s-club size	Gap (%)	CPU (sec)
2	0.15	100	29.4	3.9	2646.53	40.7	0	1320.1
		150	46.7	40.95	3600.8	79.5	8.02	3461.48
		200	78.8	43.14	3586.09	131.3	6.32	3582.31
3	0.025	100	14.6	0	1.96	15.8	0	18.12
		150	20.5	0	194.82	22.9	0	212.36
		200	25.4	0	816.62	31.9	0	883.87
	0.05	100	28	0	173.1	31.3	0	197.89
		150	47.6	23.54	3595.85	65.3	9.91	3332.801
		200	97	27.4	3607.805	126.3	8.37	3498.733

3. ASYMPTOTIC RESULTS FOR BICLIQUE COMMUNITY DETECTION PROBLEMS

3.1 Introduction

In this chapter, three classes of the maximum biclique problem are considered. Inspired by techniques introduced in [7, 34], we study the asymptotic behavior of biclique structures in large-scale uniform random graphs (URGs). Specifically, we develop asymptotic lower and upper bounds on the value of a solution to the maximum vertex biclique, maximum balanced biclique and maximum edge biclique problems in URGs. We also extend the results to the non-induced versions of these problems. Bounds are insightful in the analysis of the structure of the optimal solution in large-scale real networks. These asymptotic bounds provide knowlede about the evolution and growth rate of the biclique communities as the underlying network evolves and provides a method to predict the future behavior of these structures in a dynamic environment such as biological networks.

3.2 Asymptotic bounds on the biclique size in uniform random graphs

3.2.1 Maximum vertex biclique problem

The following result characterizes the maximum vertex biclique in URGs.

Theorem 5. *Let $G(n,p)$ be a uniform random graph on n vertices, where $p \in [0,1]$ is the probability of having an edge between any two vertices. If the maximum vertex biclique in this graph has size l_1+l_2 where l_1 and l_2 are the sizes of bipartitions, then*

$$l_1 + l_2 \in \left[\frac{\log n}{\log \frac{1}{1-p}}, \frac{16 \log n}{\log \frac{1}{1-p}} \right]$$

asymptotically almost surely.

Proof. First we prove the result for the upper bound. It is enough to show that the probability of having a biclique of size larger than $\frac{16 \log n}{\log \frac{1}{1-p}}$ is very small. Let $Z_{l_1+l_2}$ be the random variable representing the number of bicliques of size $l_1 + l_2$ in G, where l_1 and l_2 are positive integers. Note that $P_r(Z_{l_1+l_2} \geq 1) \leq E[Z_{l_1+l_2}]$ and we have:

$$\begin{aligned} P_r(Z_{l_1+l_2} \geq 1) \leq E[Z_{l_1+l_2}] &= \binom{n}{l_1}\binom{n-l_1}{l_2} p^{l_1 l_2} (1-p)^{\binom{l_1}{2}+\binom{l_2}{2}} \\ &\leq \binom{n}{l_1}\binom{n-l_1}{l_2} (1-p)^{\binom{l_1}{2}+\binom{l_2}{2}}. \end{aligned}$$

Observe that if $l_1 + l_2 \geq 3$ then $\frac{l_1^2 + l_2^2 - (l_1 + l_2)}{2} \geq (\frac{l_1 + l_2}{4})^2$ is always true. (Note that this bound is tight since for $l_1 + l_2 \geq 2$ the relationship would not hold when $l_1 = l_2 = 1$ and if $l_1 = 2$ and $l_2 = 0$ or vice versa the induced subgraph is an independent set instead of a biclique) Therefore,

$$\begin{aligned} E[Z_{l_1+l_2}] &\leq \binom{n}{l_1}\binom{n-l_1}{l_2} (1-p)^{\binom{l_1}{2}+\binom{l_2}{2}} \leq \binom{n}{l_1}\binom{n-l_1}{l_2} (1-p)^{(\frac{l_1+l_2}{4})^2} \\ &\leq (\frac{n^{l_1}}{l_1!})(\frac{(n-l_1)^{l_2}}{l_2!})(1-p)^{(\frac{l_1+l_2}{4})^2} \leq (\frac{n^{l_1}}{l_1!})(\frac{n^{l_2}}{l_2!})(1-p)^{(\frac{l_1+l_2}{4})^2}. \end{aligned}$$

Note that the third inequality above is possible using Stirling's formula. Now assuming $l_1 + l_2 > \frac{16 \log n}{\log \frac{1}{1-p}}$ is true we have:

$$\begin{aligned} (1-p)^{(\frac{l_1+l_2}{4})^2} &= \left[(1-p)^{l_1+l_2}\right]^{\frac{l_1+l_2}{16}} \leq \left[(1-p)^{\frac{16 \log n}{\log \frac{1}{1-p}}}\right]^{\frac{l_1+l_2}{16}} = \left[(1-p)^{\log_{1-p} n^{-16}}\right]^{\frac{l_1+l_2}{16}} \\ &= \left[n^{-16}\right]^{\frac{l_1+l_2}{16}} = n^{-(l_1+l_2)}. \end{aligned}$$

Hence the expected number of bicliques of size $l_1 + l_2$ is bounded above by the

following expression:

$$E[Z_{l_1+l_2}] \leq (\frac{n^{l_1+l_2}}{l_1!l_2!})(1-p)^{(\frac{l_1+l_2}{4})^2} \leq (\frac{n^{l_1+l_2}}{l_1!l_2!})n^{-(l_1+l_2)} = (\frac{1}{l_1!l_2!}) \longrightarrow 0 \quad (\text{as} \quad n \to \infty).$$

This clearly shows that there cannot be a biclique of size larger than $\frac{16 \log n}{\log \frac{1}{1-p}}$ and that if $l_1 + l_2 > \frac{16 \log n}{\log \frac{1}{1-p}}$ then $l_1!l_2! \to \infty$ as $n \to \infty$.

To prove the case for the lower bound, let Z_{1+l_2} be the random variable representing the number of bicliques of size $1+l_2$ in G, where there are one and l_2 vertices in each of the bipartitions, respectively, l_2 being a positive integer. Clearly the set of all bicliques of size $1+l_2$ is a subset of the set of all bicliques of size l_1+l_2 which results into the following:

$$P_r(Z_{l_1+l_2} \geq 1) \geq P_r(Z_{1+l_2} \geq 1) \Rightarrow P_r(Z_{l_1+l_2} = 0) \leq P_r(Z_{1+l_2} = 0).$$

Let $a = 1 + l_2$ for convenience. Therefore it is enough to show that $P_r(Z_a = 0 | a = \frac{\log n}{\log \frac{1}{1-p}})$ is very small for a sufficiently large n. We use the second moment method based on the following relation: $P_r(Z_a = 0) \leq \frac{Var(Z_a)}{E(Z_a)^2}$. Let X_{L_1,L_2} be an indicator variable which assumes value 1, if the nodes in $L_1 \subset V$ and $L_2 \subset V$ form

a biclique, and zero otherwise, where $|L_1| = 1$ and $|L_2| = l_2$.

$$E(Z_a^2) = E\left(\left(\sum_{L_1,L_2} X_{L_1,L_2}\right) \cdot \left(\sum_{\acute{L_1},\acute{L_2}} X_{\acute{L_1},\acute{L_2}}\right)\right) = \sum_{L_1,L_2}\sum_{\acute{L_1},\acute{L_2}} E[X_{L_1,L_2} \cdot X_{\acute{L_1},\acute{L_2}}]$$

$$= \sum_{L_1,L_2}\sum_{\acute{L_1},\acute{L_2}} P_r[X_{L_1,L_2} = 1, X_{\acute{L_1},\acute{L_2}} = 1]$$

$$= \sum_{L_1,L_2}\sum_{\acute{L_1},\acute{L_2}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{L_1,L_2} = 1] \cdot P_r[X_{L_1,L_2} = 1]$$

Since all of the bicliques of type (L_1, L_2) look alike, we can fix (L_1, L_2) as $(\tilde{L_1}, \tilde{L_2})$. Note that here we do not need to take into account the different combinations of L_1 and L_2 sets cardinalities that form a biclique of size $(l_1 + l_2)$ because we only consider the bicliques of type $(1, l_2)$ in which the left bipartition is assumed to have a fixed size of one.

$$E(Z_a^2) = \sum_{L_1,L_2}\sum_{\acute{L_1},\acute{L_2}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{\tilde{L_1},\tilde{L_2}} = 1] \cdot P_r[X_{L_1,L_2} = 1]$$

$$= \sum_{L_1,L_2} P_r[X_{L_1,L_2} = 1] \cdot \sum_{\acute{L_1},\acute{L_2}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{\tilde{L_1},\tilde{L_2}} = 1]$$

$$= \sum_{L_1,L_2} P_r[X_{L_1,L_2} = 1] \cdot \sum_{i=0}^{1}\sum_{j=0}^{l_2} \sum_{\substack{|\acute{L_1}\cap\tilde{L_1}|=i \\ |\acute{L_2}\cap\tilde{L_2}|=j}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{\tilde{L_1},\tilde{L_2}} = 1]$$

$$= \sum_{L_1,L_2} P_r[X_{L_1,L_2} = 1] \cdot$$

$$\sum_{i=0}^{1}\sum_{j=0}^{l_2} \binom{1}{i}\binom{n-1-l_2}{1-i}\binom{l_2}{j}\binom{n-1-l_2-(1-i)}{l_2-j} p^{l_2-ij}(1-p)^{\binom{l_2}{2}-\binom{j}{2}}.$$

Let $\Delta = \sum_{\acute{L_1},\acute{L_2}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{\tilde{L_1},\tilde{L_2}} = 1]$ and observe $E(Z_a) = \sum_{L_1,L_2} P_r[X_{L_1,L_2} =$

1], then $E(Z_a^2) = E(Z_a) \cdot \Delta$ and $P_r(Z_a = 0) \leq \frac{\Delta}{E(Z_a)} - 1$.

$$\frac{\Delta}{E(Z_a)} = \frac{\sum_{i=0}^{1}\sum_{j=0}^{l_2}\binom{1}{i}\binom{n-1-l_2}{1-i}\binom{l_2}{j}\binom{n-1-l_2-(1-i)}{l_2-j}p^{l_2-ij}(1-p)^{\binom{l_2}{2}-\binom{j}{2}}}{\binom{n}{1}\binom{n-1}{l_2}p^{l_2}(1-p)^{\binom{l_2}{2}}}.$$

Let $T_{ij} = \frac{\binom{1}{i}\binom{n-1-l_2}{1-i}\binom{l_2}{j}\binom{n-1-l_2-(1-i)}{l_2-j}p^{-ij}(1-p)^{-\binom{j}{2}}}{\binom{n}{1}\binom{n-1}{l_2}}$, then:

$$\frac{\Delta}{E(Z_a)} = \sum_{i=0}^{1}\sum_{j=0}^{l_2} T_{ij}.$$

In order to prove the case for lower bound we need to show:

$$\frac{\Delta}{E(Z_a)} = 1 + O(n^{\frac{-3}{2}}).$$

Consider the first few terms of the above summation:

$$\begin{aligned}
T_{00} &= \frac{\binom{1}{0}\binom{n-1-l_2}{1}\binom{l_2}{0}\binom{n-l_2-2}{l_2}}{\binom{n}{1}\binom{n-1}{l_2}} = \frac{(n-l_2-1)!(n-l_2-1)!}{n!(n-2l_2-2)!} \\
&= \frac{(n-l_2-1)(n-l_2-2)\cdots(n-2l_2-1)(n-2l_2-2)!(n-l_2-1)!}{n(n-1)(n-2)\cdots(n-l_2)(n-l_2-1)!(n-2l_2-2)!} \\
&= \left[(1-\frac{l_2+1}{n})(1-\frac{l_2+1}{n-1})(1-\frac{l_2+1}{n-2})\cdots(1-\frac{l_2+1}{n-l_2})\right] \\
&= \left[1-\frac{(l_2+1)^2}{n}+O(n^{\frac{-3}{2}})\right] \quad \text{(see appendix A);} \\
T_{10} &= \frac{\binom{1}{1}\binom{n-1-l_2}{0}\binom{l_2}{0}\binom{n-1-l_2}{l_2}}{\binom{n}{1}\binom{n-1}{l_2}} = \frac{1}{n}\cdot\frac{\binom{n-1-l_2}{l_2}}{\binom{n-1}{l_2}} = \frac{1}{n-2l_2-1}T_{00}; \\
T_{01} &= \frac{\binom{1}{0}\binom{n-1-l_2}{1}\binom{l_2}{1}\binom{n-l_2-2}{l_2-1}}{\binom{n}{1}\binom{n-1}{l_2}} = \frac{\binom{n-1-l_2}{1}}{\binom{n}{1}\binom{n-1}{l_2}}\cdot\frac{l_2(n-l_2-2)!}{(l_2-1)!(n-2l_2-1)!} = \frac{l_2^2}{n-2l_2-1}T_{00}.
\end{aligned}$$

Adding up the first three terms results into the following:

$$\begin{aligned}
T_{00} + T_{10} + T_{01} &= T_{00}(1 + \frac{1}{n - 2l_2 - 1} + \frac{l_2^2}{n - 2l_2 - 1}) \\
&= \left[1 - \frac{(l_2+1)^2}{n} + O(n^{\frac{-3}{2}})\right] \left(1 + \frac{(l_2+1)^2 - 2l_2}{n - 2l_2 - 1}\right) \\
&= 1 + O(n^{\frac{-3}{2}}) \quad \text{for} \quad l_2 + 1 = \frac{\log n}{\log \frac{1}{1-p}}.
\end{aligned}$$

To complete the proof we need to show that the remaining part of the summation is also small. For this we bound the remaining terms T_{ij} in terms of T_{02}.

$$\begin{aligned}
\frac{T_{ij}}{T_{02}} &= \frac{\binom{1}{i}\binom{n-1-l_2}{1-i}\binom{l_2}{j}\binom{n-l_2+i-2}{l_2-j} p^{-ij}(1-p)^{-\binom{j}{2}}}{\binom{1}{0}\binom{n-1-l_2}{1}\binom{l_2}{2}\binom{n-l_2-2}{l_2-2}(1-p)^{-1}} \\
&= \frac{2!(n-2l_2)![(l_2-2)!]^2}{i![(1-i)!]^2(n-2l_2+i+j-2)!j![(l_2-j)!]^2 p^{ij}(1-p)^{\binom{j}{2}-1}} \\
&= \frac{2![(l_2-2)(l_2-3)\cdots(l_2-j+1)]^2}{j!(n-2l_2+i+j-2)(n-2l_2+i+j-3)\cdots(n-2l_2+1)p^{ij}(1-p)^{\binom{j}{2}-1}} \\
&\leq \frac{2!(l_2^2)^{j-2}}{j!(n-2l_2)^i(n-2l_2)^{j-2}p^{ij}(1-p)^{\binom{j}{2}-1}}.
\end{aligned}$$

The third equality is possible because $i![(1-i)!]^2 = 1$ for $i \in \{0,1\}$. Note that $\frac{T_{03}}{T_{02}} = \frac{(l_2-2)^2}{3(n-2l_2+1)(1-p)^2} \leq 1$ for sufficiently large values of n. Similarly $\frac{T_{12}}{T_{02}} = \frac{1}{(n-2l_2+1)p^2} \leq 1$ for sufficiently large n. Likewise $\frac{T_{11}}{T_{02}} = \frac{2(1-p)}{(l_2-1)^2 p} \leq 1$ provided that $l_2 \geq 1 + [2(\frac{1}{p} - 1)]^{\frac{1}{2}}$. It is easy to check that if $a = \frac{\log n}{\log \frac{1}{1-p}}$, where $a = l_2 + 1$, the above condition on l_2 will always be satisfied for any fixed value of $p \in (0,1)$ and sufficiently large n. For all other values of $i \in \{0,1\}$ and $j \in [2, l_2]$ we get $\frac{T_{ij}}{T_{02}} \leq 1$

for sufficiently large n. Moreover we have:

$$\begin{aligned}
T_{02} &= \frac{\binom{1}{0}\binom{n-l_2-1}{1}\binom{l_2}{2}\binom{n-l_2-2}{l_2-2}}{\binom{n}{1}\binom{n-1}{l_2}(1-p)} = \frac{[l_2(l_2-1)]^2[(n-l_2-1)!]^2}{2!n!(n-2l_2)!} \\
&= \frac{[l_2(l_2-1)]^2(n-l_2-1)(n-l_2-2)\cdots(n-2l_2+1)}{2!n(n-1)\cdots(n-l_2)} \\
&\leq \frac{[l_2(l_2-1)]^2[(n-l_2-1)]^{l_2-1}}{2![(n-l_2)]^{l_2+1}} \\
&= \frac{[l_2(l_2-1)]^2}{2} \cdot \left[(1-\frac{1}{n-l_2})\right]^{l_2-1} \cdot \frac{1}{(n-l_2)^2} \longrightarrow 0 \quad (n \to \infty).
\end{aligned}$$

Therefore:

$$\sum_{i=0}^{1}\sum_{j=2}^{l_2} T_{ij} + T_{11} \leq \sum_{i=0}^{1}\sum_{j=2}^{l_2} T_{02} + T_{02} \to 0 \quad (\text{as} \quad n \to \infty \quad \text{for} \quad a = \frac{\log n}{\log \frac{1}{1-p}})$$

and we have the following:

$$\frac{\Delta}{E(Z_a)} = \sum_{i=0}^{1}\sum_{j=0}^{l_2} T_{ij} = T_{00} + T_{01} + T_{10} + \sum_{i=0}^{1}\sum_{j=2}^{l_2} T_{ij} + T_{11} = 1 + O(n^{\frac{-3}{2}}).$$

Hence,

$$P_r(Z_a = 0) = P_r(Z_{1+l_2} = 0) \leq O(n^{\frac{-3}{2}}) \Rightarrow P_r(Z_{l_1+l_2} = 0) \leq O(n^{\frac{-3}{2}}),$$

which completes the proof. □

The above theorem shows that the size of maximum vertex bicliques in large-scale networks is of order $O(\log n)$ implying slow growth rate on the size of these structures as the underlying network grows.

3.2.2 Maximum balanced biclique problem

The next result establishes asymptotic bounds on the size of maximum balanced bicliques in URGs. In the following, we adopt the convention to use $a \times a$ to represent a balanced biclique of vertex cardinality $2a$.

Theorem 6. *Let $G(n,p)$ be a uniform random graph on n vertices, where $p \in [0,1]$ is the probability of having an edge between any two vertices, and let $a(n) = \dfrac{\log n}{\log \frac{1}{1-p}}$. If the maximum balanced biclique in this graph has size $a \times a$, then*

$$a \in [a(n), 4a(n)]$$

asymptotically almost surely.

Proof. First we prove the result for the upper bound. It is enough to show that the probability of having a balanced biclique of size larger than $4a(n) \times 4a(n)$ is very small. Let Z_a be the random variable representing the number of bicliques of size $a \times a$ in G. Note that $P_r(Z_a \geq 1) \leq E[Z_a]$ and we have:

$$\begin{aligned} P_r(Z_a \geq 1) \leq E[Z_a] &= \binom{n}{a}\binom{n-a}{a} p^{a^2} (1-p)^{2\binom{a}{2}} \\ &\leq \binom{n}{a}\binom{n-a}{a} (1-p)^{2\binom{a}{2}} = \binom{n}{a}\binom{n-a}{a} (1-p)^{a(a-1)}. \end{aligned}$$

Observe that for $a \geq 2$ we have $a(a-1) \geq \dfrac{a^2}{2}$, which is easy to check by factoring a from both sides. Also note that we can assume $a \geq 2$ without loss of generality

since $a = 1$ is a trivial case as every edge is a balanced biclique of size two. Therefore,

$$E[Z_a] \leq \binom{n}{a}\binom{n-a}{a}(1-p)^{a(a-1)} \leq \binom{n}{a}\binom{n-a}{a}(1-p)^{\frac{a^2}{2}}$$
$$\leq (\frac{n^a}{a!})(\frac{(n-a)^a}{a!})(1-p)^{\frac{a^2}{2}} \leq (\frac{n^a}{a!})(\frac{n^a}{a!})(1-p)^{\frac{a^2}{2}}.$$

Note that the third inequality above is possible using Stirling's formula. Now assuming $a > 4a(n)$ is true we have:

$$(1-p)^{\frac{a^2}{2}} = [(1-p)^a]^{\frac{a}{2}} \leq [(1-p)^{\frac{4\log n}{\log\frac{1}{1-p}}}]^{\frac{a}{2}} = [(1-p)^{\log_{1-p} n^{-4}}]^{\frac{a}{2}}$$
$$= [n^{-4}]^{\frac{a}{2}} = n^{-2a}.$$

Hence, the expected number of balanced bicliques of size $a \times a$ is bounded above by the following expression:

$$P_r(Z_a \geq 1) \leq E[Z_a] \leq \frac{n^{2a}}{(a!)^2}(1-p)^{\frac{a^2}{2}} \leq \frac{n^{2a}}{(a!)^2}n^{-2a} = (\frac{1}{a!})^2 \longrightarrow 0 \quad (\text{as} \quad n \to \infty).$$

The above relation clearly shows that if $a > 4a(n)$, the expected number of balanced bicliques and thus the probability of having at least one such biclique is very small for sufficiently large values of n and proves the claim for upper bound.

To prove the case for lower bound, we need to show that there is a balanced biclique of size $a(n) \times a(n)$ in G with high probability. Therefore it is enough to show that $P_r(Z_a = 0|a = a(n))$ is very small for a sufficiently large n. We use the second moment method based on the following relation: $P_r(Z_a = 0) \leq \frac{Var(Z_a)}{E(Z_a)^2}$. Let X_{L_1,L_2} be an indicator variable which assumes value 1, if the nodes in $L_1 \subset V$

and $L_2 \subset V \setminus L_1$ form a balanced biclique and zero otherwise, where $|L_1| = |L_2| = a$.

$$E(Z_a^2) = E\left(\left(\sum_{L_1,L_2} X_{L_1,L_2}\right) \cdot \left(\sum_{\acute{L_1},\acute{L_2}} X_{\acute{L_1},\acute{L_2}}\right)\right) = \sum_{L_1,L_2}\sum_{\acute{L_1},\acute{L_2}} E[X_{L_1,L_2} \cdot X_{\acute{L_1},\acute{L_2}}]$$
$$= \sum_{L_1,L_2}\sum_{\acute{L_1},\acute{L_2}} P_r[X_{L_1,L_2} = 1, X_{\acute{L_1},\acute{L_2}} = 1]$$
$$= \sum_{L_1,L_2}\sum_{\acute{L_1},\acute{L_2}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{L_1,L_2} = 1] \cdot P_r[X_{L_1,L_2} = 1].$$

Since all of the bicliques of type (L_1, L_2) look alike, we can fix (L_1, L_2) as $(\tilde{L_1}, \tilde{L_2})$. This is due to the fact that the probability of all such randomly selected subsets of vertices L_1 and L_2 to form a biclique is the same. The key issue here is that the size of the bipartitions is fixed and equal, by definition of the balanced biclique, which rules out the combinatorics when considering this fixation. Therefore:

$$E(Z_a^2) = \sum_{L_1,L_2}\sum_{\acute{L_1},\acute{L_2}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{\tilde{L_1},\tilde{L_2}} = 1] \cdot P_r[X_{L_1,L_2} = 1]$$
$$= \sum_{L_1,L_2} P_r[X_{L_1,L_2} = 1] \cdot \sum_{\acute{L_1},\acute{L_2}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{\tilde{L_1},\tilde{L_2}} = 1]$$
$$= \sum_{L_1,L_2} P_r[X_{L_1,L_2} = 1] \cdot \sum_{i=0}^{a}\sum_{j=0}^{a} \sum_{\substack{|\acute{L_1} \cap \tilde{L_1}| = i \\ |\acute{L_2} \cap \tilde{L_2}| = j}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{\tilde{L_1},\tilde{L_2}} = 1]$$
$$= \sum_{L_1,L_2} P_r[X_{L_1,L_2} = 1] \cdot$$
$$\sum_{i=0}^{a}\sum_{j=0}^{a} \binom{a}{i}\binom{n-a-a}{a-i}\binom{a}{j}\binom{n-a-a-(a-i)}{a-j} p^{a^2-ij}(1-p)^{2\binom{a}{2}-\binom{i}{2}-\binom{j}{2}}.$$

Let $\Delta = \sum_{\acute{L_1},\acute{L_2}} P_r[X_{\acute{L_1},\acute{L_2}} = 1 | X_{\tilde{L_1},\tilde{L_2}} = 1]$ and observe $E(Z_a) = \sum_{L_1,L_2} P_r[X_{L_1,L_2} = 1]$ then we have $E(Z_a^2) = E(Z_a) \cdot \Delta$ and as a result $P_r(Z_a = 0) \leq \frac{\Delta}{E(Z_a)} - 1$.

$$\frac{\Delta}{E(Z_a)} = \frac{\sum_{i=0}^{a}\sum_{j=0}^{a}\binom{a}{i}\binom{n-a-a}{a-i}\binom{a}{j}\binom{n-a-a-(a-i)}{a-j}p^{a^2-ij}(1-p)^{2\binom{a}{2}-\binom{i}{2}-\binom{j}{2}}}{\binom{n}{a}\binom{n-a}{a}p^{a^2}(1-p)^{2\binom{a}{2}}}.$$

Let $T_{ij} = \frac{\binom{a}{i}\binom{n-a-a}{a-i}\binom{a}{j}\binom{n-a-a-(a-i)}{a-j}p^{-ij}(1-p)^{-\binom{i}{2}-\binom{j}{2}}}{\binom{n}{a}\binom{n-a}{a}}$, then:

$$\frac{\Delta}{E(Z_a)} = \sum_{i=0}^{a}\sum_{j=0}^{a} T_{ij}.$$

In order to prove the case for lower bound we need to show

$$\frac{\Delta}{E(Z_a)} = 1 + O(n^{\frac{-3}{2}}).$$

Consider the first few terms of the above summation:

$$\begin{aligned}
T_{00} &= \frac{\binom{a}{0}\binom{n-2a}{a}\binom{a}{0}\binom{n-3a}{a}}{\binom{n}{a}\binom{n-a}{a}} = \frac{\binom{n-2a}{a}\binom{n-3a}{a}}{\binom{n}{a}\binom{n-a}{a}} = \frac{(n-2a)!(n-2a)!}{n!(n-4a)!} \\
&= \frac{(n-2a)(n-2a-1)(n-2a-2)\cdots(n-4a+2)(n-4a+1)(n-4a)!(n-2a)!}{n(n-1)(n-2)\cdots(n-2a+2)(n-2a+1)(n-2a)!(n-4a)!} \\
&= \left[(1-\frac{2a}{n})(1-\frac{2a}{n-1})\cdots(1-\frac{2a}{n-(2a-2)})(1-\frac{2a}{n-(2a-1)})\right] \\
&= \left[1-\frac{(2a)^2}{n}+O(n^{\frac{-3}{2}})\right] \quad \text{(see appendix A);} \\
T_{10} &= \frac{\binom{a}{1}\binom{n-2a}{a-1}\binom{a}{0}\binom{n-3a+1}{a}}{\binom{n}{a}\binom{n-a}{a}} = \frac{a^2}{n-4a+1}T_{00}.
\end{aligned}$$

The second equality above follows because $\binom{n-2a}{a-1} = \frac{a}{n-3a+1}\binom{n-2a}{a}$ and $\binom{n-3a+1}{a} = \frac{n-3a+1}{n-4a+1}\binom{n-3a}{a}$. Similarly $\binom{n-3a}{a-1} = \frac{a}{n-4a+1}\binom{n-3a}{a}$ and therefore

$$T_{01} = \frac{\binom{a}{0}\binom{n-2a}{a}\binom{a}{1}\binom{n-3a}{a-1}}{\binom{n}{a}\binom{n-a}{a}} = \frac{a^2}{n-4a+1}T_{00}.$$

Adding up the first three terms results into the following:

$$T_{00}+T_{10}+T_{01} = T_{00}(1+\frac{2a^2}{n-4a+1}) = \left[1-\frac{(2a)^2}{n}+O(n^{\frac{-3}{2}})\right]\left(1+\frac{2a^2}{n-4a+1}\right)$$
$$= 1+O(n^{\frac{-3}{2}}) \quad \text{for} \quad a=\frac{\log n}{\log\frac{1}{1-p}} \quad \text{and sufficiently large} \quad n.$$

To complete the proof we need to show that the remaining part of the summation is small. For this we bound the remaining terms T_{ij} in terms of T_{11}.

$$\frac{T_{ij}}{T_{11}} = \frac{\binom{a}{i}\binom{n-2a}{a-i}\binom{a}{j}\binom{n-3a+i}{a-j}p^{-ij}(1-p)^{-\binom{i}{2}-\binom{j}{2}}}{\binom{a}{1}\binom{n-2a}{a-1}\binom{a}{1}\binom{n-3a+1}{a-1}p^{-1}}$$
$$= \frac{(n-4a+2)![(a-1)!]^2[(a-1)!]^2p^{-ij+1}(1-p)^{-\binom{i}{2}-\binom{j}{2}}}{i!j!(n-4a+i+j)![(a-i)!]^2[(a-j)!]^2}$$
$$\leq \left(\frac{a^2}{n-4a}(1-p)^{-\frac{i}{2}}\right)^{i-1}p^{\frac{-ij+1}{2}}\left(\frac{a^2}{n-4a}(1-p)^{-\frac{j}{2}}\right)^{j-1}p^{\frac{-ij+1}{2}}.$$

First note that $\frac{T_{12}}{T_{11}} = \frac{T_{21}}{T_{11}} = \frac{(a-1)^2}{2(n-4a+3)p(1-p)} \leq 1$ for sufficiently large n. Also, for $i \geq 2$, we have $-(i-1)j \leq \frac{-ij+1}{2}$ and if $j \geq 2$, then $-(j-1)i \leq \frac{-ij+1}{2}$ is true. Thus,

$$\frac{T_{ij}}{T_{11}} \leq \left(\frac{a^2}{n-4a}p^{-j}(1-p)^{-\frac{i}{2}}\right)^{i-1}\left(\frac{a^2}{n-4a}p^{-i}(1-p)^{-\frac{j}{2}}\right)^{j-1}.$$

Based on the above inequality for any value of $a = a^*(n) = \frac{(1-\epsilon)\log n}{\log\frac{1}{1-p}}$, $0 < \epsilon < 1$, we obtain $\frac{T_{ij}}{T_{11}} \leq 1$ when n is sufficiently large for all terms in which $i \geq 2$ and $j \geq 2$

as well as the terms in which $i \geq 0$ and $j \geq 2$ and vice versa. Moreover we have:

$$T_{11} = \frac{\binom{a}{1}\binom{n-2a}{a-1}\binom{a}{1}\binom{n-3a+1}{a-1}}{\binom{n}{a}\binom{n-a}{a}p} \leq \frac{\binom{a}{1}\binom{n-2a}{a-1}\binom{a}{1}\binom{n-2a}{a-1}}{\binom{n-a}{a}\binom{n-a}{a}p}$$
$$\leq \frac{\frac{a^4}{(n-3a+1)^2}\binom{n-2a}{a}^2}{\binom{n-a}{a}^2 p} \quad \to 0 \quad (\text{as} \quad n \to \infty).$$

For the same choice of $a = a^*(n) = \frac{(1-\epsilon)\log n}{\log \frac{1}{1-p}}$. Therefore,

$$\sum_{i=1}^{a}\sum_{j=1}^{a} T_{ij} + \sum_{j=2}^{a} T_{0j} + \sum_{i=2}^{a} T_{i0} \leq$$
$$\sum_{i=1}^{a}\sum_{j=1}^{a} T_{11} + \sum_{j=2}^{a} T_{11} + \sum_{i=2}^{a} T_{11} \quad \to 0 \quad (\text{as} \quad n \to \infty \quad \text{for} \quad a = a^*(n)),$$

Hence,

$$\frac{\Delta}{E(Z_a)} = \sum_{i=0}^{a}\sum_{j=0}^{a} T_{ij} = 1 + O(n^{\frac{-3}{2}}) \Longrightarrow P_r(Z_a = 0) \leq O(n^{\frac{-3}{2}}),$$

which completes the proof.

□

Theorem 6 indicates the growth rate of the largest balanced biclique with respect to the size of the underlying network in URGs. The evolution is a logarithmic function of the network size which is not a rapid growth. Also comparing the upper bounds obtained in Theorems 5 and 6, the cardinality of the maximum vertex biclique can be twice as large as that of the maximum balanced biclique in G which is expected due to a more restrictive nature of the balanced bicliques.

Using the result obtained in the above theorem, we can improve the lower bound for the maximum vertex biclique problem derived independently in Theorem 5. Note

that the cardinality of the maximum vertex biclique is at least as large as the cardinality of the maximum balanced biclique in any arbitrary graph G. Therefore the cardinality of the maximum vertex biclique in uniform random graph $G(n,p)$, represented by l_1+l_2, is bounded from below by $\frac{2\log n}{\log\frac{1}{1-p}}$, which is tighter than the previous lower bound.

Theorem 7. *Let $G(n,p)$ be a uniform random graph on n vertices, where $p\in[0,1]$ is the probability of having an edge between any two vertices and let $a(n)=\dfrac{\log n}{\log\frac{1}{p}}$. If the maximum balanced non-induced biclique in this graph has size $a\times a$, then a is bounded above by $2a(n)$ asymptotically almost surely.*

Proof. We show that the probability of having a balanced non-induced biclique of size larger than $2a(n)\times 2a(n)$ is very small. We define Z_a to be the random variable representing the number of bicliques of size $a\times a$ in G. Using known inequality $P_r(Z_a\geq 1)\leq E[Z_a]$ we have:

$$P_r(Z_a\geq 1)\leq E[Z_a]=\binom{n}{a}\binom{n-a}{a}p^{a^2}\leq\binom{n}{a}\binom{n}{a}p^{a^2}\leq\frac{n^a}{a!}\frac{n^a}{a!}p^{a^2},$$

where the last inequality follows from Stirling's formula. Assuming $a>2a(n)$ we have:

$$p^{a^2}=[p^a]^a\leq[p^{\frac{2\log n}{\log\frac{1}{p}}}]^a=[p^{\log_p n^{-2}}]^a=[n^{-2}]^a=n^{-2a}.$$

Hence, the expected number of balanced non-induced bicliques of size $a\times a$ is bounded above by the following expression:

$$P_r(Z_a\geq 1)\leq E[Z_a]\leq\frac{n^{2a}}{(a!)^2}p^{a^2}\leq\frac{n^{2a}}{(a!)^2}n^{-2a}=(\frac{1}{a!})^2\longrightarrow 0\quad(\text{as}\quad n\to\infty),$$

The above relation clearly shows that if $a > 2a(n)$, the expected number of balanced non-induced bicliques and thus the probability of having at least one such biclique is very small for sufficiently large values of n and proves the claim for upper bound. □

Corollary 1. *Let $G(n,p)$ be a uniform random graph on n vertices, where $p \in [0,1]$ is the probability of having an edge between any two vertices and let $a(n) = \frac{\log n}{\log \frac{1}{p}}$. If the maximum non-induced biclique in this graph has size $l_1 + l_2$ where l_1 and l_2 are the sizes of bipartitions, then $l_1 + l_2$ is bounded below by $2a(n)$ asymptotically almost surely.*

Proof. The upper bound obtained for the maximum balanced non-induced biclique, Theorem 7, serves as a lower bound for the *maximum non-induced biclique problem* since the cardinality of the maximum non-induced biclique is at least equal to the cardinality of the maximum balanced non-induced biclique in G. □

3.2.3 Maximum edge biclique problem

Using the inherent relationship between maximum biclique problems and the result of the previous theorems, we develop the asymptotic bounds for the maximum edge biclique problem in URGs.

Theorem 8. *Let $G(n,p)$ be a uniform random graph on n vertices, where $p \in [0,1]$ is the probability of having an edge between any two vertices. If the maximum edge biclique in this graph has size $e(n)$, then*

$$e(n) \in \left[\left(\frac{\log n}{\log \frac{1}{1-p}} \right)^2, \left(\frac{8 \log n}{\log \frac{1}{1-p}} \right)^2 \right]$$

asymptotically almost surely.

Proof. We prove the claim using the results of two previous Theorems. First note that the maximum balanced biclique is a restricted version of the maximum vertex biclique and would not necessarily provide an upper bound on the size of maximum edge biclique for any given graph G. However, the result on the maximum vertex biclique can be used to develop an upper bound. Recall that in Theorem 5, it was shown that the size of maximum vertex biclique in random graphs is bounded above by $\frac{16 \log n}{\log \frac{1}{1-p}}$. It is easy to check that the number of edges will be maximized if these vertices are equally devided between the two bipartitions which results in $\left(\frac{8 \log n}{\log \frac{1}{1-p}} \right)^2$ edges for the biclique, thus proving the claim for the upper bound.

For the lower bound, observe that the number of edges in a maximum balanced biclique in G provides a lower bound on the maximum edge biclique of G. Recall that in Theorem 6, we proved that in a random graph the size of each bipartition in the maximum balanced biclique is bounded below by $\frac{\log n}{\log \frac{1}{1-p}}$. Therefore the maximum edge biclique cannot be of size less than $\left(\frac{\log n}{\log \frac{1}{1-p}} \right)^2$, which proves the claim for lower bound. □

Therefore the growth rate for the maximum edge biclique is at most of order $O((\log n)^2)$. Although these variations of the maximum biclique problem seem to be similar, the bounds reveal interesting relationships. For example comparing the results of Theorems 6 and 8 it can be easily seen that in URG's the size of the maximum edge biclique, $64 \left(\frac{\log n}{\log \frac{1}{1-p}} \right)^2$, may be significantly larger than the cardinality of edges in a maximum balanced biclique, $16 \left(\frac{\log n}{\log \frac{1}{1-p}} \right)^2$, which may not be so obvious in the first place.

4. EXACT ALGORITHMS FOR THE MAXIMUM BICLIQUE PROBLEMS

In this chapter we propose two exact methods for solving two variants of the maximum biclique problem. Given a simple undirected graph $G = (V, E)$, the *maximum vertex biclique*, MVB, and the *maximum edge biclique*, MEB, problems are concerned with finding the maximum vertex cardinality and maximum edge cardinality induced bicliques in G respectively. The MEB is NP-complete in general and bipartite graphs and the MVB is NP-complete in general graphs. A complete review of the computational complexity and solution methods is given in Chapter 1. Many of the proposed methods lack the ability of providing an exact solution and are not suitable for practical large-scale problems. In this chapter we propose two exact solution methods, namely a novel scale-reduction algorithm and a combinatorial branch-and-bound algorithm, for solving these classes of the maximum biclique problem. The scale-reduction algorithm recursively identifies and removes the edges that cannot be included in the optimal solution and the final residual graph, which is not further reducible, serves as a new instance of the problem to be solved. On the other hand the branch-and-bound algorithm takes advantage of the heredity property of bicliques to explore the search space for local and global optimum. Computational experiments are provided to compare the performance of these methods. Next we propose the mathematical programming formulations which will be used after the scale-reduction procedure to find the optimal solution.

4.1 Mathematical programming formulations

In this section, we provide mathematical formulations for the maximum vertex and maximum edge biclique problems in a simple general graph $G = (V, E)$. Let x_{ij}

be the binary decision variable taking value one if vertex $i \in I = \{1, ..., n\}$ belongs to either of the two bipartitions $J = \{1, 2\}$ and zero otherwise. Then a binary 0-1 formulation for the maximum vertex biclique is as follows:

$$\text{Maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{2} x_{ij} \tag{4.1}$$

s.t:

$$\sum_{j \in J} x_{ij} \leq 1 \quad \forall i \in I \tag{4.2}$$

$$\sum_{i \in I} x_{ij} \geq 1 \quad \forall j \in J \tag{4.3}$$

$$x_{ij} + x_{kj} \leq 1 \quad \forall (i,k) \in E, \forall j \in J \tag{4.4}$$

$$x_{ij_1} + x_{kj_2} \leq 1 \quad \forall (i,k) \notin E, \forall \{j_1, j_2\} \in J, j_1 \neq j_2 \tag{4.5}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in I, \forall j \in J \tag{4.6}$$

In the above formulation the first constraint makes sure that each vertex will only belong to at most one partition. Second constraint rules out solutions with empty bipartitions as they result in having an independent set. Third constraint implies that vertices incident to an edge cannot belong to the same bipartition. Fourth constraint ensures that if there is no edge between two vertices, they cannot belong to different bipartitions. Considering an instance with $|V| = n$ vertices and $|E| = m$ edges, the above formulation has $2n$ variables and $n^2 + 2$ constraints, excluding the binary requirements. Next, we propose a mathematical formulation for the maximum edge biclique problem. As in the previous model, let x_{ij} be the binary decision variable taking value one if vertex $i \in I = \{1, ..., n\}$ belongs to either of the two bipartitions $J = \{1, 2\}$ and zero otherwise. Also let z_{ik} be the binary variable with value one if

edge (i,k) is included in the solution and zero otherwise.

$$\text{Maximize} \quad \sum_{(i,k)\in E} z_{ik} \tag{4.7}$$

s.t:

$$\sum_{j\in J} x_{ij} \leq 1 \quad \forall i \in I \tag{4.8}$$

$$\sum_{i\in I} x_{ij} \geq 1 \quad \forall j \in J \tag{4.9}$$

$$x_{ij} + x_{kj} \leq 1 \quad \forall (i,k) \in E, \forall j \in J \tag{4.10}$$

$$x_{ij_1} + x_{kj_2} \leq 1 \quad \forall (i,k) \notin E, \forall \{j_1, j_2\} \in J, j_1 \neq j_2 \tag{4.11}$$

$$z_{ik} \leq \sum_{j\in J} x_{ij} \quad \forall (i,k) \in E, \quad k > i = 1, ..., n \tag{4.12}$$

$$z_{ik} \leq \sum_{j\in J} x_{kj} \quad \forall (i,k) \in E, \quad k > i = 1, ..., n \tag{4.13}$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \tag{4.14}$$

$$z_{ik} \in \{0,1\}, \quad \forall (i,k) \in E, \quad k > i = 1, ..., n \tag{4.15}$$

In the above model constraints (4.8)- (4.11) are the same as (4.2)- (4.5) in the previous model. Constraints (4.12) and (4.13) together ensure that an edge is in the solution only if its incident vertices are selected to be in the solution. In fact (4.12), and likewise (4.13), is an aggregated version of $z_{ik} \leq x_{i1}$ and $z_{ik} \leq x_{i2}$. This is possible since at most one of x_{i1} and x_{i2} can be nonzero. Also note that the integrality of z_{ik} can be relaxed due to the fact that these variables are bounded by x_{ij} variables and the objective function is maximization. Considering an instance with $|V| = n$ vertices and $|E| = m$ edges, the above formulation has $2n + m$ variables and $n^2 + 2m + 2$ constraints, in addition to the binary requirements.

Due to a considerably large number of constraints, solving these mathematical

programs directly using commercial solvers is not always possible, especially for practical applications in biology and gene research, where instances are massive. In the next section we propose a scale-reduction method that is effective for solving large-scale instances.

4.2 Scale-reduction algorithm

4.2.1 Reduction technique and properties

We first explore some structural properties of the bicliques which will be used later in developing the algorithms. The properties and algorithms developed in this section are for the *maximum vertex biclique problem* and can be extended for the *maximum edge biclique problem*. Let $G = (V, E)$ be a simple undirected graph and $B = (V_1' \cup V_2', E')$ be a subgraph of G induced by a biclique with bipartitions V_1' and V_2' and $E'(B) = \{(i, j) \in E : i \in V_1', j \in V_2'\}$. Let $\mathcal{L} = |V_1' \cup V_2'|$ be a known lower bound on the size of MVB of G. The following facts about bicliques can be accepted without proof:

1. *Bicliques posses heredity property, meaning that a subset of a biclique is a biclique.*

2. *Every biclique is a 2-club. Given a pre-determined lower bound $\mathcal{L}$ on the size of MVB of G, this property can be used to rule out vertices $i \in V$ for which $|N_G^2(i)| < \mathcal{L}$.*

3. *Let B be a subgraph of G induced by a biclique defined as above. Consider any given edge $(i, j) \in E'$ where $i \in V_1'$ and $j \in V_2'$. Obviously $\alpha_{B[N(i)]} = |V_2'|$ and likewise $\alpha_{B[N(j)]} = |V_1'|$ yielding the following:*

$$\alpha_{B[N(i)]} + \alpha_{B[N(j)]} = |V_1' \cup V_2'|.$$

Consider a graph G and a subgraph B induced by a biclique of size $\mathcal{L}$ in G as defined above. For a subset of edges $H \in E$, the corresponding *edge-induced subgraph* $G[H]$ of G is given by $G[H] = (V(H), H)$, where $V(H) = \{s \in V : \exists (i,j) \in H, s = i \vee s = j\}$. Define the following operator function for $(u,v) \in H$:

$$P_G^{\mathcal{L}}(H,(u,v)) = \begin{cases} (u,v), & \text{if} \quad \alpha_{G[N_{G[H]}(u) \setminus N_G(v)]} + \alpha_{G[N_{G[H]}(v) \setminus N_G(u)]} \geq \mathcal{L}, \\ \emptyset, & \text{otherwise .} \end{cases}$$

The operator function determines whether an edge (u,v) induces two independent set partitions of total size at least $\mathcal{L}$ in subgraphs induced by $\{N_{G[H]}(u) \setminus N_G(v)\}$ and $\{N_{G[H]}(v) \setminus N_G(u)\}$ in G. Similarly we can define this operation for a graph $G[H]$ as follows:

$$P_G^{\mathcal{L}}(H) = \bigcup_{(u,v) \in H} P_G^{\mathcal{L}}(H,(u,v)).$$

The result of this operation is a subset of edges in H that satisfy the operator function condition. For a given lower bound $\mathcal{L}$ and a positive integer n, denote by $P^{\mathcal{L}}(E,n)$ the following recursively defined set:

$$P^{\mathcal{L}}(E,n) = \begin{cases} P_G^{\mathcal{L}}(E), & \text{if } n = 1, \\ P^{\mathcal{L}}(P^{\mathcal{L}}(E,n-1),1), & \text{if } n \geq 2. \end{cases}$$

According to the above definition, for $n \geq 1$, $P^{\mathcal{L}}(E,n)$ is a subset of E satisfying the condition defined for the operator function above and inducing a subgraph $G[P^{\mathcal{L}}(E,n)]$ of G. Figure 4.1 illustrates the definition on a 8-vertex graph G with a known biclique B having a vertex set $V_1' \cup V_2' = \{\{u,w\} \cup \{v,x\}\}$ that is considered as a lower bound $\mathcal{L} = |V_1' \cup V_2'| = 4$ on the size of MVB in G. Consider an edge

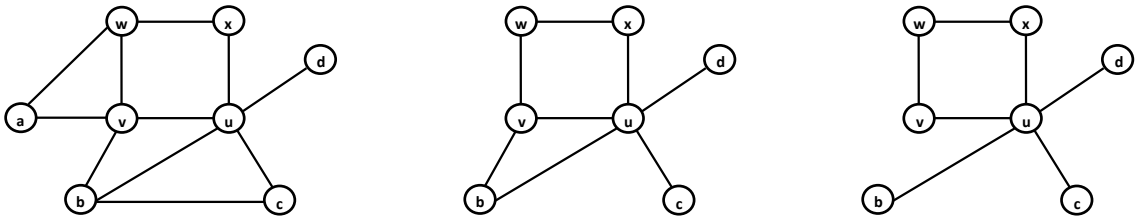

Figure 4.1: Graph G (left) and the induced subgraphs $G[P^{\mathcal{L}}(E,1)]$ (middle) and $G[P^{\mathcal{L}}(E,2)]$ (right).

(a,w) and check $P^{\mathcal{L}}(E,n)$ for $n=1$. Observe that $G[N_G(a)\setminus N_G(w)] = (\{w\},\emptyset)$ and $\alpha_{G[N_G(a)\setminus N_G(w)]} = 1$. Likewise $G[N_G(w)\setminus N_G(a)] = (\{a,x\},\emptyset)$ and $\alpha_{G[N_G(w)\setminus N_G(a)]} = 2$. As a result,

$$\alpha_{G[N_G(a)\setminus N_G(w)]} + \alpha_{G[N_G(w)\setminus N_G(a)]} = 3 < \mathcal{L} = 4.$$

Therefore (a,w) does not satisfy the condition and can be removed, *i.e.* $P^{\mathcal{L}}_G(E,(a,w)) = \emptyset$. Also for the edge (v,u), $G[N_G(v)\setminus N_G(u)] = (\{a,w,u\},(a,w))$ and $G[N_G(u)\setminus N_G(v)] = (\{x,v,c,d\},\emptyset)$ resulting in $\alpha_{G[N_G(v)\setminus N_G(u)]} + \alpha_{G[N_G(u)\setminus N_G(v)]} = 6 > \mathcal{L} = 4$ and $P^{\mathcal{L}}_G(E,(v,u)) = (v,u)$. Obviously all edges for which $P^{\mathcal{L}}_G(E,\cdot) = \emptyset$ should be removed at once when every possible edge in G has been investigated. Based on the definition, $P^{\mathcal{L}}(E,1) = \{(u,b),(u,c),(u,d),(u,v),(u,x),(v,b),(v,w),(w,x)\}$ and $P^{\mathcal{L}}(E,2) = P^{\mathcal{L}}(P^{\mathcal{L}}(E,1),1) = \{(u,b),(u,c),(u,d),(u,v),(u,x),(v,w),(w,x)\}$ for which the edge-induced subgraphs have been shown in Fig 4.1. Moreover, for any $n \geq 2$ the result of applying the recursive function on edge-induced subgraph $G[P^{\mathcal{L}}(E,2)]$ remains the same, thus, $P^{\mathcal{L}}(E,n) = \{(u,b),(u,c),(u,d),(u,v),(u,x),(v,w),(w,x)\}$.

Next we discuss properties of the recursive function which will be used to develop and validate the scale-reduction algorithm.

Lemma 2. *Let $B = (V_1' \cup V_2', E')$ and $B^* = (V_1^* \cup V_2^*, E^*)$ be bicliques in $G = (V,E)$ with $|B| = \mathcal{L}$, $|B^*| = \mathcal{L}^*$ and $\mathcal{L} \leq \mathcal{L}^*$. Then for any integer $n \geq 1$ the following relations hold.*

1. $P^{\mathcal{L}}(E, n+1) \subseteq P^{\mathcal{L}}(E, n)$;

2. $P^{\mathcal{L}^*}(E, n) \subseteq P^{\mathcal{L}}(E, n)$;

3. $P^{\mathcal{L}^*}(E, n+1) \subseteq P^{\mathcal{L}}(E, n)$.

Proof. The first inclusion results from the fact that $P^{\mathcal{L}}(E, n+1) = P^{\mathcal{L}}(P^{\mathcal{L}}(E, n), n+1)$ and the definition of the operator function. We show the second claim by contradiction. Assume there exists an edge $(i, j) \in P^{\mathcal{L}^*}(E, n)$ that is not in $P^{\mathcal{L}}(E, n)$. Then by definition of the operator function we have the following:

$$\mathcal{L}^* \leq \alpha_{G[N_{G[H]}(i) \setminus N_G(j)]} + \alpha_{G[N_{G[H]}(j) \setminus N_G(i)]} < \mathcal{L},$$

resulting in $\mathcal{L}^* < \mathcal{L}$, which contradicts our assumption. Therefore the claim is proved. The third statement can easily be established using the results in parts 1 and 2 as follows:

$$P^{\mathcal{L}^*}(E, n+1) \subseteq P^{\mathcal{L}}(E, n+1) \subseteq P^{\mathcal{L}}(E, n),$$

which completes the proof. □

Based on the second property in the above lemma, a good quality lower bound solution is crucial to begin the scale-reduction phase and is more likely to produce a better reduction. It also affects the number of iterations and improves the computational effort.

Theorem 9. *Consider a simple undirected graph $G = (V, E)$ and let $B^* = (V_1^* \cup V_2^*, E^*) \subseteq G$ be the subgraph induced by the maximum vertex biclique. Let $B = (V_1' \cup V_2', E') \subset G$ be another biclique of size $\mathcal{L} = |V_1' \cup V_2'|$ that is not maximum. Then there exists a positive integer n such that:*

1. $P^{\mathcal{L}}(E, n) = P^{\mathcal{L}}(E, n-1)$;

2. $E^* \subseteq P^{\mathcal{L}}(E, n)$.

Proof. We show the first property by contradiction. Assume there is no positive integer n for which the above relation holds. Therefore $\forall n \geq 1$ we have $P^{\mathcal{L}}(E, n) \subset P^{\mathcal{L}}(E, n-1)$ and there exists one such n, say t, for which $P^{\mathcal{L}}(E, t) = \emptyset \subset P^{\mathcal{L}}(E, t-1)$. This is a contradiction. Note that there exists a known biclique B and using property 3 of bicliques and the definition of the operator function for all $(u, v) \in E' \subseteq E$ we have:

$$\alpha_{G[N_{G[E']}(u) \setminus N_G(v)]} + \alpha_{G[N_{G[E']}(v) \setminus N_G(u)]} = \alpha_{B[N(u)]} + \alpha_{B[N(v)]} = \mathcal{L}.$$

Hence, there are at least $|E'|$ edges that remain in $G[P^{\mathcal{L}}(E, n)]$ for any $n \geq 1$. This proves the claim.

To show the second part observe that $E^* = P^{\mathcal{L}^*}(E^*, n) \subseteq P^{\mathcal{L}^*}(E, n) \subseteq P^{\mathcal{L}}(E, n)$, in which the first equality is the direct result of the third property of bicliques and the other two relations follow from $E^* \subseteq E$ and Lemma 2.

□

Note that Theorem 9 states that, given a graph G and a known lower bound $\mathcal{L}$, the successive application of the recursive function on G and its subgraphs terminates after finite number of iterations and the resulting residual graph is a subgraph of G that contains the optimal solution of the original instance, namely the MVB of G. This result provides the basis for the scale-reduction algorithm that aims to remove edges whose elimination does not change the structure of the optimal solution. The scale-reduction algorithm proceeds with checking the operator function condition and keeping track of edges that are eligible for elimination. All such edges are removed and the edge list is updated before proceeding to the next iteration.

Algorithm 3 Reduction procedure.

Input: $G = (V, E)$, $\mathcal{L}$: a lower bound on the size of MVB
Set $E' \leftarrow E$
repeat
    Set $E^* \leftarrow E'$
    Set $E' \leftarrow \emptyset$
    for all $(u, v) \in E^*$ **do**
        $E' = E' \cup \{P_G^{\mathcal{L}}(E^*, (u, v))\}$ ▷ see Sec. 4.2.1
    end for
    if $|E'| = |E^*|$ **then**
        return E^*;
    end if
until $|E'| < |E^*|$

The algorithm terminates when no further reduction is possible. The steps of the reduction algorithm are illustrated in Alg. 3. This may significantly reduce the size of the graph especially in low-density instances making it more affordable for the exact methods to solve large-scale instances to optimality. To compute the independence numbers used to check the operator function condition, a well-known [79] algorithm for the maximum clique problem has been used, which can easily be adopted to solve the maximum independent set problem.

The mathematical programming formulation for the maximum biclique problem has a nonconvex feasible region. A general solution approach for such problems is the branch-and-bound method which is computationally demanding for large scale instances, thus making it necessary to use hybrid methods. We propose a hybrid exact algorithm for MVB that attempts to shrink the feasible region of a given instance, using scale-reduction technique, and solves the reduced problem either using the mathematical formulation or a combinatorial optimization approaches. The overall structure of the proposed hybrid exact algorithm consists of the following steps.

1. *Find lower bound.* Use a heuristic algorithm to obtain a lower bound on the optimal solution.

2. *Apply scale-reduction.* Given the lower bound solution, apply the scale-reduction technique iteratively until no further reduction is possible.

3. *Preprocess.* Update the residual graph and add valid inequalities.

4. *Solve using exact method.* Solve the MVB on residual graph using an exact method.

The procedure starts with finding a lower bound solution using a heuristic algorithm described in section 4.2.2. Given this lower bound, the scale-reduction algorithm recursively checks and removes those edges that cannot be part of the optimal solution. Then the residual graph is extracted and if possible, valid inequalities are added to the binary program which is derived from the updated residual graph. Lastly, this binary program is solved using a standard method in the literature. Alternatively, combinatorial branch-and-bound methods can be used to solve MVB after the scale-reduction step. One such algorithm has been proposed in section 4.3.

4.2.2 Initial feasible solution

The scale-reduction algorithm employs a lower bound, namely $\mathcal{L}$, on the size of MVB to initiate the reduction process. To obtain a lower bound we propose a greedy algorithm, GBICLIQUE, that aims at finding large star graphs based on the closed neighborhood of each vertex. These structures are highly unbalanced bicliques and provide a lower bound on the optimal solution. The steps of the greedy algorithm are outlined in Alg. 4.

The vertices in G are sorted based on non-increasing order of their degree in the initialization step and the subgraph induced by the closed neighborhood of each

Algorithm 4 Construction phase algorithm: GBICLIQUE

```
1: Initialization: Sort vertices based on non-increasing order of their degree in G; 𝓛 ← 0;
2: for i ← 1 to |V| do
3:     if |N[i]| > 𝓛 then
4:         X_i = X_i ∪ {i};
5:         Sort vertices in N_i based on increasing order of their degree in G[N[i]];
6:         for all j ∈ N(i) do
7:             if |N_{G[N[i]]}(j)| = 1 then
8:                 X_i = X_i ∪ {j};
9:                 N(i) = N(i) \ {j};
10:            end if
11:        end for
12:        while |N(i)| > 0 do
13:            p ← the least degree vertex in N(i);
14:            if p ∉ N(k)  (∀k ∈ {X_i \ {i}}) then
15:                X_i = X_i ∪ {p};
16:                N(i) = N(i) \ N(p);
17:            end if
18:            N(i) = N(i) \ {p};
19:        end while
20:        if |X_i| > 𝓛 then
21:            𝓛 ← |X_i|;
22:        end if
23:    end if
24: end for
25: return 𝓛;
```

vertex $i \in G$ is then considered. This vertex and all its neighbors that have degree one in $G[N[i]]$ are added to the solution, represented by X_i. The neighborhood list, $N(i)$, is updated and the algorithm proceeds with the next vertex in the neighborhood, say p, having the minimum degree in $G[N[i]]$. If vertex p is not adjacent to any of the neighbors of i in the current partial solution, it is added to the solution and removed from $N(i)$ along with all its neighbors. Otherwise only vertex p is removed and the next vertex in $N(i)$ is considered. The above procedure continues until $N(i)$ gets empty and incumbent, $\mathcal{L}$, is updated if $|X_i| > \mathcal{L}$. At each iteration i, the number of vertices in $N[i]$ is compared against the cardinality of the incumbent solution and the algorithm proceeds with that vertex only if $|N[i]| > \mathcal{L}$.

4.2.3 Preprocessing and valid inequalities

Termination of the scale-reduction algorithm results into a residual graph that contains two types of vertices. The first group, having degree zero, have lost all their incident edges during the scale-reduction and thus can be removed from the residual graph. The second group of vertices have positive degree although some of their incident edges might have been removed. Obviously, for this set of vertices all the removed edges should be added back to the residual graph before solving the MVB using an exact method. Although this might seem as a drawback, the fact that this edge cannot be part of the optimal solution, confirmed by scale-reduction algorithm, can be used to develop a tighter relaxation for the binary program.

Proposition 1. *Consider a graph $G = (V, E)$ and an edge (i, k) that has been removed from G during the scale-reduction algorithm. Assume that both vertices i and k have positive degrees in the residual graph G' at the end of the scale-reduction phase. Let $P(G')$ be the convex hull of the vectors x satisfying constraints* (4.2)- (4.6)*. Then the following inequality is valid for $P(G')$*

$$x_{i1} + x_{i2} + x_{k1} + x_{k2} \leq 1 \tag{4.16}$$

Proof. During the scale-reduction phase, it has been confirmed that edge (i, k) is not included in the optimal solution. Therefore at most one of the two vertices i and k can belong to the solution and since each vertex can only be in one bipartition, it implies that $x_{i1} + x_{i2} + x_{k1} + x_{k2} \leq 1$. Note that any arbitrary feasible solution $x \in P(G')$ including at most one of i and j satisfies (4.16). Therefore (4.16) is valid for $P(G')$. □

In the preprocessing step and after removing all vertices with degree zero from

G', for all edges whose incident vertices satisfy the conditions of Proposition 1, valid inequality (4.16) is generated and added to the binary program.

4.3 Combinatorial branch-and-bound method

Network topology is an important factor in the effectiveness of an algorithm. In some applications where the underlying network is bipartite or a dense general simple graph, the scale-reduction approach may not be effective. Solving the maximum edge biclique problem on bipartite graphs is one example and needs to be addressed using other solution approaches. In the following section we propose a combinatorial branch-and-bound algorithm for solving the maximum vertex biclique and maximum edge biclique problems, MVB and MEB, as well as their weighted versions. We also give an extension for solving the non-induced version of MVB. The general framework of algorithm has been adapted from the idea behind the method for solving the maximum clique problem in [28, 79]. We propose the algorithm in its general form for the maximum vertex weighted biclique problem. Obviously, setting all the weights to unit would solve for MVB.

4.3.1 General framework

Consider a graph $G = (V, E)$, a weight function $\mathcal{W}$ that assigns a positive weight w to each $v \in V$. The proposed branch-and-bound algorithm (B&B), illustrated in Alg. 5, solves the maximum vertex weight biclique problem. The algorithm proceeds with a simple list ordering of vertices $V = \{v_1, v_2, ..., v_n\}$ and at iteration i considers a partial ordered set of vertices $S_i = \{v_i, v_{i+1}, ..., v_n\}$ to construct the candidate set C. Since every biclique is a 2-club, only the two-neighborhood of vertex v_i in a subgraph induced by S_i, namely $N^2_{G[S_i]}(v_i)$, will be included in the candidate set. Then vertex v_i will be placed in the current partial solution set, P, and IsBiclique procedure is called. Two pruning conditions have been designed in this procedure.

First, if sum of weights of the vertices remaining in the current candidate set C, and total weight of vertices in P is less than that of the best solution found so far, continuing this branch any further would not improve the solution and it is pruned. Second, assume that in the current branch vertex $v_j \in C$ is being investigated and let $c(j)$ be the largest vertex weight biclique found when the algorithm was searching set C developed based on $N^2_{G[S_j]}(v_j)$ at some previous iteration. If the value of this solution, $c(j)$, plus the total weight of the vertices in P is less than the current best solution, this branch is pruned. If these two pruning conditions are not satisfied, v_j is added to the current partial solution set and removed from the candidate set. Then, candidate set is updated to make sure all the remaining vertices form a feasible solution with the ones already in the partial solution and the procedure IsBiclique recalls itself with the updated inputs. This recursive procedure is terminated when the candidate set is empty and the incumbent is updated only if a better solution is found. Then algorithm continues in the next iteration with S_{i-1} to generate the new candidate set.

Two important issues need to be addressed. First, at iteration i, the order in which vertices in the candidate set are investigated in IsBiclique procedure is important. All direct neighbors of vertex v_i should be considered prior to its distance-two neighbors in C. This prevents a premature pruning in the initial steps and improves the solution size. Second, to update the candidate set in Alg. 5, Biclique verification procedure is called, which takes a partial solution set P' and a vertex v as input. All vertices in P' that share an edge with v are placed in set L and the others, including v, belong to set R. The next step involves checking each of the sets R and L for being an independent set and if this condition is satisfied, a complete set of edges must exist between vertices in two sets. In case of success, vertex v will be a member of the updated candidate set. This procedure, Alg. 6, has a quadratic running time

Algorithm 5 Maximum vertex weight biclique algorithm

```
1: procedure MaxBiclique(G)
2: Order(V)
3: max=0
4: for i := n downto 1 do
5:       C := {v ∈ S_i \ {v_i} : v ∈ N^2_{G[S_i]}(v_i)}
6:       IsBiclique(C, {v_i})
7:       c(i) := max
8: end for
9: return  max;
10: procedure IsBiclique(C,P)
11: if C = ∅ then
12:       if w(P) > max then
13:             max := w(P)
14:       end if
15:       return
16: end if
17: while C ≠ ∅ do
18:       if w(C) + w(P) < max then
19:             return
20:       end if
21:       j := min{k : v_k ∈ C}
22:       if c(j) + w(P) < max then
23:             return
24:       end if
25:       C := C \ {v_j}
26:       P' := P ∪ {v_j}
27:       C' := {v ∈ C : Biclique(v, P')}
28:       IsBiclique(C', P')
29: end while
```

complexity which is a function of $|P'|$.

To improve the performance of the algorithm, the partial solution set P', at iteration i, is divided into two lists R and L that are maintained and updated throughout the iteration while recursive calls to IsBiclique are being executed. A vertex v is admitted to the solution set, feasibility check, if it shares an edge with all the vertices in one of these lists and no edge with vertices in the other one. Note that this is when the pruning conditions are not satisfied and therefore the algorithm decides to continue that branch further down by admitting v to the solution provided that the solution remains feasible. At any point during the iteration if a branch is

Algorithm 6 Biclique verification

```
1: Initialization:R ← ∅, L ← ∅
2: procedure Biclique(v, P′)
3: R ← R ∪ {v}
4: for s ∈ P′ do
5:       if (s, v)∈ E(G) then
6:             L ← L ∪ {s}
7:       else
8:             R ← R ∪ {s}
9:       end if
10: end for
11: if |L| = 0 then
12:       return 0;
13: end if
14: for u, s ∈ R do
15:       if (u, s)∈ E(G) then
16:             return 0;
17:       end if
18: end for
19: for u, s ∈ L do
20:       if (u, s)∈ E(G) then
21:             return 0;
22:       end if
23: end for
24: for u ∈ R, s ∈ L do
25:       if (u, s)∉ E(G) then
26:             return 0;
27:       end if
28: end for
29: return 1;
```

pruned, these lists that contain the partial solution are updated based on the level of B&B tree at which pruning condition has been satisfied. This method increases the overall efficiency of the B&B algorithm by eliminating the need for unnecessary computations of the candidate set at every call to Biclique procedure.

4.3.2 Non-induced MVB

The B&B algorithm can be modified to solve the *maximum vertex non-induced biclique* (MVNB) problem. Note that for the non-induced case the two bipartitions are not necessarily an independent set, requiring a change in the verification procedure as illustrated in Alg. 7. The procedure starts with placing vertex v and all

Algorithm 7 Non-induced biclique verification

```
1: Initialization:R ← ∅
2: procedure Non-induced Biclique(v, P′)
3: R ← R ∪ {v}
4: for s ∈ P′ do
5:       if (s, v)∉ E(G) then
6:             R ← R ∪ {s}
7:       end if
8: end for
9: P′ ← P′ \ R
10: repeat
11:       bool ← false
12:       for s ∈ P′ do
13:             for u ∈ R do
14:                   if (s, u)∉ E(G) then
15:                         R ← R ∪ {s}
16:                         P′ ← P′ \ {s}
17:                         bool ← true
18:                         break
19:                   end if
20:             end for
21:             if bool then
22:                   break
23:             end if
24:       end for
25: until bool
26: if |P′| = 0 then
27:       return 0;
28: end if
29: return 1;
```

vertices in P' that are not adjacent to v in partition R and removing them from P'. Next, all the remaining vertices in P' are checked to be adjacent with current vertices in R. If at least one edge is missing between vertex $s \in P'$ and vertices in R, then s cannot belong to P' and will be placed in R. Set P' is updated and the above procedure is continued until no additional vertex can be added to R or $|P'|$=0. Then, if $|P'| > 0$ the algorithm returns success with sets R and P' being the two partitions of the non-induced biclique. As in the previous case, the algorithm has been designed to solve the weighted version which is a more general case.

4.4 Computational experiments

The purpose of these experiments is to evaluate the performance of the proposed methods in terms of the solution quality and computational effort. In addition, for the scale-reduction algorithm we are specifically interested in observing the ability of the algorithm in reducing the size of the instances and the time required for that. All of the proposed algorithms were implemented in C++. To solve the mathematical model derived from the residual graph in the scale-reduction algorithm, IBM ILOG CPLEX OPTIMIZER 12.1® has been employed and default settings were used for preprocessing, branching strategies and cutting planes. The numerical experiments were conducted on Dell workstation with Intel 2.4 GHz dual quad-core Xeon E5620 processor and 12.00 GB RAM. Three different sets of instances were considered. The first set includes some of the test instances from Stanford Large Network Dataset Collection, SNAP, and contains both directed and undirected real life instances [95] from collaboration networks, peer-to-peer file sharing and route networks. For our purpose, some of the directed instances have been converted to undirected instances and used in the experiments. The second and third group of test instances are from the second and tenth DIMACS implementation challenges [38, 39]. Computational experiments are reported in Tables 4.1-4.5. All cases in which the optimal solution was found, are shown in bold.

Table 4.1 shows the results for the scale-reduction algorithm on instances from SNAP and DIMACS clustering challenge. In addition to the instance size and lower bound solution cardinality, the size of the residual graph $G' = (V', E')$ after scale-reduction and the time required for reduction, SR-CPU, are illustrated. The last two columns represent the solution size and total CPU time required to solve the problem including the scale-reduction phase. In all cases that the solution size is reported, it

represents the optimal solution to the problem and only for one instance the solution size is not reported due to the huge size of the residual graph that resulted in reaching the memory limit when loading the mathematical model. This indicates that MIP solvers cannot be used to tackle these large-scale instances directly. Although we did not consider a time limit, in 27 cases out of 37 instances solved to optimality, the solution was obtained in less than an hour. Comparing the size of the original and residual graphs in each of the test cases, the proposed scale-reduction technique is successful in shrinking the instance size by specifying and eliminating those edges that cannot belong to the optimal solution of the problem. Although for this class of algorithms the magnitude of reduction and the required computational time depend on the size and density of the graph, it seems that topology of instance is another important factor. Based on our numerical experiments, those instances that have one or multiple large star subgraphs, or structures that are close to stars where numerous branches emanate from few central vertices with some edges between vertices in different branches, are harder to reduce and need more computational effort. This is due to the fact that vertices incident to many of the edges in these subgraphs induce independent sets of large cardinality. Also the central vertices in such subgraphs have large neighborhoods and appear in many of the subproblems solved to obtain the independence number, resulting an increase in the computational time.

Results for the branch-and-bound algorithm tested on instances from well-known DIMACS clique challenge are illustrated in Table 4.2. These graphs have densities in the range [0.5, 0.99] and vary in their size from 28 to 3000 vertices and up to 4.6 million edges. The time limit of 9 hours is considered and the best solution found is reported. A total of 40 instances tried in this category, from which all but 7 were solved to optimality. Among these, 31 instances were solved in less than an hour. For those instances not solved to optimality by the time limit, the best solution found

and level at which the algorithm was terminated have been reported. Level, indicates the number of vertices, in the list ordering, whose distance-two neighborhood was investigated, according to the procedure explained in section 4.3.1, before the time limit. As an example, instance *keller5* was terminated at level 340 which implies that by the time limit, distance-two neighborhood of all vertices within the range 1 to 340 were considered and the best solution, 30, was obtained. Experiments show that the performance of the B&B algorithm is in direct relation with the density of an instance. Instances with high density are easier to solve and obviously the size of MVB is small in such cases due to the fact that the size of independent sets in dense graphs is rather small. Therefore the pruning conditions in the B&B algorithm are satisfied with higher frequency, due to the structural requirements of bicliques, resulting into a better running time for dense instances. Moreover, the optimal solution size decreases as density increases which can be observed by comparing *p-hat300* and *p-hat500* family of instances.

Because of the structural properties of non-induced bicliques, the proposed scale-reduction technique is not applicable for solving the *maximum vertex non-induced biclique*. Therefore experiments for MVNB were conducted using the branch-and-bound algorithm, Sec. 4.3.2, considering a one hour time limit and the results are reported in Table 4.3. First note that the MVNB problem is a generalization of the *maximum vertex biclique* and the *maximum clique* (MC) problems and the optimal solution to these problems is a lower bound on the size of the optimal solution for MVNB. Based on our experiments, the optimal solution size for MVNB is significantly larger than that of MVB and MC problems. This can be verified using Tables 4.1-4.3 and also with the *clique number* of the DIMACS instances used in the experiments available in [38]. Second, in all instances tried the time required to solve MVNB is significantly higher than the MVB. This is a result of more struc-

tural freedom inside the bipartitions for non-induced bicliques as compared to the bicliques and has a direct impact on the size and frequency of non-induced bicliques found at each iteration of the algorithm. As a result, the pruning conditions and the *non-induced biclique* verification procedure are influenced, leading to an increase in the CPU time. Interestingly, results suggest that the optimal solution size and the computational effort for solving the problem have direct relation with the density of an instance, which is the opposite of what was observed earlier for the MVB problem.

Table 4.4 is a comparision between the two proposed algorithms on relatively sparse instances. For each instance, the scale-reduction algorithm was permitted to run without any time limit until an optimal solution is obtained and the time limit considered for the branch-and-bound method is at least as long as the time required by the scale-reduction to solve the instance. The result shows that the scale-reduction algorithm is superior to the B&B method in almost all cases tried. Also note that scale-reduction technique is more effective for large low-density networks while the B&B method is suitable for dense graphs which can be inferred using Tables 4.1, 4.2 and 4.4.

Table 4.5 represents the solution to MEB and MVB solved using B&B algorithm. In addition to optimal solution and CPU time, the cardinality of each bipartition at optimum is reported. For some instances like *brock200-3*, *san200-0.9-2* and *hamming10-2* the optimal solution to MEB and MVB are the same that also serves as the solution to MBB which is easy to infer. These are mostly dense instances from DIMACS. On the other hand, large sparse networks like *email* and all other instances afterwards tend to have highly unbalanced bicliques that serve as the solution to MVB and MEB.

Table 4.1: Computational results using scale-reduction algorithm for MVB on instances from DIMACS Clustering challenge and SNAP dataset

Graph	**|V|**	**|E|**	**LB**	**|V′|**	**|E′|**	**SR-CPU(s)**	**Soln**	**Total-CPU(s)**
jazz	198	2742	18	85	704	1.21	**20**	2.16
email	1133	5451	34	35	35	1.31	**34**	1.74
netscience	1589	2742	16	26	42	0.20	**16**	0.53
add20	2395	7462	30	68	186	891.48	**30**	892.14
data	2851	15093	8	41	99	4.06	**8**	5.72
as19971108	3015	5347	540	584	765	2025.4	**540**	2263.73
add32	4960	9462	16	66	108	0.998	**17**	5.66
CA-GrQC	5241	14484	29	35	41	1.7	**29**	2.12
as19991204	6296	12830	1294	1407	2234	3529.44	**1294**	7139.04
p2p-Gnutella08	6301	20777	88	88	87	4.41	**88**	6.19
as20000102	6474	12572	1338	1454	2430	1475.89	**1340**	5127.27
p2p-Gnutella09	8114	26013	98	98	97	4.11	**98**	6.31
hep-th	8361	15751	22	81	151	2.94	**23**	4.41
p2p-Gnutella06	8717	31525	104	113	121	2.46	**104**	4.94
p2p-Gnutella05	8846	31839	87	88	88	3.60	**87**	4.85
CA-HepTH	9877	25973	32	43	59	7.41	**32**	7.83
PGPgiantcompo	10680	24316	105	114	122	4551.9	**105**	4555.21
p2p-Gnutella04	10876	39994	97	100	102	3.07	**97**	5.72
oregon1-010519	11050	22723	2203	2384	4289	1550.44	**2207**	16920.5
oregon1-010526	11173	23408	2199	2385	4308	1719.27	**2203**	14454.2
oregon2-010526	11460	16365	2230	2428	4676	1982.54	**2234**	19477.9
CA-HepPh	12006	118489	50	186	1600	28511.1	**50**	28520.4
cond-mat	16726	47594	41	149	419	5904.39	**41**	5912.61
p2p-Gnutella25	22687	54705	64	66	67	9.43	**64**	10.64
as-22july06	22963	48436	2243	2387	4125	2746.9	**2245**	17060.6
Ca-condmat	23133	93439	77	2024	13840	81753	-	-
p2p-Gnutella24	26518	65369	304	356	426	1052.04	**304**	1090.12
p2p-Gnutella30	36682	88328	54	54	53	20.18	**54**	21.34
Email-enron	36692	183831	1276	1384	1831	54601.4	**1276**	57089.1
p2p-Gnutella31	62586	147892	90	95	100	56.63	**90**	60.06
delaunay-n14	16384	49122	9	26	39	12.55	**9**	13.06
delaunay-n16	65536	196575	9	18	34	137.81	**9**	141.69
delaunay-n17	131072	393176	9	67	123	402.26	**10**	421.09
delaunay-n18	262144	786396	11	65	123	890.08	**12**	964.44
delaunay-n19	524288	1572823	11	54	92	3145.98	**11**	3459.2
delaunay-n20	1048576	3145686	12	24	45	6997.9	**13**	8320.37
delaunay-n21	2097152	6291408	12	48	67	26658.3	**13**	33631.8

Table 4.2: Computational results using pure B&B algorithm for MVB on instances from DIMACS Clique challenge

Graph	**|V|**	**|E|**	**density**	**Soln**	**CPU(s)**	**Level**
johnson8-2-4	28	210	0.55	**7**	0.00	-
johnson8-4-4	70	1855	0.77	**10**	0.17	-
johnson16-2-4	120	5460	0.76	**15**	21.70	-
johnson32-2-4	496	107880	0.88	$\geq$24	32400	282
keller4	171	9435	0.65	**19**	52.64	-
keller5	776	225990	0.75	$\geq$ 30	32400	340
brock200-1	200	14834	0.75	**11**	14.16	-
brock200-2	200	9876	0.5	**12**	19.58	-
brock200-3	200	12048	0.6	**12**	23.96	-
brock400-1	400	59723	0.75	**13**	526.35	-
brock800-1	800	207505	0.65	**15**	40647	-
hamming6-2	64	1824	0.9	**4**	0.05	-
hamming6-4	64	704	0.35	**14**	0.08	-
hamming8-2	256	31616	0.97	**4**	1.00	-
hamming8-4	256	20864	0.64	**32**	9419.15	-
hamming10-2	1024	518656	0.99	**4**	97.1	-
hamming10-4	1024	434176	0.83	$\geq$34	32400	306
c-fat200-1	200	1534	0.07	**3**	0.16	-
c-fat200-5	200	8473	0.42	**3**	0.33	-
c-fat500-1	500	4459	0.035	**3**	0.31	-
c-fat500-5	500	23191	0.19	**3**	0.98	-
c-fat500-10	500	46627	0.37	**3**	5.44	-
p-hat300-1	300	10933	0.24	**25**	270.81	-
p-hat300-3	300	33390	0.75	**13**	108.84	-
p-hat500-1	500	31569	0.25	**33**	35449.60	-
p-hat500-3	500	93800	0.75	**13**	1614.63	-
p-hat700-1	700	60999	0.25	$\geq$33	32400	502
p-hat700-2	700	121728	0.5	$\geq$32	32400	505
p-hat700-3	700	183010	0.75	**14**	10068.76	-
p-hat1000-3	1000	371746	0.75	$\geq$14	32400	851
p-hat1500-3	1500	847244	0.75	$\geq$14	32400	866
san200-0.7-1	200	13930	0.7	**15**	7.52	-
san200-0.7-2	200	13930	0.7	**24**	6.29	-
san200-0.9-1	200	17910	0.9	**8**	1.51	-
san200-0.9-2	200	17910	0.9	**8**	1.78	-
san400-0.5-1	400	39900	0.5	**62**	40.17	-
san400-0.7-1	400	55860	0.7	**20**	274.19	-
san400-0.9-1	400	71820	0.9	**10**	35.16	-
sanr400-0.5	400	39984	0.5	**14**	1016.43	-
sanr400-0.7	400	55869	0.7	**14**	766.69	-

Table 4.3: Computational results using pure B&B algorithm for non-induced MVB on instances from SNAP dataset and DIMACS Clique and Clustering challenges

Graph	**|V|**	**|E|**	**density**	**Soln**	**CPU(s)**	**Level**
jazz	198	2742	0.14	**93**	117.06	-
email	1133	5451	0.008	**72**	3.42	-
netscience	1589	2742	0.002	**35**	0.59	-
add20	2395	7462	0.002	**124**	75.46	-
data	2851	15093	0.003	**18**	1.95	-
as19971108	3015	5347	0.001	537	3600	2379
add32	4960	9462	0.0007	**32**	1.97	-
CA-GrQC	5241	14484	0.001	**79**	4.2	-
p2p-Gnutella08	6301	20777	0.001	**97**	17.66	-
p2p-Gnutella09	8114	26013	0.0007	**100**	15.58	-
hep-th	8361	15751	0.0004	**51**	3.65	-
p2p-Gnutella06	8717	31525	0.0008	67	3600	7491
CA-HepTH	9877	25973	0.0005	**65**	12.82	-
johnson8-2-4	28	210	0.55	**16**	0.19	-
johnson8-4-4	70	1855	0.77	**54**	1144.4	-
keller4	171	9435	0.65	68	3600	86
brock200-1	200	14834	0.75	58	3600	73
brock200-2	200	9876	0.5	58	3600	101
brock200-3	200	12048	0.6	60	3600	86
hamming6-2	64	1824	0.9	**58**	97.41	-
hamming6-4	64	704	0.35	**23**	0.87	-
hamming8-2	256	31616	0.97	106	3600	111
c-fat200-1	200	1534	0.07	**18**	0.62	-
c-fat200-2	200	3235	0.16	**35**	75.11	-
c-fat200-5	200	8473	0.42	50	3600	114
c-fat500-1	500	4459	0.035	**21**	3.04	-
c-fat500-2	500	9139	0.073	**39**	795.67	-
c-fat500-5	500	23191	0.19	48	3600	243
p-hat300-1	300	10933	0.24	64	3600	173
p-hat300-2	300	21928	0.5	63	3600	87
san200-0.7-1	200	13930	0.7	58	3600	78
san200-0.7-2	200	13930	0.7	72	3600	86
san200-0.9-1	200	17910	0.9	127	3600	132
san200-0.9-2	200	17910	0.9	99	3600	104
san200-0.9-3	200	17910	0.9	96	3600	100
sanr200-0.7	200	13868	0.7	58	3600	76
sanr200-0.9	200	17868	0.9	86	3600	91

Table 4.4: Comparison between scale-reduction and pure B&B algorithms for MVB

Graph	**|V|**	**|E|**	**SR**		**B&B**		
			Soln	**CPU(s)**	**Soln**	**CPU(s)**	**Level**
jazz	198	2742	**20**	2.16	**20**	168.73	-
email	1133	5451	**34**	1.74	**34**	38.59	-
add20	2395	7462	**30**	892.14	23	3600	877
data	2851	15093	**8**	5.72	**8**	1.06	-
add32	4960	9462	**17**	5.66	**17**	1.78	-
as19971108	3015	5347	**540**	2263.73	96	3600	758
CA-GrQC	5241	14484	**29**	2.12	**29**	12.63	-
as19991204	6296	12830	**1294**	7139.04	94	9041	844
hep-th	8361	15751	**23**	4.41	**23**	50.36	-
as20000102	6474	12572	**1340**	5127.27	87	5197.37	115
CA-HepTH	9877	25973	**32**	7.83	**32**	1099.61	-
PGPgiantcompo	10680	24316	**105**	4555.21	34	8194.64	3832
oregon1-010519	11050	22723	**2207**	16920.5	109	18954.21	880
oregon1-010526	11173	23408	**2203**	14454.2	105	24275.55	888
oregon2-010526	11460	16365	**2234**	19477.9	103	25862.29	901
cond-mat	16726	47594	**41**	5912.61	39	7015.77	13726

Table 4.5: Solution to MEB and MVB problems solved by pure B&B algorithm

Graph	**MEB**			**MVB**		
	Soln	**Bipartitions**	**CPU(s)**	**Soln**	**Bipartitions**	**CPU(s)**
football	**8**	(4,2)	0.03	**8**	(7,1)	0.05
brock200-2	**42**	(7,6)	14.71	**13**	(7,6)	42.7
brock200-3	**36**	(6,6)	27.33	**12**	(5,7)	55.33
san200-0.9-2	**16**	(4,4)	1.9	**8**	(4,4)	2.89
san200-0.9-3	**25**	(5,5)	1.08	**10**	(5,5)	3.18
p-hat300-1	**33**	(11,3)	55.35	**25**	(24,1)	595.14
p-hat300-2	**44**	(11,4)	152.32	**24**	(23,1)	539.31
p-hat300-3	**42**	(7,6)	166.25	**13**	(7,6)	296.1
sanr400-0.5	**49**	(7,7)	1718.52	**14**	(9,5)	4615.83
hamming10-2	**4**	(2,2)	134.27	**4**	(2,2)	119.45
email	**33**	(33,1)	32.28	**34**	(33,1)	67.17
netscience	**15**	(15,1)	0.41	**16**	(15,1)	0.77
data	**7**	(7,1)	0.67	**8**	(7,1)	0.68
add32	**15**	(15,1)	1.34	**16**	(15,1)	1.78
CA-GrQC	**28**	(28,1)	6.88	**29**	(28,1)	12.63
hep-th	**22**	(22,1)	89.43	**23**	(22,1)	50.36

5. CONTRIBUTIONS AND FUTURE RESEARCH

This dissertation focuses on two cluster-detection models from theoretical and algorithmic perspectives. First, the s-clubs that are ideal models for detecting low-diameter clusters are investigated. Graph-based data mining and robust network design serve as two application areas for s-club models. Second, biclique structures and their variations with applications in biclustering, marketing and ranking systems are studied. This chapter summarizes our contribution and provides future research directions.

5.1 Contributions

In Chapter 1, an analytical review of distance-based clique relaxations and the biclique models is provided to identify possible research areas in graph-based cluster-detection models. This research answers some of the open questions posed from theoretical and algorithmic point of view:

1. The s-clubs are nonheredity in nature and as a direct result, s-club maximality testing is an intractable problem. We developed a sufficient condition for checking maximality, by inclusion, of the s-clubs. The proposed sufficient condition can be employed in the design of algorithms for the maximum s-club problem.

2. Scalability is important in providing solution to the practical applications of the maximum s-club problem. A variable neighborhood search algorithm is proposed and implemented to enable the solution for large-scale instances of the problem. In addition to incorporating the sufficient condition for checking maximality, a new construction phase heuristic, multiple neighborhood struc-

tures and a local improvement procedure using k-add moves are embedded in the design of VNS algorithm.

3. A hybrid exact algorithm for the maximum s-club problem has been proposed to investigate the effect of initial starting solution, obtained using VNS, on the performance of an existing combinatorial branch-and-bound method. Extensive computational experiments for VNS and the hybrid algorithms and comparison to other existing methods are reported.

4. Asymptotic lower and upper bounds are established for three classes of the maximum biclique problem on uniform random graphs. These bounds are insightful in understanding the structure and size of the bicliques in large networks. Bounds also give clear indication about the evolution of biclique structures in underlying graphs especially protein interaction and gene networks. For example, it was observed that the solution to the *maximum edge biclique problem* is of order $O((\log n)^2)$ implying the relatively slow growth rate of MEB with respect to the size of the containing network.

5. A scale-reduction algorithm is proposed for solving two important classes of the maximum biclique problem, MVB and MEB, in large-scale sparse networks. This algorithm enables the exact solution to these problems for practical applications where the underlying network is large and commercial solvers cannot be used directly.

6. A combinatorial branch-and-bound algorithm is developed for solving different variations of the maximum biclique problem. This algorithm is suitable for attempting dense instances of the problem where scale-reduction approaches are less effective due to homogeneous nature of vertex degrees. The algorithm has

been implemented for MVB and MEB and can be easily modified to solve the weighted versions of these problems. Computational experiments and analytical results have been reported for both scale-reduction and branch-and-bound methods.

5.2 Future research

For distance-based clique-relaxations, an interesting research question is to investigate the complexity of maximum 2-clique and 2-club problems in unit disk graphs. It is also interesting to identify graph classes in which every connected s-clique is an s-club. In such cases checking maximality of an s-club reduces to checking maximality of a connected s-clique and is easy. Performing a polyhedral study and developing generalized valid inequalities for s-club polytope is another research direction in this area. It would also be helpful to employ such valid inequalities in a branch-and-cut algorithm and compare the effectivness of the algorithm with other existing methods.

Many real-life networks are in fact power-law graphs. It is insightful to develop asymptotic bounds for variants of the maximum biclique problem in such networks. Another research direction is to develop algorithms and asymptotic bounds for the *maximum quasi-biclique* problem with application in finding interacting protein group pairs in protein interaction networks. The *maximum edge biclique packing* problem has been considered recently and is used to model product bundling problem in marketing. The scale-reduction and the combinatorial branch-and-bound methods proposed in this research can be adopted in a decomposition algorithm proposed for the *maximum edge biclique packing* problem.

REFERENCES

[1] J. Abello, M.G.C. Resende, and S. Sudarsky. Massive quasi-clique detection. In S. Rajsbaum, editor, *LATIN 2002: Theoretical Informatics*, pages 598–612, London, 2002. Springer-Verlag.

[2] V. Acuña, C.E. Ferreira, A.S. Freire, and E. Moreno. Solving the maximum edge biclique packing problem on unbalanced bipartite graphs. *Discrete Applied Mathematics*, 2011.

[3] N. Adler. Competition in a deregulated air transportation market. *European Journal of Operational Research*, 129:337–345, 2001.

[4] R.D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:113–126, 1973.

[5] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L. Hammer, and B. Simeone. Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145(1):11–21, 2004.

[6] M.T. Almeida and F.D. Carvalho. Integer models and upper bounds for the 3-club problem. *Networks*, 60:155–166, 2012.

[7] N. Alon and J. H. Spencer. *The Probabilistic Method, 3rd ed.* John Wiley, New Jersey, 2008.

[8] C. Ambühl, M. Mastrolilli, and O. Svensson. Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM J. Comput*, 2011.

[9] J. Amilhastre, M.C. Vilarem, and P. Janssen. Complexity of minimum biclique cover and minimum biclique decomposition for bipartite domino-free graphs. *Discrete Applied Mathematics*, 86:125–144, 1998.

[10] S. Arora, C. Lund, R. Motwani, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45:501–555, 1998.

[11] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[12] Y. Asahiro, E. Miyano, and K. Samizo. Approximating maximum diameter-bounded subgraphs. In *Proceedings of the 9th Latin American conference on Theoretical Informatics (LATIN'10)*, pages 615–626. Springer, 2010.

[13] C. Avanthay, A. Hertz, and N. Zufferey. A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151:379–388, 2003.

[14] J. S. Bader, A. Chaudhuri, J. M. Rothberg, and J. Chant. Gaining confidence in high-throughput protein interaction networks. *Nature Biotechnology*, 22:78–85, 2004.

[15] B. Balasundaram and S. Butenko. Graph domination, coloring and cliques in telecommunications. In M. G. C. Resende and P. M. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 865–890. Spinger Science + Business Media, New York, 2006.

[16] B. Balasundaram, S. Butenko, and S. Trukhanov. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10:23–39, 2005.

[17] V. Batagelj. Networks/Pajek Graph Files, 2005. http://vlado.fmf.uni-lj.si/pub/networks/pajek/data/gphs.htm. Accessed February 2013.

[18] N. Berry, T. Ko, T. Moy, J. Smrcka, J. Turnley, and B. Wu. Emergent clique formation in terrorist recruitment. *The AAAI-04 Workshop on Agent Organizations: Theory and Practice, July 25, 2004, San Jose, California*, 2004. Online: http://www.cs.uu.nl/ virginia/aotp/papers.htm.

[19] D. Binkele-Raible, H. Fernau, S. Gaspers, and M. Liedloff. Exact exponential-time algorithms for finding bicliques. *Information Processing Letters*, 111(2):64–67, 2010.

[20] B. Bollobás. *Extremal Graph Theory.* Academic Press, New York, 1978.

[21] B. Bollobás. *Random Graphs.* Academic Press, New York, 1985.

[22] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 1–74, Dordrecht, The Netherlands, 1999. Kluwer Academic Publishers.

[23] J.-M. Bourjolly, G. Laporte, and G. Pesant. Heuristics for finding k-clubs in an undirected graph. *Computers & Operations Research*, 27:559–569, 2000.

[24] J.-M. Bourjolly, G. Laporte, and G. Pesant. An exact algorithm for the maximum k-club problem in an undirected graph. *European Journal Of Operational Research*, 138:21–28, 2002.

[25] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[26] A. Buchanan, J. S. Sung, S. Butenko, V. Boginski, and E. Pasiliao. On connected dominating sets of restricted diameter. Working paper, 2012.

[27] S. Butenko and W. Wilhelm. Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research*, 173:1–17, 2006.

[28] R. Carraghan and P. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9:375–382, 1990.

[29] F. D. Carvalho and M. T. Almeida. Upper bounds and heuristics for the 2-club problem. *European Journal of Operational Research*, 210(3):489 – 494, 2011.

[30] M.S. Chang, L.J. Hung, C.R. Lin, and P.C. Su. Finding large k-clubs in undirected graphs. In *Proc. 28th Workshop on Combinatorial Mathematics and Computation Theory*. 2011.

[31] J. Chen, X. Huang, I. A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72:1346–1367, 2006.

[32] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, pages 93–100, 2000.

[33] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[34] M. Dawande, P. Keskinocak, J. Swaminathan, and S. Tayur. On bipartite and multipartite clique problems. *Journal of Algorithms*, 41:388–403, 2001.

[35] A. Dekker, H. Pérez-Rosés, G. Pineda-Villavicencio, and Watters P. The maximum degree & diameter-bounded subgraph and its applications. *Journal of Mathematical Modelling and Algorithms*, 11:249–268, 2012.

[36] A. H. Dekker. Social network analysis in military headquarters using CAVALIER. In *Proceedings of Fifth International Command and Control Research and Technology Symposium*, pages 24–26, Canberra, Australia, 2000.

[37] R. Diestel. *Graph Theory*. Springer-Verlag, Berlin, 1997.

[38] DIMACS. Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1992. http://dimacs.rutgers.edu/Challenges/. Accessed November 2012.

[39] DIMACS. Graph Partitioning and Graph Clustering: Tenth DIMACS Implementation Challenge, 2012. http://cc.gatech.edu/dimacs10/. Accessed November 2012.

[40] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[41] U. Feige. Relations between average case complexity and approximation complexity (extended abstract). In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 534–543, 2002.

[42] U. Feige and S. Kogan. Hardness of approximation of the balanced complete bipartite subgraph problem. Technical report, 2004.

[43] J. Gagneur, R. Krause, T. Bouwmeester, and G. Casari. Modular decomposition of protein-protein interaction networks. *Genome Biology*, 5(8):R57.1–R57.12, 2004.

[44] B. Ganter and R. Wille. *Formale Begriffsanalyse-Mathematische Grundlagen*. Springer-Verlag, New York, 1996.

[45] F. García-López, B. Melián-Batista, J.A. Moreno-Pérez, and J.M. Moreno-Vega. The parallel variable neighborhood search for the p-median problem. *Journal of Heuristics*, 8:375–388, 2002.

[46] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

[47] H. Gavranovic. Local search and suffix tree for car-sequencing problem with colors. *European Journal of Operational Research*, 191(3):972–980, 2008.

[48] A. Gély, L. Nourine, and B. Sadi. Enumeration aspects of maximal cliques and bicliques. *Discrete Applied Mathematics*, 157(7):1447–1459, 2009.

[49] M. Gendreau, P. Soriano, and L. Salvail. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, 41:385–403, 1993.

[50] F.W. Glover and G.A. Kochenberger, editors. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston, MA, 2003.

[51] C. Goffman. And what is your Erdös number? *American Mathematical Monthly*, 76:791, 1969.

[52] J. Grossman, P. Ion, and R. De Castro. The Erdös Number Project. http://web.archive.org/web/20080207010024/, 1995. Accessed February 2013.

[53] W. H. Haemers. Bicliques and eigenvalues. *Journal of Combinatorial Theory, Series B*, 82(1):56 – 66, 2001.

[54] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[55] P. Hansen, N. Mladenović, and D. Uroševic. Variable neighborhood search for the maximum clique. *Discrete Applied Mathematics*, 145:117–125, 2004.

[56] S. Hartung, C. Komusiewicz, and A. Nichterlein. Parameterized algorithmics and computational experiments for finding 2-clubs. In D. Thilikos and G. Woeginger, editors, *Parameterized and Exact Computation*, volume 7535 of *Lecture Notes in Computer Science*, pages 231–241. Springer, 2012.

[57] S. Hartung, C. Komusiewicz, and A. Nichterlein. On structural parameterizations for the 2-club problem. In *Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM '13)*, volume 7741 of *Lecture Notes in Computer Science*, pages 233–243. Springer, 2013.

[58] D. S. Hochbaum. Approximating clique and biclique problems. *Journal of Algorithms*, 29:174–200, 1998.

[59] D. S. Hochbaum. Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations. *European Journal of Operational Research*, 140(2):291– 321, 2002.

[60] P. Jaillet, G. Song, and G. Yu. Airline network design and hub location problems. *Location Science*, 4:195–212, 1996.

[61] H. Jeong, S. Mason, A.L. Barabási, and Z. N. Oltvai. Lethality and certainty in protein networks. *Nature*, 411:41–42, 2001.

[62] R.M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, New York, 1972.

[63] D. Kim, Y. Wu, Y. Li, F. Zou, and D.-Z. Du. Constructing minimum connected dominating sets with bounded diameters in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(2):147–157, 2009.

[64] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. In *Proceedings of the 8th international conference on World Wide Web*, pages 1481–1493, 1999.

[65] J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Comput. Oper. Res.*, 34:2743–2757, 2007.

[66] S. Lehmann, M. Schwartz, and L. K. Hansen. Biclique communities. *Physical Review E*, 78(1):016108, 2008.

[67] G. Liu, K. Sim, and J. Li. Efficient mining of large maximal bicliques. In *Data Warehousing and Knowledge Discovery*, volume 4081 of *LNCS*, pages 437–448. Springer-Verlag, 2006.

[68] R.D. Luce. Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15:169–190, 1950.

[69] R.D. Luce and A.D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14:95–116, 1949.

[70] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey, 2004. IEEE/ACM Transactions on Computational Biology and Bioinformatics.

[71] F. Mahdavi Pajouh. *Polyhedral Combinatorics, Complexity & Algorithms for k-Clubs in Graphs*. PhD thesis, Oklahoma State University, July 2012.

[72] J. Marinček and B. Mohar. On approximating the maximum diameter ratio of graphs. *Discrete Mathematics*, 244:323–330, 2002.

[73] N. Memon, K. C. Kristoffersen, D. L. Hicks, and H. L. Larsen. Detecting critical regions in covert networks: A case study of 9/11 terrorists network. In *The Second International Conference on Availability, Reliability and Security*, pages 861–870, 2007.

[74] J. Miao and D. Berleant. From paragraph networks to document networks. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2004)*, volume 1, pages 295–302, 2004.

[75] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[76] R.J. Mokken. Cliques, clubs and clans. *Quality and Quantity*, 13:161–173, 1979.

[77] M. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[78] D. Nussbaum, S. Pu, J. Sack, T. Uno, and H. Zarrabi-Zadeh. Finding maximum edge bicliques in convex bipartite graphs. *Algorithmica*, 64:311–325, 2012.

[79] P. R. J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120:197–207, 2002.

[80] F. Mahdavi Pajouh and B. Balasundaram. On inclusionwise maximal and maximum cardinality k-clubs in graphs. *Discrete Optimization*, 9(2):84–97, 2012.

[81] F. Parreño, R. Alvarez-Valdes, J. Oliveira, and J. Tamarit. Neighborhood structures for the container loading problem: a VNS implementation. *Journal of Heuristics*, 16:1–22, 2010.

[82] S. Pasupuleti. Detection of protein complexes in protein interaction networks using n-clubs. In *Proceedings of the 6th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, volume 4973 of *Lecture Notes in Computer Science*, pages 153–164. Springer, 2008.

[83] J. Pattillo, Y. Wang, and S. Butenko. On distance-based clique relaxations in unit disk graphs. Working paper, 2012.

[84] J. Pattillo, N. Youssef, and S. Butenko. On clique relaxation models in network analysis. *European Journal of Operational Research*, In press, doi: 10.1016/j.ejor.2012.10.021, 2012.

[85] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131:651–654, 2003.

[86] R. B. Rothenberg, J. J. Potterat, and D. E. Woodhouse. Personal risk taking and the spread of disease: Beyond core groups. *Journal of Infectious Diseases*, 174 (Supp. 2):S144–S149, 1996.

[87] M. Sageman. *Understanding Terrorist Networks*. University of Pennsylvania Press, Philadelphia, PA, 2004.

[88] R. J. Sampson and B. W. Groves. Community structure and crime: Testing social-disorganization theory. *American Journal of Sociology*, 94:774–802, 1989.

[89] M. J. Sanderson, A. C. Driskell, R. H. Ree, O. Eulenstein, and S. Langley. Obtaining maximal concatenated phylogenetic data sets from large sequence databases. *Molecular Biology and Evolution*, 20(7):1036–1042, 2003.

[90] A. Schäfer. Exact algorithms for s-club finding and related problems. Master's thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2009.

[91] A. Schäfer, C. Komusiewicz, H. Moser, and R. Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6:883–891, 2012.

[92] J. Scott. *Social Network Analysis: A Handbook*. Sage Publications, London, 2 edition, 2000.

[93] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

[94] S. B. Seidman and B. L. Foster. A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.

[95] SNAP. Stanford Large Network Dataset Collection, 2012. http://snap.stanford.edu/data/. Accessed November 2012.

[96] V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123–12128, 2003.

[97] J. Tan. Inapproximability of maximum weighted edge biclique and its applications. In *Proceedings of the 5th international conference on theory and application of models of computation*, pages 282–293. Springer-Verlag, 2008.

[98] L. Terveen, W. Hill, and B. Amento. Constructing, organizing, and visualizing collections of topically related web resources. *ACM Transactions on Computer-Human Interaction*, 6:67–94, 1999.

[99] A. Veremyev and V. Boginski. Identifying large robust network clusters via new compact formulations of maximum k-club problems. *European Journal of Operational Research*, 218(2):316–326, 2012.

[100] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge University Press, New York, 1994.

[101] M. Yannakakis. Node and edge-deletion NP-complete problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 253–264, New York, NY, USA, 1978. ACM.

[102] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.

APPENDIX A

PROOF OF COMPLEXITY RESULT FOR ASYMPTOTIC BOUNDS ON MAXIMUM BICLIQUE PROBLEMS

We show the correctness of the following relation used in the proof of Theorem 5. Note that the same argument can be used to show the relation used in the proof of Theorem 5. We prove the following equality:

$$\left[(1-\frac{l_2+1}{n})(1-\frac{l_2+1}{n-1})(1-\frac{l_2+1}{n-2})\cdots(1-\frac{l_2+1}{n-l_2})\right] = \left[1-\frac{(l_2+1)^2}{n}+O(n^{\frac{-3}{2}})\right].$$

Expanding the left side we get,

$$\begin{aligned}
&\left[(1-\frac{l_2+1}{n})(1-\frac{l_2+1}{n-1})(1-\frac{l_2+1}{n-2})\cdots(1-\frac{l_2+1}{n-l_2})\right] = \\
&\left[1-\frac{l_2+1}{n}-\frac{l_2+1}{n-1}-\frac{l_2+1}{n-2}-\cdots-\frac{l_2+1}{n-l_2}\right]+ \\
&\left[\frac{(l_2+1)^2}{n(n-1)}+\frac{(l_2+1)^2}{n(n-2)}+\cdots+\frac{(l_2+1)^2}{n(n-l_2)}+\frac{(l_2+1)^2}{(n-1)(n-2)}+\frac{(l_2+1)^2}{(n-1)(n-3)}+\cdots+\frac{(l_2+1)^2}{(n-1)(n-l_2)}\right]+ \\
&\left[\frac{(l_2+1)^2}{(n-2)(n-3)}+\frac{(l_2+1)^2}{(n-2)(n-4)}+\cdots+\frac{(l_2+1)^2}{(n-2)(n-l_2)}+\frac{(l_2+1)^2}{(n-3)(n-4)}+\cdots+\frac{(l_2+1)^2}{(n-3)(n-l_2)}+\cdots\right]+ \\
&\left[-\frac{(l_2+1)^3}{n(n-1)(n-2)}-\frac{(l_2+1)^3}{n(n-1)(n-3)}-\cdots-\frac{(l_2+1)^3}{n(n-1)(n-l_2)}\right]+ \\
&\left[-\frac{(l_2+1)^3}{(n-1)(n-2)(n-3)}-\cdots-\frac{(l_2+1)^3}{(n-1)(n-2)(n-l_2)}\right]+ \\
&\left[-\frac{(l_2+1)^3}{(n-2)(n-3)(n-4)}-\cdots-\frac{(l_2+1)^3}{(n-2)(n-3)(n-l_2)}-\cdots\right]+ \\
&\left[\frac{(l_2+1)^4}{n(n-1)(n-2)(n-3)}+\cdots+\frac{(l_2+1)^4}{n(n-1)(n-2)(n-l_2)}+\cdots\right]+\cdots
\end{aligned}$$

The first bracket in the above extended form can be written as follows:

$$\left[1-\frac{l_2+1}{n}-\frac{l_2+1}{n-1}-\frac{l_2+1}{n-2}-\cdots-\frac{l_2+1}{n-l_2}\right]=1-\sum_{i=0}^{l_2}\frac{l_2+1}{n-i}=1-\frac{l_2+1}{n}\sum_{i=0}^{l_2}\frac{n}{n-i}=$$
$$1-\frac{l_2+1}{n}\sum_{i=0}^{l_2}(1+\frac{i}{n-i})=1-\frac{l_2+1}{n}(l_2+1+\sum_{i=0}^{l_2}\frac{i}{n-i})=1-\frac{(l_2+1)^2}{n}-\frac{l_2+1}{n}\sum_{i=0}^{l_2}\frac{i}{n-i}=$$
$$1-\frac{(l_2+1)^2}{n}-O(n^{-2}).$$

Note that in the first part of Theorem 4 we proved that l_1+l_2 is bounded above by $\frac{16\log n}{\log\frac{1}{1-p}}$ which implies that l_2 is also bounded by the same value and we can deduce that $l_2 = O(n^{\frac{1}{2}})$. The next two brackets contain terms that are of order $O(n^{-2})$ with respect to n and all other remaining terms are of order $-O(n^{-3})$, $O(n^{-4})$, $-O(n^{-5})\cdots$, respectively. Also note that the number of terms with order $O(n^{-2})$ is larger than the number of terms with order $-O(n^{-3})$ and in general the number of terms reduces as order of n in the denominators increases. This implies that the terms in the third and fourth brackets are dominant over all others and proves our claim that

$$\left[(1-\frac{l_2+1}{n})(1-\frac{l_2+1}{n-1})(1-\frac{l_2+1}{n-2})\cdots(1-\frac{l_2+1}{n-l_2})\right]=\left[1-\frac{(l_2+1)^2}{n}+O(n^{\frac{-3}{2}})\right].$$

APPENDIX B

DETAILED RESULTS OF EXPERIMENTS FOR THE MAXIMUM s-CLUB PROBLEM

Table B.1: Results of solving the MsCP using VNS and B&B algorithms(s=2, d=0.0125)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	2	2	0.094	2	2	0	0.063
Graph-50-2	5	5	0.109	5	5	0	0.078
Graph-50-3	5	5	0.093	5	5	0	0.078
Graph-50-4	3	3	0.094	3	3	0	0.078
Graph-50-5	3	3	0.093	3	3	0	0.078
Graph-50-6	5	5	0.109	5	5	0	0.078
Graph-50-7	5	5	0.11	5	5	0	0.062
Graph-50-8	5	5	0.109	5	5	0	0.063
Graph-50-9	5	5	0.109	5	5	0	0.063
Graph-50-10	4	4	0.093	4	4	0	0.062
Graph-100-1	6	6	0.281	6	6	0	0.265
Graph-100-2	6	6	0.265	6	6	0	0.281
Graph-100-3	6	6	0.343	6	6	0	0.265
Graph-100-4	7	7	0.297	7	7	0	0.296
Graph-100-5	7	7	0.28	7	7	0	0.297
Graph-100-6	6	6	0.28	6	6	0	0.28
Graph-100-7	6	6	0.281	6	6	0	0.281
Graph-100-8	6	6	0.281	6	6	0	0.281
Graph-100-9	5	5	0.219	5	5	0	0.281
Graph-100-10	5	5	0.234	5	5	0	0.28
Graph-150-1	7	7	0.748	7	7	0	0.592
Graph-150-2	8	8	0.64	8	8	0	0.765
Graph-150-3	8	8	0.64	8	8	0	0.765
Graph-150-4	6	6	0.562	6	6	0	0.717
Graph-150-5	6	6	0.562	6	6	0	0.733
Graph-150-6	10	10	0.811	10	10	0	0.764
Graph-150-7	10	10	0.811	10	10	0	0.749
Graph-150-8	8	8	0.624	8	8	0	0.78
Graph-150-9	8	8	0.624	8	8	0	0.78
Graph-150-10	8	8	0.624	8	8	0	0.78
Graph-200-1	9	9	1.139	9	9	0	1.263
Graph-200-2	9	9	1.139	9	9	0	1.076
Graph-200-3	8	8	1.154	8	8	0	1.202
Graph-200-4	8	8	1.123	8	8	0	1.123
Graph-200-5	8	8	1.123	8	8	0	1.124
Graph-200-6	8	8	1.155	8	8	0	1.077
Graph-200-7	13	13	1.436	13	13	0	1.124
Graph-200-8	9	9	1.31	9	9	0	1.092
Graph-200-9	9	9	1.311	9	9	0	1.108
Graph-200-10	9	9	1.03	9	9	0	0.983

Table B.2: Results of solving the MsCP using VNS and B&B algorithms(s=2, d=0.025)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	5	5	0.14	5	5	0	0.078
Graph-50-2	6	6	0.249	6	6	0	0.078
Graph-50-3	5	5	0.125	5	5	0	0.062
Graph-50-4	5	5	0.14	5	5	0	0.062
Graph-50-5	4	4	0.11	4	4	0	0.063
Graph-50-6	6	6	0.25	6	6	0	0.093
Graph-50-7	6	6	0.265	6	6	0	0.093
Graph-50-8	5	5	0.14	5	5	0	0.063
Graph-50-9	5	5	0.14	5	5	0	0.078
Graph-50-10	5	5	0.141	5	5	0	0.078
Graph-100-1	7	7	0.468	7	7	0	0.328
Graph-100-2	10	10	0.608	10	10	0	0.343
Graph-100-3	7	7	0.437	7	7	0	0.328
Graph-100-4	9	9	0.718	9	9	0	0.422
Graph-100-5	9	9	0.562	9	9	0	0.327
Graph-100-6	10	10	0.687	10	10	0	0.327
Graph-100-7	10	10	0.686	10	10	0	0.328
Graph-100-8	10	10	0.671	10	10	0	0.359
Graph-100-9	10	10	0.67	10	10	0	0.359
Graph-100-10	10	10	0.671	10	10	0	0.359
Graph-150-1	12	12	1.466	12	12	0	1.201
Graph-150-2	9	9	1.216	9	9	0	1.014
Graph-150-3	11	11	1.498	11	11	0	1.092
Graph-150-4	11	11	1.482	11	11	0	1.092
Graph-150-5	11	11	1.716	11	11	0	1.326
Graph-150-6	11	11	1.42	11	11	0	1.419
Graph-150-7	11	11	1.42	11	11	0	1.42
Graph-150-8	12	12	1.653	12	12	0	1.155
Graph-150-9	12	12	1.794	12	12	0	1.248
Graph-150-10	12	12	1.794	12	12	0	1.263
Graph-200-1	13	13	2.854	13	13	0	3.214
Graph-200-2	14	14	2.621	14	14	0	3.042
Graph-200-3	13	13	2.387	13	13	0	2.808
Graph-200-4	11	11	2.293	11	11	0	2.543
Graph-200-5	14	14	3.12	14	14	0	3.214
Graph-200-6	15	15	3.105	15	15	0	3.12
Graph-200-7	15	15	3.65	15	15	0	3.837
Graph-200-8	15	15	3.136	15	15	0	2.761
Graph-200-9	13	13	2.855	13	13	0	3.213
Graph-200-10	15	15	2.964	15	15	0	2.871

Table B.3: Results of solving the MsCP using VNS and B&B algorithms(s=2, d=0.05)

Instance		VNS			B&B		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	6	6	0.249	6	6	0	0.109
Graph-50-2	7	7	0.312	7	7	0	0.109
Graph-50-3	6	6	0.312	6	6	0	0.093
Graph-50-4	8	8	0.297	8	8	0	0.078
Graph-50-5	6	6	0.218	6	6	0	0.094
Graph-50-6	7	7	0.218	7	7	0	0.109
Graph-50-7	7	7	0.219	7	7	0	0.109
Graph-50-8	6	6	0.203	6	6	0	0.093
Graph-50-9	6	6	0.202	6	6	0	0.078
Graph-50-10	6	6	0.202	6	6	0	0.078
Graph-100-1	14	14	2.262	14	14	0	1.482
Graph-100-2	13	13	1.841	13	13	0	0.998
Graph-100-3	11	11	1.7	11	11	0	1.045
Graph-100-4	11	11	1.56	11	11	0	1.014
Graph-100-5	11	11	1.575	11	11	0	0.999
Graph-100-6	13	13	1.435	13	13	0	1.108
Graph-100-7	13	13	1.435	13	13	0	1.123
Graph-100-8	13	13	2.418	13	13	0	1.092
Graph-100-9	12	12	1.529	12	12	0	1.077
Graph-100-10	12	12	1.544	12	12	0	1.077
Graph-150-1	17	17	5.117	17	17	0	5.554
Graph-150-2	16	16	3.791	16	16	0	5.834
Graph-150-3	17	17	4.134	17	17	0	4.259
Graph-150-4	18	18	5.99	18	18	0	4.851
Graph-150-5	16	16	4.244	16	16	0	5.132
Graph-150-6	21	21	5.663	21	21	0	4.134
Graph-150-7	16	16	5.023	16	16	0	6.942
Graph-150-8	18	18	5.07	18	18	0	5.179
Graph-150-9	20	20	7.207	20	20	0	6.911
Graph-150-10	22	22	8.361	22	22	0	6.177
Graph-200-1	20	20	8.19	20	20	0	17.394
Graph-200-2	21	21	11.902	21	21	0	18.392
Graph-200-3	19	19	9.813	19	19	0	24.118
Graph-200-4	22	22	10.374	22	22	0	24.648
Graph-200-5	20	20	9.438	20	20	0	27.019
Graph-200-6	21	21	8.486	21	21	0	22.589
Graph-200-7	20	20	10.046	20	20	0	26.13
Graph-200-8	22	22	11.655	22	22	0	30.311
Graph-200-9	27	27	14.021	27	27	0	15.428
Graph-200-10	21	21	9.969	21	21	0	22.527

Table B.4: Results of solving the MsCP using VNS and B&B algorithms(s=2, d=0.1)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	9	9	2.231	9	9	0	0.312
Graph-50-2	11	11	3.292	11	11	0	10.296
Graph-50-3	12	12	3.713	12	12	0	0.343
Graph-50-4	13	13	3.962	13	13	0	0.468
Graph-50-5	11	11	3.026	11	11	0	0.483
Graph-50-6	12	12	3.588	12	12	0	10.857
Graph-50-7	12	12	3.619	12	12	0	11.107
Graph-50-8	10	10	2.449	10	10	0	0.265
Graph-50-9	10	10	2.449	10	10	0	0.266
Graph-50-10	10	10	2.45	10	10	0	0.265
Graph-100-1	18	18	14.892	18	18	0	166.96
Graph-100-2	20	20	13.307	20	20	0	113.89
Graph-100-3	15	15	9.843	15	15	0	118.67
Graph-100-4	19	19	14.446	19	19	0	119.22
Graph-100-5	19	19	12.277	19	19	0	108.37
Graph-100-6	26	26	17.254	26	26	0	113.42
Graph-100-7	20	20	22.808	20	20	0	223.074
Graph-100-8	20	20	15.116	20	20	0	127.73
Graph-100-9	22	22	16.676	22	22	0	137.53
Graph-100-10	22	22	35.662	22	23	0	239.77
Graph-150-1	24	24	43.539	24	24	0	2613.967
Graph-150-2	29	29	48.048	29	29	0	1376.435
Graph-150-3	27	27	55.271	27	27	0	3165.567
Graph-150-4	26	26	43.462	26	26	0	2116.218
Graph-150-5	26	26	49.109	26	26	0	2013.024
Graph-150-6	30	30	55.287	30	30	0	1458.163
Graph-150-7	29	29	73.647	29	29	12.12	3600.199
Graph-150-8	33	33	74.459	33	33	0	2114.892
Graph-150-9	37	37	71.152	37	37	0	517.296
Graph-150-10	29	29	58.687	29	29	0	2704.166
Graph-200-1	33	33	157.278	33	33	43.1	3601.462
Graph-200-2	31	31	134.952	31	31	44.64	3601.073
Graph-200-3	35	35	136.22	35	35	38.6	3618.498
Graph-200-4	31	31	142.396	31	31	41.51	3622.179
Graph-200-5	36	36	154.923	36	36	41.94	3617.281
Graph-200-6	39	39	398.78	39	39	42.65	3630.51
Graph-200-7	45	45	245.653	45	45	30.77	3625.72
Graph-200-8	35	35	236.854	35	35	45.31	3620.511
Graph-200-9	40	40	161.772	40	40	25.93	3623.248
Graph-200-10	37	37	176.139	37	37	28.85	3610.636

Table B.5: Results of solving the MsCP using VNS and B&B algorithms(s=2, d=0.15)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	15	16	11.076	16	16	0	11.919
Graph-50-2	13	15	10.654	15	16	0	10.671
Graph-50-3	9	15	8.439	15	15	0	8.923
Graph-50-4	10	13	8.471	13	13	0	8.221
Graph-50-5	12	15	13.338	15	15	0	9.672
Graph-50-6	17	18	75.319	18	18	0	9.469
Graph-50-7	14	14	8.424	14	14	0	10.468
Graph-50-8	14	14	8.393	14	14	0	10.873
Graph-50-9	14	14	8.393	14	14	0	11.014
Graph-50-10	25	26	66.893	26	26	0	20.873
Graph-100-1	24	26	103.29	25	30	0	2398.392
Graph-100-2	18	24	220.63	23	25	0	1916.104
Graph-100-3	23	30	194.28	26	30	0	831.282
Graph-100-4	28	28	119.71	26	28	0	3600.118
Graph-100-5	19	24	88.11	23	26	0	2570.422
Graph-100-6	34	41	544.973	34	43	0	3418.552
Graph-100-7	38	43	809.65	38	43	0	2404.615
Graph-100-8	33	39	839.72	35	46	0	3025.386
Graph-100-9	38	48	1179.396	39	48	0	1243.476
Graph-100-10	32	34	175.164	33	35	0	1003.407
Graph-150-1	29	**40**	992.737	39	39	50.63	3620.379
Graph-150-2	26	**43**	1929.423	34	34	56.41	3624.623
Graph-150-3	38	**46**	3009.129	36	36	52.63	3618.92
Graph-150-4	24	**40**	736.892	34	34	54.05	3618.451
Graph-150-5	34	**55**	1561.173	44	44	43.59	3602.633
Graph-150-6	93	93	3600.823	93	93	0	1882.92
Graph-150-7	72	**82**	4054.29	75	75	15.73	3610.495
Graph-150-8	45	**49**	561.631	43	43	27.12	3611.868
Graph-150-9	72	**75**	3432.234	72	72	11.11	3607.687
Graph-150-10	88	**91**	3571.682	89	89	4.3	3654.051
Graph-200-1	44	**70**	3878.301	52	52	60.9	3682.424
Graph-200-2	52	**78**	3839.554	50	50	64.29	3683.23
Graph-200-3	80	**88**	4183.047	71	71	47.79	3605.31
Graph-200-4	87	**101**	3669.486	87	87	35.56	3667.71
Graph-200-5	42	**55**	3601.057	45	45	63.41	3643.58
Graph-200-6	97	**101**	3600.729	97	97	15.65	3672.48
Graph-200-7	154	**157**	4099.889	154	154	3.14	3710.07
Graph-200-8	118	**124**	4070.715	118	118	9.23	3642.55
Graph-200-9	147	147	3924.059	147	147	0.00	2353.25
Graph-200-10	110	**123**	3979.932	110	110	17.29	3669.3

Table B.6: Results of solving the MsCP using VNS and B&B algorithms(s=2, d=0.2)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	18	19	22.198	18	19	0	15.381
Graph-50-2	20	21	44.32	20	22	0	15.585
Graph-50-3	24	26	50.56	25	26	0	26.005
Graph-50-4	26	26	55.443	26	26	0	24.352
Graph-50-5	20	22	33.228	22	22	0	26.005
Graph-50-6	24	26	54.568	24	26	0	13.774
Graph-50-7	24	26	54.444	24	26	0	15.054
Graph-50-8	24	25	51.464	25	26	0	29.141
Graph-50-9	23	24	65.692	23	24	0	13.65
Graph-50-10	23	24	65.333	23	24	0	13.572
Graph-100-1	58	59	63.648	59	68	0	1014.562
Graph-100-2	42	45	65.067	49	49	18.33	3605.667
Graph-100-3	78	79	139.573	80	82	0	306.554
Graph-100-4	75	75	73.133	75	75	0	253.863
Graph-100-5	68	69	74.241	69	70	0	3495.815
Graph-100-6	76	76	80.293	76	77	0	172.726
Graph-100-7	68	68	66.69	68	70	0	729.458
Graph-100-8	69	70	69.888	70	71	0	329.216
Graph-100-9	67	68	66.206	67	69	0	1022.31
Graph-100-10	77	77	83.772	77	77	0	203.699
Graph-150-1	133	135	443.884	133	139	0	490.561
Graph-150-2	125	**132**	382.184	126	126	5.97	3600.41
Graph-150-3	135	136	408.829	136	138	0	515.708
Graph-150-4	131	131	214.141	131	133	0	1844.509
Graph-150-5	137	139	345.15	137	139	0	570.539
Graph-150-6	120	120	201.022	120	120	0	651.461
Graph-150-7	125	126	190.508	125	126	0	916.084
Graph-150-8	127	127	228.317	127	127	0	447.567
Graph-150-9	123	123	239.148	123	123	0	566.003
Graph-150-10	114	114	163.144	114	114	0.87	3607.679
Graph-200-1	197	197	859.764	197	197	0	1026.526
Graph-200-2	192	192	796.099	192	192	0	1038.243
Graph-200-3	195	195	916.406	195	195	0	1033.516
Graph-200-4	197	197	995.158	197	197	0	1030.505
Graph-200-5	193	193	582.8	193	194	0	1190.967
Graph-200-6	190	190	1179.672	190	190	0	1196.988
Graph-200-7	188	189	1359.509	189	189	0	1049.1
Graph-200-8	172	**173**	851.823	172	172	0.58	3615.206
Graph-200-9	186	187	1044.732	186	187	0	1460.118
Graph-200-10	165	167	656.323	166	167	0.60	3715.873

Table B.7: Results of solving the MsCP using VNS and B&B algorithms(s=2, d=0.25)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	27	30	26.52	27	32	0	43.633
Graph-50-2	35	36	29.655	35	36	0	15.709
Graph-50-3	42	42	81.978	42	42	0	9.282
Graph-50-4	32	32	19.251	32	32	0	21.965
Graph-50-5	26	28	30.155	26	30	0	44.148
Graph-50-6	37	37	33.244	37	37	0	4.446
Graph-50-7	25	28	16.614	25	29	0	15.475
Graph-50-8	30	31	20.389	30	31	0	14.18
Graph-50-9	30	31	20.374	30	31	0	14.165
Graph-50-10	29	29	16.879	29	29	0	16.333
Graph-100-1	95	95	96.205	95	96	0	101.01
Graph-100-2	91	94	170.336	91	94	0	112.788
Graph-100-3	95	97	190.039	95	97	0	96.782
Graph-100-4	96	97	110.011	96	97	0	90.386
Graph-100-5	97	97	111.228	97	97	0	68.874
Graph-100-6	83	84	125.466	83	85	0	135.455
Graph-100-7	89	89	94.66	89	89	0	69.311
Graph-100-8	82	83	127.202	82	83	0	284.325
Graph-100-9	88	88	90.184	88	88	0	68.141
Graph-100-10	97	97	93.943	97	97	0	69.186
Graph-150-1	150	150	0	150	150	0	0
Graph-150-2	150	150	0	150	150	0	0
Graph-150-3	150	150	0	150	150	0	0
Graph-150-4	150	150	0	150	150	0	0
Graph-150-5	150	150	0	150	150	0	0
Graph-150-6	142	142	208.322	142	143	0	403.322
Graph-150-7	142	142	188.442	142	142	0	336.663
Graph-150-8	145	145	298.587	145	145	0	337.007
Graph-150-9	145	145	176.085	145	145	0	336.352
Graph-150-10	145	145	300.019	145	145	0	335.946
Graph-200-1	200	200	0	200	200	0	0
Graph-200-2	200	200	0	200	200	0	0
Graph-200-3	200	200	0	200	200	0	0
Graph-200-4	200	200	0	200	200	0	0
Graph-200-5	200	200	0	200	200	0	0
Graph-200-6	197	197	489.43	197	197	0	110.87
Graph-200-7	194	194	847.44	194	194	0	343.077
Graph-200-8	200	200	0	200	200	0	0
Graph-200-9	196	196	718.69	196	196	0	131.494
Graph-200-10	197	197	472.33	197	197	0	111.603

Table B.8: Results of solving the MsCP using VNS and B&B algorithms(s=3, d=0.0125)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	2	2	0.046	2	2	0	0.078
Graph-50-2	5	5	0.055	5	5	0	0.078
Graph-50-3	5	5	0.055	5	5	0	0.094
Graph-50-4	4	4	0.069	4	4	0	0.078
Graph-50-5	4	4	0.068	4	4	0	0.078
Graph-50-6	5	5	0.051	5	5	0	0.078
Graph-50-7	5	5	0.051	5	5	0	0.062
Graph-50-8	5	5	0.051	5	5	0	0.063
Graph-50-9	5	5	0.051	5	5	0	0.063
Graph-50-10	4	4	0.048	4	4	0	0.062
Graph-100-1	10	10	0.269	10	10	0	0.327
Graph-100-2	8	8	0.356	8	8	0	0.328
Graph-100-3	9	9	0.405	9	9	0	0.375
Graph-100-4	9	9	0.303	9	9	0	0.344
Graph-100-5	9	9	0.301	9	9	0	0.343
Graph-100-6	8	8	0.249	8	8	0	0.28
Graph-100-7	8	8	0.248	8	8	0	0.296
Graph-100-8	8	8	0.248	8	8	0	0.297
Graph-100-9	7	7	0.181	7	7	0	0.281
Graph-100-10	7	7	0.18	7	7	0	0.281
Graph-150-1	13	13	1.622	13	13	0	1.388
Graph-150-2	11	11	1.034	11	11	0	1.014
Graph-150-3	11	11	1.027	11	11	0	0.998
Graph-150-4	12	12	0.84	12	12	0	0.858
Graph-150-5	12	12	0.842	12	12	0	0.842
Graph-150-6	14	14	0.984	14	14	0	1.108
Graph-150-7	14	14	0.985	14	14	0	1.108
Graph-150-8	12	12	1.454	12	12	0	1.201
Graph-150-9	12	12	1.463	12	12	0	1.202
Graph-150-10	12	12	1.461	12	12	0	1.201
Graph-200-1	14	14	3.271	14	14	0	3.307
Graph-200-2	14	14	3.272	14	14	0	3.307
Graph-200-3	14	14	4.978	13	14	0	282.284
Graph-200-4	14	14	4.221	14	14	0	3.4
Graph-200-5	14	14	4.168	14	14	0	3.401
Graph-200-6	15	15	3.947	15	15	0	293.515
Graph-200-7	19	19	3.495	19	19	0	315.091
Graph-200-8	14	14	4.93	13	14	0	271.91
Graph-200-9	14	14	4.93	13	14	0	266.971
Graph-200-10	12	12	3.525	12	12	0	2.606

Table B.9: Results of solving the MsCP using VNS and B&B algorithms(s=3, d=0.025)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	7	7	0.109	7	7	0	0.093
Graph-50-2	9	9	0.219	9	9	0	0.14
Graph-50-3	7	7	0.062	7	7	0	0.062
Graph-50-4	7	7	0.062	7	7	0	0.063
Graph-50-5	5	5	0.078	5	5	0	0.078
Graph-50-6	11	11	0.25	11	11	0	0.171
Graph-50-7	11	11	0.265	11	11	0	0.172
Graph-50-8	7	7	0.11	7	7	0	0.093
Graph-50-9	7	7	0.109	7	7	0	0.078
Graph-50-10	7	7	0.109	7	7	0	0.094
Graph-100-1	11	11	1.311	10	11	0	38.36
Graph-100-2	16	16	1.357	16	16	0	1.451
Graph-100-3	12	12	1.466	9	12	0	41.059
Graph-100-4	17	17	2.23	17	17	0	2.293
Graph-100-5	14	14	1.341	14	14	0	1.248
Graph-100-6	16	16	1.388	16	16	0	1.497
Graph-100-7	16	16	1.372	16	16	0	1.498
Graph-100-8	15	15	2.215	15	15	0	70.512
Graph-100-9	15	15	2.2	15	15	0	69.764
Graph-100-10	15	15	2.231	15	15	0	67.002
Graph-150-1	22	22	9.859	18	22	0	286.166
Graph-150-2	17	17	6.773	14	17	0	210.256
Graph-150-3	21	21	7.279	19	21	0	252.61
Graph-150-4	21	21	7.189	19	21	0	242.518
Graph-150-5	23	23	10.168	19	23	0	310.784
Graph-150-6	21	21	13.361	16	21	0	388.284
Graph-150-7	21	21	13.505	16	21	0	389.345
Graph-150-8	22	22	10.698	20	22	0	284.778
Graph-150-9	29	29	9.658	27	29	0	186.093
Graph-150-10	29	29	9.68	27	29	0	177.887
Graph-200-1	28	28	36.737	27	28	0	1046.121
Graph-200-2	29	29	36.364	25	29	0	1071.58
Graph-200-3	23	23	27.753	22	23	0	932.74
Graph-200-4	21	21	21.479	20	21	0	741.858
Graph-200-5	28	28	36.832	27	29	0	1071.642
Graph-200-6	31	32	44.647	30	32	0	1074.107
Graph-200-7	34	36	71.791	38	38	0	1272.385
Graph-200-8	31	31	36.925	28	31	0	978.853
Graph-200-9	29	29	43.821	23	29	0	1444.906
Graph-200-10	26	28	40.997	25	29	0	1048.382

Table B.10: Results of solving the MsCP using VNS and B&B algorithms(s=3, d=0.05)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	10	10	0.513	9	10	0	4.867
Graph-50-2	11	11	0.47	11	11	0	4.914
Graph-50-3	10	10	0.47	9	10	0	3.495
Graph-50-4	11	11	0.351	11	11	0	0.218
Graph-50-5	9	9	0.282	9	9	0	0.187
Graph-50-6	12	12	0.409	12	12	0	0.265
Graph-50-7	12	12	0.411	12	12	0	0.281
Graph-50-8	10	10	0.249	10	10	0	0.172
Graph-50-9	10	10	0.25	10	10	0	0.171
Graph-50-10	10	10	0.25	10	10	0	0.172
Graph-100-1	33	33	40.219	31	33	0	525.785
Graph-100-2	32	32	20.851	33	33	0	153.848
Graph-100-3	27	27	18.152	27	27	0	146.438
Graph-100-4	27	27	15.668	23	27	0	176.265
Graph-100-5	27	27	15.824	23	27	0	177.295
Graph-100-6	33	33	27.448	33	33	0	197.341
Graph-100-7	33	33	27.545	33	33	0	204.799
Graph-100-8	31	31	30.56	31	32	0	235.639
Graph-100-9	29	29	28.261	27	29	0	353.109
Graph-100-10	29	29	28.019	27	29	0	359.27
Graph-150-1	52	53	377.021	55	55	21.43	3608.639
Graph-150-2	45	50	329.893	50	50	20.63	3611.316
Graph-150-3	42	**43**	228.087	42	42	23.64	3605.425
Graph-150-4	41	**48**	368.877	34	34	51.43	3606.882
Graph-150-5	47	**53**	314.324	50	50	23.08	3613.397
Graph-150-6	69	69	303.342	60	69	0.00	2258.417
Graph-150-7	57	**57**	373.9	54	54	21.74	3619.574
Graph-150-8	52	**58**	322.967	51	51	25	3606.392
Graph-150-9	79	**80**	616.2	71	71	19.32	3600.199
Graph-150-10	67	**72**	510.775	69	69	13.75	3623.785
Graph-200-1	72	**77**	2072.716	73	73	42.97	3660.759
Graph-200-2	66	**74**	1987.643	49	49	60.48	3669.635
Graph-200-3	95	**101**	2065.05	82	82	40.58	3629.964
Graph-200-4	121	122	2054.738	123	123	13.38	3660.306
Graph-200-5	98	**107**	2047.859	98	98	27.41	3628.435
Graph-200-6	88	**95**	1641.525	90	90	18.92	3652.459
Graph-200-7	100	**107**	2024.412	85	85	30.89	3605.062
Graph-200-8	118	**131**	3405.085	114	114	19.15	3684.766
Graph-200-9	113	**115**	1944.281	106	106	17.19	3648.092
Graph-200-10	97	**99**	2181.707	97	97	23.02	3660.634

Table B.11: Results of solving the MsCP using VNS and B&B algorithms(s=3, d=0.1)

Instance		**VNS**			**B&B**		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	21	21	6.88	19	22	0	20.952
Graph-50-2	38	38	29.125	34	38	0	22.106
Graph-50-3	29	29	11.309	29	29	0	10.593
Graph-50-4	35	35	16.785	35	35	0	14.102
Graph-50-5	32	33	20.03	32	33	0	15.99
Graph-50-6	41	41	16.392	41	41	0	8.424
Graph-50-7	41	41	16.367	41	41	0	8.518
Graph-50-8	26	26	4.034	26	26	0	12.168
Graph-50-9	26	26	4.047	26	26	0	11.935
Graph-50-10	26	26	4.03	26	26	0	12.792
Graph-100-1	99	99	4.83	99	99	0	12.091
Graph-100-2	94	94	16.24	94	94	0	15.632
Graph-100-3	94	94	15.382	94	94	0	173.85
Graph-100-4	95	95	17.55	95	95	0	15.351
Graph-100-5	92	92	16.614	92	92	0	17.176
Graph-100-6	92	92	21.184	92	92	0	17.691
Graph-100-7	93	93	22.416	93	93	0	19.781
Graph-100-8	93	93	20.247	93	93	0	18.253
Graph-100-9	93	93	21.466	93	93	0	18.065
Graph-100-10	92	92	27.502	92	92	0	229.308
Graph-150-1	150	150	0	150	150	0	0
Graph-150-2	150	150	0	150	150	0	0
Graph-150-3	150	150	0	150	150	0	0
Graph-150-4	150	150	0	150	150	0	0
Graph-150-5	150	150	0	150	150	0	0
Graph-150-6	146	146	82.539	146	146	0	52.695
Graph-150-7	147	147	96.969	147	147	0	51.774
Graph-150-8	149	149	8.63	149	149	0	47.984
Graph-150-9	148	148	79.452	148	148	0	47.674
Graph-150-10	150	150	0	150	150	0	0
Graph-200-1	200	200	0	200	200	0	0
Graph-200-2	200	200	0	200	200	0	0
Graph-200-3	200	200	0	200	200	0	0
Graph-200-4	200	200	0	200	200	0	0
Graph-200-5	200	200	0	200	200	0	0
Graph-200-6	200	200	0	200	200	0	0
Graph-200-7	200	200	0	200	200	0	0
Graph-200-8	199	199	21.69	199	199	0	127.422
Graph-200-9	199	199	23.057	199	199	0	121.823
Graph-200-10	200	200	0	200	200	0	0

Table B.12: Results of solving the MsCP using VNS and B&B algorithms(s=3, d=0.15)

Instance		VNS			B&B		
	LB-1	s-club size	CPU (sec)	LB*	s-club size	Gap (%)	CPU (sec)
Graph-50-1	49	49	1.077	49	49	0	1.514
Graph-50-2	46	46	1.201	46	46	0	1.919
Graph-50-3	48	48	1.107	48	48	0	1.638
Graph-50-4	44	45	1.014	44	45	0	9.159
Graph-50-5	48	48	1.014	48	48	0	1.7
Graph-50-6	43	43	1.061	43	43	0	1.872
Graph-50-7	43	43	0.952	43	43	0	1.514
Graph-50-8	43	43	0.952	43	43	0	1.514
Graph-50-9	43	43	0.936	43	43	0	1.498
Graph-50-10	50	50	0	50	50	0	0
Graph-100-1	100	100	0	100	100	0	0
Graph-100-2	100	100	0	100	100	0	0
Graph-100-3	100	100	0	100	100	0	0
Graph-100-4	100	100	0	100	100	0	0
Graph-100-5	100	100	0	100	100	0	0
Graph-100-6	99	99	5.92	99	99	0	13.839
Graph-100-7	100	100	0	100	100	0	0
Graph-100-8	100	100	0	100	100	0	0
Graph-100-9	100	100	0	100	100	0	0
Graph-100-10	99	99	5.76	99	99	0	13.449
Graph-150-1	150	150	0	150	150	0	0
Graph-150-2	150	150	0	150	150	0	0
Graph-150-3	150	150	0	150	150	0	0
Graph-150-4	150	150	0	150	150	0	0
Graph-150-5	150	150	0	150	150	0	0
Graph-150-6	150	150	0	150	150	0	0
Graph-150-7	150	150	0	150	150	0	0
Graph-150-8	150	150	0	150	150	0	0
Graph-150-9	150	150	0	150	150	0	0
Graph-150-10	150	150	0	150	150	0	0
Graph-200-1	200	200	0	200	200	0	0
Graph-200-2	200	200	0	200	200	0	0
Graph-200-3	200	200	0	200	200	0	0
Graph-200-4	200	200	0	200	200	0	0
Graph-200-5	200	200	0	200	200	0	0
Graph-200-6	200	200	0	200	200	0	0
Graph-200-7	200	200	0	200	200	0	0
Graph-200-8	200	200	0	200	200	0	0
Graph-200-9	200	200	0	200	200	0	0
Graph-200-10	200	200	0	200	200	0	0

Table B.13: Results of solving the MsCP using hybrid algorithm(s=2, d=0.15)

Instance	Hybrid			
	LB-4	s-club size	Gap (%)	CPU (sec)
Graph-100-1	26	30	0	2225.15
Graph-100-2	23	25	0	1955.17
Graph-100-3	26	30	0	888.42
Graph-100-4	28	28	0	2815.3
Graph-100-5	23	26	0	2623.78
Graph-100-6	41	43	0	2263.01
Graph-100-7	43	43	0	946.17
Graph-100-8	43	46	0	2382.44
Graph-100-9	46	48	0	1208.29
Graph-100-10	34	35	0	1013.28
Graph-150-1	39	39	51.85	3584.47
Graph-150-2	41	41	48.75	3589.15
Graph-150-3	46	46	40.26	3609.77
Graph-150-4	45	45	40.79	3596.47
Graph-150-5	52	52	35	3598.95
Graph-150-6	93	93	0	2229.05
Graph-150-7	80	80	11.11	3599.93
Graph-150-8	49	49	18.33	3598.37
Graph-150-9	75	75	7.41	3614.8
Graph-150-10	91	91	3.19	3599.27
Graph-200-1	59	59	56.3	3556.32
Graph-200-2	78	78	44.68	3577.35
Graph-200-3	87	87	37.41	3564.67
Graph-200-4	101	101	26.28	3616.7
Graph-200-5	48	48	61.6	3602.94
Graph-200-6	101	101	13.68	3624.53
Graph-200-7	157	157	1.88	3604.6
Graph-200-8	124	124	6.77	3594.3
Graph-200-9	147	147	0	3444.36
Graph-200-10	120	120	10.45	3602.7

Table B.14: Results of solving the MsCP using hybrid algorithm(s=3, d=0.025, 0.05)

Instance	d=0.025				d=0.05			
	LB-4	s-club size	Gap (%)	CPU (sec)	LB-4	s-club size	Gap (%)	CPU (sec)
Graph-100-1	11	11	0	1.52	33	33	0	378.34
Graph-100-2	16	16	0	1.74	32	33	0	165.69
Graph-100-3	12	12	0	1.90	27	27	0	148.70
Graph-100-4	17	17	0	2.82	27	27	0	128.79
Graph-100-5	14	14	0	1.66	27	27	0	130.64
Graph-100-6	16	16	0	1.73	33	33	0	196.5
Graph-100-7	16	16	0	1.70	33	33	0	199.56
Graph-100-8	15	15	0	56.20	31	32	0	218.94
Graph-100-9	15	15	0	54.46	29	29	0	262.36
Graph-100-10	15	15	0	53.33	29	29	0	257.41
Graph-150-1	22	22	0	219.39	53	53	23.19	3600.293
Graph-150-2	17	17	0	170.24	50	50	19.35	3605.634
Graph-150-3	21	21	0	185.45	43	43	20.37	3607.645
Graph-150-4	21	21	0	181.23	48	48	29.41	3613.699
Graph-150-5	23	23	0	234.1	53	53	17.19	3600.679
Graph-150-6	21	21	0	283.51	69	69	0	1530.063
Graph-150-7	21	21	0	278.81	57	57	16.18	3554.371
Graph-150-8	22	22	0	211.58	58	58	13.43	3605.501
Graph-150-9	29	29	0	141.25	80	80	8.05	3607.561
Graph-150-10	29	29	0	146.34	72	72	7.69	3602.87
Graph-200-1	28	28	0	859.98	77	77	39.84	3621.68
Graph-200-2	29	29	0	864.59	74	74	40.32	3609.823
Graph-200-3	23	23	0	765.88	101	101	26.81	3607.112
Graph-200-4	21	21	0	620.55	122	122	14.08	3602.481
Graph-200-5	28	29	0	890.37	107	107	20.15	3591.639
Graph-200-6	32	32	0	853.6	95	95	15.18	3604.171
Graph-200-7	36	38	0	1322.53	107	107	12.3	3602.751
Graph-200-8	31	31	0	831.01	131	131	7.09	3609.511
Graph-200-9	29	29	0	934.565	115	115	10.16	3609.152
Graph-200-10	28	29	0	837.539	99	99	20.8	3600.769