MAXIMUM INDEPENDENT SET AND RELATED PROBLEMS, WITH APPLICATIONS

By

SERGIY BUTENKO

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2003

Dedicated to my family

ACKNOWLEDGMENTS

First of all, I would like to thank Panos Pardalos, who has been a great supervisor and mentor throughout my four years at the University of Florida. His inspiring enthusiasm and energy have been contagious, and his advice and constant support have been extremely helpful.

Special thanks go to Stan Uryasev, who recruited me to the University of Florida and was always very supportive. I would also like to acknowledge my committee members William Hager, Edwin Romeijn, and Max Shen for their time and guidance.

I am grateful to my collaborators James Abello, Vladimir Boginski, Stas Busygin, Xiuzhen Cheng, Ding-Zhu Du, Alexander Golodnikov, Carlos Oliveira, Mauricio G. C. Resende, Ivan V. Sergienko, Vladimir Shylo, Oleg Prokopyev, Petro Stetsyuk, and Vitaliy Yatsenko, who have been a pleasure to work with.

I would like to thank Donald Hearn for facilitating my graduate studies by providing financial as well as moral support. I am also grateful to the faculty, staff, and students of the Industrial and Systems Engineering Department at the University of Florida for helping make my experience here unforgettable.

Finally, my utmost appreciation goes to my family members, and especially my wife Joanna, whose love, understanding and faith in me made it a key ingredient of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

MAXIMUM INDEPENDENT SET AND RELATED PROBLEMS, WITH APPLICATIONS

By

Sergiy Butenko

August 2003

Chair: Panagote M. Pardalos
Major Department: Industrial and Systems Engineering

This dissertation develops novel approaches to solving computationally difficult combinatorial optimization problems on graphs, namely maximum independent set, maximum clique, graph coloring, minimum dominating sets and related problems. The application areas of the considered problems include information retrieval, classification theory, economics, scheduling, experimental design, and computer vision among many others.

The maximum independent set and related problems are formulated as nonlinear programs, and new methods for finding good quality approximate solutions in reasonable computational times are introduced. All algorithms are implemented and successfully tested on a number of examples from diverse application areas. The proposed methods favorably compare with other competing approaches.

A large part of this dissertation is devoted to detailed studies of selected applications of the problems of interest. Novel techniques for analyzing the structure of financial markets based on their network representation are proposed and verified using massive data sets generated by the U.S. stock markets. The network

representation of a market is based on cross-correlations of price fluctuations of the financial instruments, and provides a valuable tool for classifying the instruments.

In another application, new exact values and estimates of size of the largest error correcting codes are computed using optimization in specially constructed graphs. Error correcting codes lie in the heart of digital technology, making cell phones, compact disk players and modems possible. They are also of a special significance due to increasing importance of reliability issues in Internet transmissions.

Finally, efficient approximate algorithms for construction of virtual backbone in wireless networks by means of solving the minimum connected dominating set problem in unit-disk graphs are developed and tested.

CHAPTER 1
INTRODUCTION

Optimization has been expanding in all directions at an astonishing rate during the last few decades. New algorithmic and theoretical techniques have been developed, the diffusion into other disciplines has proceeded at a rapid pace, and our knowledge of all aspects of the field has grown even more profound [88, 169]. At the same time, one of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field. Optimization today is a basic research tool in all areas of engineering, medicine and the sciences. The decision making tools based on optimization procedures are successfully applied in a wide range of practical problems arising in virtually any sphere of human activities, including biomedicine, energy management, aerospace research, telecommunications and finance.

Having been applied in many diverse areas, the problems studied in this thesis provide a perfect illustration of the interdisciplinary developments in optimization. For instance, the *maximum independent set* problem put forward in the title of this thesis has applications in numerous fields, including information retrieval, signal transmission analysis, classification theory, economics, scheduling, and biomedical engineering. New findings concerning some of these applications will be discussed in a latter part of this thesis. But before we will present a formal description of results obtained in this work, let us gently touch the historical grounds of optimization theory and mathematical programming.

The problems of finding the "best" and the "worst" have always been of great interest. For example, given n sites, what is the fastest way to visit all of them consecutively? In manufacturing, how should one cut plates of a material so that the waste is minimized? Some of the first optimization problems were solved in ancient

Greece and are regarded among the most significant discoveries of that time. In the first century A.D., the Alexandrian mathematician Heron solved the problem of finding the shortest path between two points by way of the mirror. This result, also known as the Heron's theorem of the light ray, can be viewed as the origin of the theory of geometrical optics. The problem of finding extreme values gained a special importance in the seventeenth century when it served as one of motivations in the invention of differential calculus. The soon-after-developed calculus of variations and the theory of stationary values lie in the foundation of the modern mathematical theory of optimization.

The invention of the digital computer served as a powerful spur to the field of numerical optimization. During World War II optimization algorithms were used to solve military logistics and operations problems. The military applications motivated the development of linear programming (LP), which studies optimization problems with linear objective function and constraints. In 1947 George Dantzig invented the simplex method for solving linear programs arising in U.S. Air Force operations. Linear programming has become one of the most popular and well studied optimization topics ever since.

Despite a fine performance of the simplex method on a wide variety of practical instances, it has an exponential worst-case time complexity and therefore is unacceptably slow in some large-scale cases. The question of existence of a theoretically efficient algorithm for LP remained open until 1979, when Leonid Khachian published his polynomial time ellipsoid algorithm for linear programming [138]. This theoretical breakthrough was followed by the interior point algorithm of Narendra Karmarkar [137] published in 1984. Not only does this algorithm have a polynomial time theoretical complexity, it is also extremely efficient practically, allowing for solving larger instances of linear programs. Nowadays,

various versions of interior point methods are an integral part of the state-of-the-art optimization solvers.

Linear programming can be considered as a special case of a broad optimization area called *combinatorial optimization*, in which a the feasible region is a finite, but usually very large, set. All of the problems studied in this work are essentially combinatorial optimization problems. In nonlinear optimization, one deals with optimizing a nonlinear function over a feasible domain described by a set of, in general, nonlinear functions. It is fascinating to observe how naturally nonlinear and combinatorial optimization are bridged with each other to yield new, better optimization techniques. Combining the techniques for solving combinatorial problems with nonlinear optimization approaches is especially promising since it provides an alternative point of view and leads to new characterizations of the considered problems. These ideas also give a fresh insight into the complexity issues and frequently provide a guide to the discovery of nontrivial connections between problems of seemingly different nature. For example, the ellipsoid and interior point methods for linear programming mentioned above are based on nonlinear optimization techniques. Let us also mention that an integrality constraint of the form $x \in \{0, 1\}$ is equivalent to the nonconvex quadratic constraint $x^2 - x = 0$. This straightforward fact suggests that it is the presence of nonconvexity, not integrality that makes an optimization problem difficult [123]. The remarkable association between combinatorial and nonlinear optimization can also be observed throughout this thesis. In Chapter 2 we prove several continuous formulations of the maximum independent set problem. Consequently, in Chapter 3 we derive efficient algorithms for finding large independent sets based on these formulations.

As a result of ongoing enhancement of the optimization methodology and of improvement of available computational facilities, the scale of the problems solvable to optimality is continuously rising. However, many large-scale optimization problems

encountered in practice cannot be solved using traditional optimization techniques. A variety of new computational approaches, called *heuristics*, have been proposed for finding good sub-optimal solutions to difficult optimization problems. Etymologically, the word "heuristic" comes from the Greek *heuriskein* (to find). Recall the famous "Eureka! Eureka!" (I have found it! I have found it!) by Archimedes (287-212 B.C.). A heuristic in optimization is any method that finds an "acceptable" feasible solution. Many classical heuristics are based on local search procedures, which iteratively move to a better solution (if such solution exists) in a neighborhood of the current solution. A procedure of this type usually terminates when the first local optimum is obtained. Randomization and restarting approaches used to overcome poor quality local solutions are often ineffective. More general strategies known as metaheuristics usually combine some heuristic approaches and direct them towards solutions of better quality than those found by local search heuristics. Heuristics and metaheuristics play a key role in the solution of large difficult applied optimization problems. New efficient heuristics for the maximum independent set and related problems have been developed and successfully tested in this work.

In the remainder of this chapter we will first formally introduce definitions and notations used in this thesis. Then we will state some fundamental facts concerning the considered problems. Finally, in Section 1.7 we will outline the organization of the remaining chapters of this thesis.

1.1 Definitions and Notations

Let $G = (V, E)$ be a simple undirected graph with vertex set $V = \{1, \ldots, n\}$ and set of edges E. The *complement graph* of G is the graph $\bar{G} = (V, \bar{E})$, where $\bar{E}$ is the complement of E. For a subset $W \subseteq V$ let $G(W)$ denote the subgraph induced by W on G. $N(i)$ will denote the set of neighbors of vertex i and $d_i = |N(i)|$ is the degree of vertex i. We denote by $\Delta \equiv \Delta(G)$ the maximum degree of G.

A subset $I \subseteq V$ is called an *independent set* (*stable set, vertex packing*) if the edge set of the subgraph induced by I is empty. An independent set is *maximal* if it is not a subset of any larger independent set, and *maximum* if there are no larger independent sets in the graph. The *independence number* $\alpha(G)$ (also called the *stability number*) is the cardinality of a maximum independent set in G.

Along with the maximum independent set problem, we consider some other related problems for graphs, including the *maximum clique*, the *minimum vertex cover*, and the *maximum matching* problems. A *clique* C is a subset of V such that the subgraph $G(C)$ induced by C on G is complete. The maximum clique problem is to find a clique of maximum cardinality. The *clique number* $\omega(G)$ is the cardinality of a maximum clique in G. A *vertex cover* V' is a subset of V, such that every edge $(i, j) \in E$ has at least one endpoint in V'. The minimum vertex cover problem is to find a vertex cover of minimum cardinality. Two edges in a graph are *incident* if they have an endpoint in common. A set of edges is independent if no two of them are incident. A set M of independent edges in a graph $G = (V, E)$ is called a *matching*. The maximum matching problem is to find a matching of maximum cardinality.

It is easy to see that I is a maximum independent set of G if and only if I is a maximum clique of $\bar{G}$ and if and only if $V \setminus I$ is a minimum vertex cover of G. The last fact yields Gallai's identity [92]

$$\alpha(G) + |S| = |V(G)|, \tag{1.1}$$

where S is a minimum vertex cover of the graph G. Due to the close relation between the maximum independent set and maximum clique problems, we will operate with both problems while describing the properties and algorithms for the maximum independent set problem. In this case, it is clear that a result holding for the maximum clique problem in G will also be true for the maximum independent set problem in $\bar{G}$.

Sometimes, each vertex $i \in V$ is associated with a positive weight w_i. Then for a subset $S \subseteq V$ its weight $W(S)$ is defined as the sum of weights of all vertices in S:

$$W(S) = \sum_{i \in S} w_i.$$

The *maximum weight independent set* problem seeks for independent sets of maximum weight. The maximum weight clique problem is defined likewise.

The following definitions generalize the concept of clique. Namely, in some applications, instead of cliques one is interested in dense subgraphs, or *quasi-cliques*. A γ-*clique* C_γ, also called a *quasi-clique*, is a subset of V such that $G(C_\gamma)$ has at least $\lfloor \gamma q(q-1)/2 \rfloor$ edges, where q is the cardinality of C_γ.

A *legal (proper) coloring* of G is an assignment of colors to its vertices so that no pair of adjacent vertices has the same color. A coloring induces naturally a partition of the vertex set such that the elements of each set in the partition are pairwise nonadjacent. These sets are precisely the subsets of vertices being assigned the same color, which are independent sets by definition. If there exists a coloring of G that uses no more than k colors, we say that G admits a k-coloring (G is k-colorable). The minimal k for which G admits a k-coloring is called the *chromatic number* and is denoted by $\chi(G)$. The graph coloring problem is to find $\chi(G)$ as well as the partition of vertices induced by a $\chi(G)$-coloring.

Minimum clique partition problem, which is to partition vertices into minimum number of cliques, is analogous to graph coloring problem. In fact, any proper coloring in G is a clique partition in $\bar{G}$.

For other standard definitions which are used in this paper the reader is referred to a standard textbook in graph theory [31].

König's theorem (see page 30 in Diestel [74] and Rizzi [178] for a short proof) states that the maximum cardinality of a matching in a bipartite graph G is equal to the minimum cardinality of a vertex cover.

1.2 Complexity Results

The maximum clique and the graph coloring problems are NP-hard [96]; moreover, they are associated with a series of recent results about hardness of approximations. The discovery of a remarkable connection between probabilistically checkable proofs and approximability of combinatorial optimization problems [20, 21, 83] yielded new hardness of approximation results for many problems. Arora and Safra [21] proved that for some positive ϵ the approximation of the maximum clique within a factor of n^{ϵ} is NP-hard. Recently, Håstad [116] has shown that in fact for any $\delta > 0$ the maximum clique is hard to approximate in polynomial time within a factor $n^{1-\delta}$. Similar approximation complexity results hold for the graph coloring problem as well. Garey and Johnson [95] have shown that obtaining colorings using $s\chi(G)$ colors, where $s < 2$, is NP-hard. It has been shown by Lund and Yannakakis [152] that $\chi(G)$ is hard to approximate within n^{ϵ} for some $\epsilon > 0$, and following Håstad [116], Feige and Kilian [84] have shown that for any $\delta > 0$ the chromatic number is hard to approximate within a factor of $n^{1-\delta}$, unless NP $\subseteq$ ZPP. All of the above facts together with practical evidence [136] suggest that the maximum clique and coloring problems are hard to solve even in graphs of moderate sizes. The theoretical complexity and the huge sizes of data make these problems especially hard in massive graphs.

Alternatively, the maximum matching problem can be solved in polynomial time even for the weighted case (see, for instance, Papadimitriou and Steiglitz [168]).

1.3 Lower and Upper Bounds

In this section we briefly review some well-known lower and upper bounds on the independence and clique numbers.

Perhaps the best known lower bound based on degrees of vertices is given by Caro and Tuza [57], and Wei [200]:

$$\alpha(G) \geq \sum_{i \in V} \frac{1}{d_i + 1}.$$

In 1967, Wilf [201] showed that

$$\omega(G) \leq \rho(A_G) + 1,$$

where $\rho(A_G)$ is the spectral radius of the adjacency matrix of G (which is, by definition, the largest eigenvalue of A_G).

Denote by N_{-1} the number of eigenvalues of A_G that do not exceed -1, and by N_0 the number of zero eigenvalues. Amin and Hakimi [16] proved that

$$\omega(G) \leq N_{-1} + 1 < n - N_0 + 1.$$

In the above, the equality holds if G is a complete multipartite graph.

One of the most remarkable methods for computing an upper bound on independence number is based on semidefinite programming. In 1979, the Lovász theta number $\theta(G)$ was introduced, which gives an upper bound on the stability number $\alpha(G)$ [149]. This number is defined as the optimal value of the following semidefinite program:

$$\begin{aligned} \max \quad & e^T X e \\ s.t. \quad & \text{trace}(X) = 1, \\ & X_{ij} = 0 \text{ if } (i,j) \in E, \\ & X \succeq 0, \end{aligned}$$

where X is a symmetric matrix of size $n \times n$, the constraint $X \succeq 0$ requires X to be positive semidefinite, and e is the unit vector of length n. The famous *sandwich theorem* [142] states that the Lovász number $\theta(\bar{G})$ of the complement of a graph is sandwiched between the graph's clique number $\omega(G)$ and the chromatic number $\chi(G)$:

$$\omega(G) \leq \theta(\bar{G}) \leq \chi(G).$$

Since the above semidefinite program can be solved in polynomial time, the sandwich theorem also shows that in a *perfect* graph G (such that $\omega(G) = \chi(G)$ [154]), the clique number can be computed in polynomial time.

1.4 Exact Algorithms

In 1957, Harary and Ross [114] published, allegedly, the first algorithm for enumerating all cliques in a graph. Interestingly, their work was motivated by an application in sociometry. The idea of their method is to reduce the problem on general graphs to a special case with graphs having at most three cliques, and then solve the problem for this special case. This work was followed by many other algorithms. Paull and Unger [174], and Marcus [158] proposed algorithms for minimizing the number of states in sequential switching functions. The work of Bonner [41] was motivated by clustering problem. Bednarek and Taulbee [29] were looking for all maximal chains in a set with a given binary relation. Although all these approaches were designed to solve problems arising in different applications, their idea is essentially to enumerate all cliques in some graph.

The development of computer technology in 1960's made it possible to test the algorithms on graphs of larger sizes. As a result, in the early 1970's, many new enumerative algorithms were proposed and tested. Perhaps the most notable of these algorithms was the backtracking method by Bron and Kerbosch [45]. The advantages of their approach include its polynomial storage requirement and exclusion of the possibility of generating the same clique twice. The algorithm was successfully tested on graphs with 10 to 50 vertices and with edge density in the range between 10% and 95%. A modification of this approach considered by Tomita et al. [192] was claimed to have the time complexity of $O(3^{n/3})$, which is the best possible for enumerative algorithms due to existence of graphs with $3^{n/3}$ maximal cliques [163]. Recently, the Bron-Kerbosch algorithm was shown to be quite efficient with graphs arising in matching 3-dimensional molecular structures [93].

All of the algorithms mentioned above in this section were designed for finding all maximal cliques in a graph. However, to solve the maximum clique problem, one needs to find only the clique number and the corresponding maximum clique. There are many exact algorithms for the maximum clique and related problems available in the literature. Most of them are variations of the branch and bound method, which can be defined by different techniques for determining lower and upper bounds and by proper branching strategies. Tarjan and Trojanowski [191] proposed a recursive algorithm for the maximum independent set problem with the time complexity of $O(2^{n/3})$. Later, this result was improved by Robson [179], who modified the algorithm of Tarjan and Trojanowski to obtain the time complexity of $O(2^{0.276n})$.

Another important algorithm was developed by Balas and Yu in 1986 [25]. Using an interesting new implementation of the implicit enumeration, they were able to compute maximum cliques in graphs with up to 400 vertices and 30,000 edges. Carraghan and Pardalos [58] proposed yet another implicit enumerative algorithm for the maximum clique problem based on examining vertices in the order corresponding to the nondecreasing order of their degrees. Despite its simplicity, the approach proved to be very efficient, especially for sparse graphs. A publicly available implementation of this algorithm currently serves as a benchmark for comparing different algorithms [136]. Recently, Östergård [167] proposed a branch-and-bound algorithm which analyzes vertices in order defined by their coloring and employs a new pruning strategy. He compared the performance of this algorithm with several other approaches on random graphs and DIMACS benchmark instances and claimed that his algorithm is superior in many cases.

As most combinatorial optimization problems, the maximum independent set problem can be formulated as an integer program. Several such formulations will be mentioned in Chapter 2. The most powerful integer programming solvers used by modern optimization packages such as CPLEX of ILOG [125] and Xpress of Dash

Optimization [68] usually combine branch-and-bound algorithms with cutting plane methods, efficient preprocessing schemes, including fast heuristics, and sophisticated decomposition techniques in order to find an exact solution.

1.5 Heuristics

Although exact approaches provide an optimal solution, they become impractical (too slow) even on graphs with several hundreds of vertices. Therefore, when one deals with maximum independent set problem on very large graphs, the exact approaches cannot be applied, and heuristics provide the only available option.

Perhaps the simplest heuristics for the maximum independent set problem are *sequential greedy heuristics* which repeatedly add a vertex to an intermediate independent set (*"Best in"*) or remove a vertex from a set of vertices which is not an independent set (*"Worst out"*), based on some criterion, say the degrees of vertices [135, 144].

Sometimes in a search for efficient heuristics people turn to nature, which seems to always find the best solutions. In the recent decades, new types of optimization algorithms have been developed and successfully tested, which essentially attempt to imitate certain natural processes. The natural phenomena observed in annealing processes, nervous systems and natural evolution were adopted by optimizers and led to design of the simulated annealing [139], neural networks [121] and evolutionary computation [119] methods in the area of optimization. Other popular metaheuristics include greedy randomized adaptive search procedures or GRASP [86] and tabu search [101]. Various versions of these heuristics have been successfully applied to many important combinatorial optimization problems, including the maximum independent set problem. Below we will briefly review some of such heuristics.

1.5.1 Simulated Annealing

The annealing process is used to obtain a pure lattice structure in physics. It consists of first melting a solid by heating it up, and then solidifying it by slowly

cooling it down to a low-energy state. As a result of this process, the system's free energy is minimized. This property of the annealing process is used for optimization purposes in the simulated annealing method, where each feasible solution embodies a state of the hypothetical physical system, and the objective function represents the energy corresponding to the state. The basic idea of simulated annealing is the following. Generate an initial feasible solution $x^{(0)}$. At iteration $k+1$, given a feasible solution $x^{(k)}$, accept its neighbor $x^{(k+1)}$ as the next feasible solution with probability

$$p_{k+1} = \begin{cases} 1, & \text{if } f(x^{k+1}) < f(x^k); \\ \exp \frac{f(x^{k+1}) - f(x^k)}{t}, & \text{otherwise,} \end{cases}$$

where $f(x)$ is the cost function, and t is a parameter representing the temperature, which is modified as the optimization procedure is executed. A proper choice of the *cooling schedule* describing the change in the temperature parameter is one of the most important parts of the algorithm. A logarithmically slow cooling schedule leads to the optimal solution in exponential time; however in practice, faster cooling schedules yielding acceptable solutions are used.

An example of simulated annealing for the maximum independent set problem, using a penalty function approach was described in the textbook by Aarts and Korst [1]. They used the set of all possible subsets of vertices as the solution space, and the cost function in the form $f(V') = |V'| - \lambda|E'|$, where $V' \subset V$ is the given subset of vertices, $E' \subset V' \times V'$ is the set of edges in $G(V')$. Homer and Peinado [120] implemented a variation of the algorithm of Aarts and Korst with a simple cooling schedule, and compared its performance with the performance of several other heuristics for the maximum clique problem. Having run experiments on graphs with up to 70,000 vertices, they concluded that the simulated annealing approach was superior to the competing heuristics on the considered instances.

1.5.2 Neural Networks

Artificial neural networks (or simply, neural networks) represent an attempt to imitate some of the useful properties of biological nervous systems, such as adaptive biological learning. A neural network consists of a large number of highly interconnected processing elements emulating neurons, which are tied together with weighted connections analogous to synapses. Just like biological nervous systems, neural networks are characterized by massive parallelism and a high interconnectedness.

Although neural networks had been introduced as early as in the late 1950's, they were not widely applied until the mid-1980's, when sufficiently advanced related methodology has been developed. Hopfield and Tank [122] proved that certain neural network models can be used to approximately solve some difficult combinatorial optimization problems. Nowadays, neural networks are applied to many complex real-world problems. More detail on neural networks in optimization can be found, for example, in Zhang [205].

Various versions of neural networks have been applied to the maximum independent set problem by many researchers since the late 1980's. The efficiency of early attempts is difficult to evaluate due to the lack of experimental results. Therefore, we will mention some of the more recent contributions. Grossman [107] considered a discrete version of the Hopfield model for the maximum clique. The results of computational experiments with this approach on DIMACS benchmarks were satisfactory but inferior to other competing methods, such as simulated annealing. Jagota [127] proposed sevaral different discrete and continuous versions of the Hopfield model for the maximum clique problem. Later, Jagota et al. [128] improved some of these algorithms to obtain considerably larger cliques than those found by simpler heuristics, which ran only slightly faster.

1.5.3 Genetic Algorithms

Genetic algorithms in optimization were motivated by evolution processes in natural systems. Optimization in a genetic algorithm is carried out on a *population of points*, which are also called individuals or chromosomes. In the simplest version, the chromosomes are represented by binary vectors. Each chromosome is associated with a *fitness* value, which is the probability that the individual defined by this chromosome in the next generation will survive to the adulthood. Individual fitness values are used to compute values of the objective function representing the *total fitness* of the population. The fundamental law of natural selection stating that the total fitness of the population is non-decreasing from generation to generation, can be used as a basis for an optimization procedure. In its simplest version, the genetic algorithm starts with a population chosen randomly, and computes a new population using one of the three basic mechanisms: reproduction, crossover, or mutation [104]. Reproduction operator chooses the chromosomes used in the next generation according to the probability given by their fitness. The crossover operator is applied to produce new children from pairs of individuals. Finally, the mutation operator reverses the value of each bit in a chromosome with a given probability.

Early attempts to apply genetic algorithms to the maximum independent set and maximum clique problems were made in the beginning of 1990's. Many successful implementations have appeared in the literature ever since [50, 118, 157]. Most of the genetic algorithms can be easily parallelized.

1.5.4 Greedy Randomized Adaptive Search Procedures

A greedy randomized adaptive search procedure (GRASP) is an iterative randomized sampling technique in which each iteration provides an heuristic solution to the problem at hand [86]. The best solution over all GRASP iterations is kept as the final result. There are two phases within each GRASP iteration: the first constructs a list of solutions called *restricted candidate list (RCL)* via an adaptive randomized

greedy function; the second applies a local search technique to the constructed solution in hope of finding an improvement. GRASP has been applied successfully to a variety of combinatorial optimization problems including the maximum independent set problem [85].

A recent addition to GRASP, the so-called *path relinking* [102] has been proposed in order to enhance the performance of the heuristic by linking good quality solutions with a path of intermediate feasible points. It appears that in many cases some of these feasible points provide a better quality than the solutions used as the endpoints in the path. GRASP procedure with path relinking can be applied for the maximum independent set and the graph coloring problems.

1.5.5 Tabu Search

Tabu search [99, 100] is a variation of local search algorithms, which uses a *tabu* strategy in order to avoid repetitions while searching the space of feasible solutions. Such a strategy is implemented by maintaining a set of *tabu lists*, containing the paths of feasible solutions previously examined by the algorithm. The next visited solution is defined as the best *legal* (not prohibited by the tabu strategy) solution in the neighborhood of the current solution, even if it is worse than the current solution.

Various versions of tabu search have been successfully applied to the maximum independent set and maximum clique problems [28, 90, 155, 189].

1.5.6 Heuristics Based on Continuous Optimization

As it was discussed above, continuous approaches to combinatorial optimization problems turn out to be especially attractive. Much of the recent efforts in developing continuous-based heuristics for the maximum independent set and maximum clique problems focused around the Motzkin-Straus [165] formulation which relates the clique number of a graph to some quadratic program. This formulation will be proved and discussed in more detail in Chapter 2 of this work. We will also prove some other

continuous formulations of the considered problems and develop and test efficient heuristics based on these formulations.

Recently, we witnessed foundation and rapid development of semidefinite programming techniques, which are essentially continuous. The remarkable result by Goemans and Williamson [103] served as a major step forward in development of approximation algorithms and proved a special importance of semidefinite programming for combinatorial optimization. Burer et al. [52] considered rank-one and rank-two relaxations of the Lovász semidefinite program [108, 149] and derived two continuous optimization formulations for the maximum independent set problem. Based on these formulations, they developed and tested new heuristics for finding large independent sets.

1.6 Applications

Practical applications of the considered optimization problems are abundant. They appear in information retrieval, signal transmission analysis, classification theory, economics, scheduling, experimental design, and computer vision [4, 23, 25, 30, 38, 72, 171, 193]. In this section, we will briefly describe selected applications. Our work concerning applications in massive graphs, coding theory and wireless networks will be presented in Chapters 4-6.

1.6.1 Matching Molecular Structures by Clique Detection

Two graphs G_1 and G_2 are called isomorphic if there exists a one-to-one correspondence between their vertices, such that adjacent pairs of vertices in G_1 are mapped to adjacent pairs of vertices in G_2. A common subgraph of two graphs G_1 and G_2 consists of subgraphs $G_1^{'}$ and $G_2^{'}$ of G_1 and G_2, respectively, such that $G_1^{'}$ is isomorphic to $G_2^{'}$. The largest such common subgraph is the maximum common subgraph (MCS). For a pair of graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their correspondence graph C has all possible pairs (v_1, v_2), where $v_i \in V_i, i = 1, 2$, as

its set of vertices; two vertices (v_1, v_2) and $(v_1^{'}, v_2^{'})$ are connected in C if the values of the edges from v_1 to $v_1^{'}$ in G_1 and from v_2 to $v_2^{'}$ in G_2 are the same.

For a pair of three-dimensional chemical molecules the MCS is defined as the largest set of atoms that have matching distances between atoms (given user-defined tolerance values). It can be shown that the problem of finding the MCS can be solved efficiently using clique-detection algorithms applied to the correspondence graph [93].

1.6.2 Macromolecular Docking

Given two proteins, the *protein docking problem* is to find whether they interact to form a stable complex, and, if they do, then how. This problem is fundamental to all aspects of biological function, and development of reliable theoretical protein docking techniques is an important goal. One of the approaches to the macromolecular docking problem consists in representing each of two proteins as a set of potential hydrogen bond donors and acceptors and using a clique-detection algorithm to find maximally complementary sets of donor/acceptor pairs [94].

1.6.3 Integration of Genome Mapping Data

Due to differences in the methods used for analyzing overlap and probe data, the integration of these data becomes an important problem. It appears that overlap data can be effectively converted to probe-like data elements by finding maximal sets of mutually overlapping clones [115]. Each set determines a site in the genome corresponding to the region which is common among the clones of the set; therefore these sets are called virtual probes. Finding the virtual probes is equivalent to finding the maximal cliques of a graph.

1.6.4 Comparative Modeling of Protein Structure

The rapidly growing number of known protein structures requires the construction of accurate comparative models. This can be done using a clique-detection approach as follows. We construct a graph in which vertices correspond to each possible conformation of a residue in an amino acid sequence, and edges

connect pairs of residue conformations (vertices) that are consistent with each other (*i.e.*, clash-free and satisfying geometrical constraints). Based on the interaction between the atoms corresponding to the two vertices, weights are assigned to the edges. Then the cliques with the largest weights in the constructed graph represent the optimal combination of the various main-chain and side-chain possibilities, taking the respective environments into account [180].

1.6.5 Covering Location Using Clique Partition

Covering problems arise in location theory. Given a set of *demand points* and a set of *potential sites* for locating facilities, a demand point located within a prespecified distance from a facility is said to be *covered* by that facility. Covering problems are usually classified into two main classes [69]: *mandatory coverage* problems aim to cover all demand points with the minimum number of facilities; and *maximal coverage* problems intend to cover a maximum number of demand points with a given number of facilities.

Location problems are also classified by the nature of sets of demand points and potential sites, each of which can be discrete or continuous. In most applications both sets are discrete, but in some cases at least one of them is continuous. For example, Brotcorne et al. [46] consider an application of mandatory coverage problem arising in cytological screening tests for cervical cancer, in which the set of demand points is discrete and the set of potential sites is continuous. In this application, a servical specimen on a glass slide has to be viewed by a screener, which is relocated on the glass slide in order to explore the entire specimen. The goal is to maximize efficiency by minimizing the number of viewing locations. The area covered by the screener can be square or circular. To be specific, consider the case of the square screener, which can move in any of four directions parallel to the sides of the rectangular glass slide. In this case, we need to cover the viewing area of the slide by squares called *tiles*. To locate a tile it is suffices to specify the position of its center on the tile.

Let us formulate the problem in mathematical terms. Given n demand points inside a rectangle, we want to cover these points with a minimum number of squares with side s (tiles).

Lemma 1.1. *The following two statements are equivalent:*

1. *There exists a covering of n demand points in the rectangle using k tiles.*
2. *Given n tiles centered in the demand points, there exist k points in the rectangle such that each of the tiles contains at least one of them.*

Let us introduce an undirected graph $G = (V, E)$ associated with this problem. The set of vertices $V = \{1, 2, \ldots, n\}$ corresponds to the set of demand points. The set of edges E is constructed as follows. Consider the set $T = \{t_1, t_2, \ldots, t_n\}$ of tiles, each centered in a demand point. Then two vertices i and j are connected by an edge if and only if $t_i \cap t_j \neq \emptyset$. Then, in order to cover the demand points with minimum number of tiles, or the same, minimize the number of viewing locations, it suffices to solve minimum clique partition problem in the constructed graph (or the graph coloring problem in its complement).

1.7 Organization of the Dissertation

Organizationally, this thesis is divided into seven chapters. Chapter 2 introduces mathematical programming formulations of the maximum independent set and related problems. In particular, we prove several continuous optimization formulations. These formulations and some classical results concerning the problem of optimizing a quadratic over a sphere are utilized in Chapter 3, where we propose several heuristics for the maximum independent set problem. To demonstrate their efficiency, we provide the results of numerical experiments with our approaches, as well as comparison with other heuristics.

Chapter 4 discusses applications of the considered graph problems in studying massive data sets. First, we review recent advances and challenges in modeling and optimization of massive graphs. Then we introduce the notion of the so-called market

graph, in which financial instruments are represented by vertices, and edges are constructed based on the values of pairwise cross-correlations of the price fluctuations computed over a certain time period. We analyze structural properties of this graph and suggest some practical implications of the obtained results.

New exact solutions and improved lower bounds for the problem of finding the largest correcting codes dealing with certain types of errors are reported in Chapter 5. These results were obtained using efficient techniques for solving the maximum independent set and graph coloring problems in specially constructed graphs. Yet another application of the considered graph optimization problems is presented in Chapter 6, in which we solve the problem of constructing a virtual backbone in ad hoc wireless networks by approximating it with the minimum connected dominating set problem in unit-disk graphs. The proposed distributed algorithm has a fixed performance ratio and consists of two stages. Namely, it first computes a maximal independent set and then connects it using some additional vertices. The results of numerical simulation are also included. Finally, we conclude with some remarks and ideas for the future research in Chapter 7.

CHAPTER 2
MATHEMATICAL PROGRAMMING FORMULATIONS

The maximum independent set problem has many equivalent formulations as an integer programming problem and as a continuous nonconvex optimization problem [38, 171]. In this chapter we will review some existing approaches and will present proofs for some of the formulations. We will start with mentioning some integer programming formulations in the next section. In Sections 2.2 and 2.3, we will consider several continuous formulations of the maximum independent set problem and will give proofs for some of them. This will follow by a generalization of one of the formulations to dominating sets in Section 2.4. Most results presented in this chapter previously appeared in Abello et al. [3].

2.1 Integer Programming Formulations

Given a vector $w \in \mathbb{R}^n$ of positive weights w_i (associated with each vertex $i, i = 1, \ldots, n$), the *maximum weight independent set* problem asks for independent sets of maximum weight. Obviously, it is a generalization of the maximum independent set problem. One of the simplest formulations of the maximum weight independent set problem is the following *edge formulation*:

$$\max \ f(x) = \sum_{i=1}^{n} w_i x_i, \tag{2.1}$$

subject to

$$x_i + x_j \leq 1, \forall (i,j) \in E, \tag{2.1a}$$

$$x_i \in \{0,1\}, i = 1, \ldots, n. \tag{2.1b}$$

An alternative formulation of this problem is the following *clique formulation* [108]:

$$\max \ f(x) = \sum_{i=1}^{n} w_i x_i, \tag{2.2}$$

subject to

$$\sum_{i \in S} x_i \leq 1, \forall S \in \mathcal{C} \equiv \{\text{maximal cliques of } G\}, \tag{2.2a}$$

$$x_i \in \{0, 1\}, i = 1, \ldots, n. \tag{2.2b}$$

The advantage of formulation (2.2) over (2.1) is a smaller gap between the optimal values of (2.2) and its linear relaxation. However, since there is an exponential number of constraints in (2.2a), finding an efficient solution of (2.2) is difficult.

Here we mention one more integer programming formulation. Let A_G be the adjacency matrix of a graph G, and let J denote the $n \times n$ identity matrix. The maximum independent set problem is equivalent to the global quadratic zero-one problem

$$\max f(x) = x^T A x, \tag{2.3}$$

subject to

$$x_i \in \{0, 1\}, i = 1, \ldots, n, \tag{2.3a}$$

where $A = J - A_G$. If x^* is a solution to (2.3), then the set I defined by $I = \{i \in V : x_i^* = 1\}$ is a maximum independent set of G with $|I| = f(x^*)$. See Pardalos and Rodgers [170] for details.

2.2 Continuous Formulations

Shor [183] considered an interesting formulation of the maximum weight independent set problem by noticing that formulation (2.1) is equivalent to the

quadratically constrained global optimization problem

$$\max\ f(x) = \sum_{i=1}^{n} w_i x_i, \tag{2.4}$$

subject to

$$x_i x_j = 0,\ \forall\ (i,j) \in E, \tag{2.4a}$$

$$x_i{}^2 - x_i = 0,\ i = 1, 2, ..., n. \tag{2.4b}$$

Applying dual quadratic estimates, Shor reported good computational results and presented a new way to compute the Lovász number of a graph [149].

Motzkin and Straus [165] established a remarkable connection between the maximum clique problem and a certain standard quadratic programming problem. The original proof of the Motzkin–Straus theorem was by induction. Below we present a new proof. Let A_G be the adjacency matrix of G and let e be the n-dimensional vector with all components equal to 1.

Theorem 2.1 (Motzkin-Straus). *The global optimal value of the following quadratic program*

$$\max f(x) = \frac{1}{2} x^T A_G x, \tag{2.5}$$

subject to

$$e^T x = 1, \tag{2.5a}$$

$$x \geq 0. \tag{2.5b}$$

is given by

$$\frac{1}{2}\left(1 - \frac{1}{\omega(G)}\right),$$

where $\omega(G)$ is the clique number of G.

Proof. We will use the following well known inequality for the proof:

$$\sum_{i=1}^{n} a_i^2 \geq \frac{\left(\sum_{i=1}^{n} a_i\right)^2}{n}, \tag{2.6}$$

where $a_1, a_2, \ldots, a_n$ are positive numbers. Equality takes place if and only if $a_1 = a_2 = \ldots = a_n$.

Now consider the program (2.5). Let J denote the $n \times n$ identity matrix and let O be the $n \times n$ matrix of all ones. Then

$$A_G = O - J - A_{\bar{G}}$$

and

$$y^T A_G y = y^T O y - y^T J y - y^T A_{\bar{G}} y = 1 - (y^T J y + y^T A_{\bar{G}} y),$$

where $A_{\bar{G}}$ is the adjacency matrix of the complement graph $\bar{G}$.

Let $R(x) = x^T J x + x^T A_{\bar{G}} x$. Program (2.7) is equivalent to (2.5).

$$\begin{aligned} \min\ & R(x) = x^T J x + x^T A_{\bar{G}} x, \\ \text{s.t.}\ & e^T x = 1, \\ & x \geq 0. \end{aligned} \tag{2.7}$$

To check that there always exists an optimal solution x^* of (2.7) such that ${x^*}^T A_{\bar{G}} x^* = 0$, consider any optimal solution $\hat{x}$ of (2.7). Assume that $\hat{x}^T A_{\bar{G}} \hat{x} > 0$. Then there exists a pair $(i, j) \in \bar{E}$ such that $\hat{x}_i \hat{x}_j > 0$. Consider the following representation for $R(x)$:

$$R(x) = R_{ij}(x) + \bar{R}_{ij}(x),$$

where

$$R_{ij}(x) = x_i^2 + x_j^2 + 2x_i x_j + 2x_i \sum_{(i,k)\in\bar{E}, k\neq j} x_k + 2x_j \sum_{(j,k)\in\bar{E}, k\neq i} x_k;$$

$$\bar{R}_{ij}(x) = R(x) - R_{ij}(x).$$

Without loss of generality, assume that $\sum_{(i,k)\in\bar{E},k\neq i} \hat{x}_k \leq \sum_{(j,k)\in\bar{E},k\neq j} \hat{x}_k$. Then, if we set

$$\tilde{x}_k = \begin{cases} \hat{x}_i + \hat{x}_j, & \text{if} \quad k = i, \\ 0, & \text{if} \quad k = j, \\ \hat{x}_k, & \text{otherwise}, \end{cases}$$

we have:

$$R(\tilde{x}) = R_{ij}(\tilde{x}) + \bar{R}_{ij}(\tilde{x}) =$$

$$(\hat{x}_i + \hat{x}_j)^2 + 2(\hat{x}_i + \hat{x}_j) \cdot \sum_{(i,k)\in\bar{E},k\neq i} \hat{x}_k \leq$$

$$\hat{x}_i^2 + \hat{x}_j^2 + 2\hat{x}_i\hat{x}_j + 2\hat{x}_i \cdot \sum_{(i,k)\in\bar{E},k\neq j} \hat{x}_k + 2\hat{x}_j \cdot \sum_{(j,k)\in\bar{E},k\neq i} \hat{x}_k = R(\hat{x}).$$

If we denote by $Z(x) = \{(i,j) \in \bar{E} : x_i x_j > 0\}$, then $\tilde{x}$ is an optimal solution of (2.7) with $|Z(\tilde{x})| < |Z(\hat{x})|$. Repeating this procedure a finite number of times we will finally obtain an optimal solution x^* for which $|Z(x^*)| = 0$ and thus ${x^*}^T A_{\bar{G}} x^* = 0$.

Note that ${x^*}^T A_{\bar{G}} x^* = 0$ if and only if $\forall (i,j) \in \bar{E} : x_i^* x_j^* = 0$. This means that if we consider the set $C = \{i : x_i^* > 0\}$ then C is a clique.

Without loss of generality, assume that $x_i^* > 0$ for $i = 1, 2, \ldots, m$ and $x_i^* = 0$ for $m + 1 \leq i \leq n$. Consider the objective function of (2.7),

$$R(x^*) = {x^*}^T J x^* = \sum_{i=1}^{m} {x_i^*}^2.$$

By inequality (2.6) and the feasibility of x^* for (2.7),

$$\sum_{i=1}^{n} {x_i^*}^2 \geq \frac{\left(\sum_{i=1}^{n} x_i^*\right)^2}{m} = \frac{1}{m}.$$

Since C is a clique of cardinality m, it follows $m \leq \omega(G)$ and

$$R(x^*) \geq \frac{1}{m} \geq \frac{1}{\omega(G)}.$$

On the other hand, if we consider

$$x_k^* = \begin{cases} \frac{1}{\omega(G)}, \text{if} & k \in C^*, \\ 0, & \text{otherwise}, \end{cases}$$

where C^* is a maximum clique of G, then x^* is feasible and $R(x^*) = \frac{1}{\omega(G)}$. Thus x^* is an optimal solution of (2.7). Returning back to the original quadratic program, the result of the theorem follows. □

This result is extended by Gibbons et al. [98], who provided a characterization of maximal cliques in terms of local solutions. Moreover, optimality conditions of the Motzkin-Straus program have been studied and properties of a newly introduced parametrization of the corresponding QP have been investigated. Sós and Straus [190] further generalized the same theorem to hypergraphs.

2.3 Polynomial Formulations Over the Unit Hypercube

In this section we consider some of the continuous formulations originally proved by Harant et al. [112, 113] using probabilistic methods. We prove deterministically that the independence number of a graph G can be characterized as an optimization problem based on these formulations. We consider two polynomial formulations, a degree $(\Delta + 1)$ formulation and a quadratic formulation.

2.3.1 Degree $(\Delta + 1)$ Polynomial Formulation

Consider the degree $(\Delta + 1)$ polynomial of n variables

$$F(x) = \sum_{i=1}^{n} (1 - x_i) \prod_{(i,j) \in E} x_j \ , \quad x \in [0,1]^n.$$

The following theorem characterizes the independence number of the graph G as the maximization of $F(x)$ over the n-dimensional hypercube.

Theorem 2.2. *Let $G = (V, E)$ be a simple graph on n nodes $V = \{1, \ldots, n\}$ and set of edges E, and let $\alpha(G)$ denote the independence number of G. Then*

$$\alpha(G) = \max_{0 \le x_i \le 1, i=1,\ldots,n} F(x) = \max_{0 \le x_i \le 1, i=1,\ldots,n} \sum_{i=1}^{n} (1 - x_i) \prod_{(i,j) \in E} x_j, \quad \textbf{(P1)}$$

where each variable x_i corresponds to node $i \in V$.

Proof. Denote the optimal objective function value by $f(G)$, *i.e.*,

$$f(G) = \max_{0 \le x_i \le 1, i=1,\ldots,n} F(x) = \max_{0 \le x_i \le 1, i=1,\ldots,n} \sum_{i=1}^{n} (1 - x_i) \prod_{(i,j) \in E} x_j. \quad (2.8)$$

We want to show that $\alpha(G) = f(G)$.

First we show that (2.8) always has an optimal 0-1 solution. This is so because $F(x)$ is a continuous function and $[0, 1]^n = \{(x_1, x_2, \ldots, x_n) : 0 \le x_i \le 1, i = 1, \ldots, n\}$ is a compact set. Hence, there always exists $x^* \in [0, 1]^n$ such that $F(x^*) = \max_{0 \le x_i \le 1, i=1,\ldots,n} F(x)$.

Now, fix any $i \in V$. We can rewrite $F(x)$ in the form

$$F(x) = (1 - x_i)A_i(x) + x_i B_i(x) + C_i(x), \quad (2.9)$$

where

$$A_i(x) = \prod_{(i,j) \in E} x_j, \quad (2.10)$$

$$B_i(x) = \sum_{(i,k) \in E} (1 - x_k) \prod_{(k,j) \in E, j \ne i} x_j, \quad (2.11)$$

$$C_i(x) = \sum_{(i,k) \notin E} (1 - x_k) \prod_{(k,j) \in E} x_j. \quad (2.12)$$

Expressions (2.10 -2.12) can be interpreted in terms of neighborhoods. $A_i(x)$ and $B_i(x)$ characterize the first- and the second-order neighborhoods of vertex i, respectively, and $C_i(x)$ is complementary to $B_i(x)$ with respect to i in the sense that it describes neighborhoods of all vertices, other than i, which are not characterized by $B_i(x)$.

Notice that x_i is absent in (2.10 –2.12), and therefore $F(x)$ is linear with respect to each variable. It is also clear from the above representation that if x^* is any optimal solution of (2.8), then $x_i^* = 0$ if $A_i(x^*) > B_i(x^*)$, and $x_i^* = 1$, if $A_i(x^*) < B_i(x^*)$. Finally, if $A_i(x^*) = B_i(x^*)$, we can set $x_i^* = 1$. This shows that (2.8) always has an optimal 0-1 solution.

To show that $f(G) \geq \alpha(G)$, assume that $\alpha(G) = m$ and let I be a maximum independent set. Set

$$x_i^* = \begin{cases} 0, & \text{if } i \in I; \\ 1, & \text{otherwise.} \end{cases} \tag{2.13}$$

Then, $f(G) = \max\limits_{0 \leq x_i \leq 1, i=1,\ldots,n} F(x) \geq F(x^*) = m = \alpha(G)$.

To complete the proof, we need to show that $f(G) \leq \alpha(G)$. Since the considered problem always has an optimal 0-1 solution, it follows that $f(G)$ must be integer. Assume $f(G) = m$ and take any optimal 0-1 solution of (2.8). Without loss of generality we can assume that this solution is $x_1^* = x_2^* = \ldots = x_k^* = 0$; $x_{k+1}^* = x_{k+2}^* = \ldots = x_n^* = 1$, for some k. Then we have:

$$(1 - x_1^*) \prod_{(1,j) \in E} x_j^* + (1 - x_2^*) \prod_{(2,j) \in E} x_j^* + \ldots + (1 - x_k^*) \prod_{(k,j) \in E} x_j^* = m. \tag{2.14}$$

Each term in (2.14) is either 0 or 1. Therefore $k \geq m$ and there exists a subset $I \subset \{1, \ldots, k\}$ such that $|I| = m$ and

$$\forall i \in I : \prod_{(i,j)\in E} x_j^* = 1.$$

Therefore, if $(i,j) \in E$, then $x_j^* = 1$. Note that since $x_1^* = x_2^* = \ldots = x_k^* = 0$, it follows that $\forall\{i,j\} \subset I$ we have $(i,j) \notin E$ and so I is an independent set by definition. Thus, $\alpha(G) \geq |I| = m = f(G)$, which completes the proof. □

Corollary 2.1. *The clique number $\omega(G)$ in a simple undirected graph $G = (V, E)$ satisfies*

$$\omega(G) = \max_{0\leq x_i\leq 1, i=1,\ldots,n} \sum_{i=1}^{n} (1 - x_i) \prod_{(i,j)\notin E, i\neq j} x_j.$$

Proof. The statement follows from Theorem 2.2 and the fact that any clique in G is an independent set for $\bar{G}$. □

Corollary 2.2. *The size $|S|$ of a minimum vertex cover S in a simple graph $G = (V, E)$ satisfies*

$$|S| = n - \max_{0\leq x_i\leq 1, i=1,\ldots,n} \sum_{i=1}^{n} (1 - x_i) \prod_{(i,j)\in E} x_j$$

Proof. The result follows from Theorem 2.2 and Gallai's identity (1.1). □

Corollary 2.3. *The size $|M|$ of a maximum matching M in a bipartite graph $G = (V, E)$ satisfies*

$$|M| = n - \max_{0\leq x_i\leq 1, i=1,\ldots,n} \sum_{i=1}^{n} (1 - x_i) \prod_{(i,j)\in E} x_j.$$

Proof. The statement follows from Corollary 2.2 and König's theorem. □

2.3.2 Quadratic Polynomial Formulation

Consider now the quadratic polynomial

$$H(x) = \sum_{i=1}^{n} x_i - \sum_{(i,j)\in E} x_i x_j,$$

defined for $x \in [0,1]^n$.

The following theorem characterizes the independence number of the graph G as the maximization of $H(x)$ over the n-dimensional hypercube.

Theorem 2.3. *Let $G = (V, E)$ be a simple graph on n nodes $V = \{1, \ldots, n\}$ and set of edges E, and let $\alpha(G)$ denote the independence number of G. Then*

$$\alpha(G) = \max_{0 \le x_i \le 1, i=1,\ldots,n} H(x) = \max_{0 \le x_i \le 1, i=1,\ldots,n} \left(\sum_{i=1}^{n} x_i - \sum_{(i,j) \in E} x_i x_j \right), \quad \textbf{(P2)}$$

where each variable x_i corresponds to node $i \in V$.

Proof. Denote the optimal objective function value by $h(G)$, *i.e.*,

$$h(G) = \max_{0 \le x_i \le 1, i=1,\ldots,n} \left(\sum_{i=1}^{n} x_i - \sum_{(i,j) \in E} x_i x_j \right) \tag{2.15}$$

and let I be a maximum independent set. To prove that $h(G) \ge \alpha(G)$, let

$$x_i^* = \begin{cases} 1, & \text{if } i \in I; \\ 0, & \text{otherwise.} \end{cases} \tag{2.16}$$

Since I is an independent set and $x_i^* = 0$ for $i \notin I$, then $\sum_{(i,j) \in E} x_i^* x_j^* = 0$. Furthermore, $\sum_{i=1}^{n} x_i^* = |I| = \alpha(G)$. This yields $h(G) \ge H(x^*) = \alpha(G)$.

To complete the proof, we need to show that $h(G) \le \alpha(G)$. Assume $h(G) = m$. Since $H(x)$ is linear with respect to each variable, problem (2.15) always has an optimal 0-1 solution. Take any optimal 0-1 solution x of (2.15). Suppose, that there exists $(i_0, j_0) \in E$ such that $x_{i_0} = x_{j_0} = 1$. Changing x_{i_0} to 0 decreases $\sum_{i=1}^{n} x_i$ by 1 and decreases $\sum_{(i,j) \in E} x_i x_j$ by at least 1. Thus, the objective function will not decrease. Doing this for all such pairs (i_0, j_0) will finally lead to an optimal solution x^* such

that $\forall (i,j) \in E : x_i^* x_j^* = 0$, and an independent set $I = \{i : x_i^* = 1\}$ of cardinality $h(G)$. This yields $h(G) \leq \alpha(G)$ and the theorem is proved. □

2.3.3 Relation Between (P2) and Motzkin-Straus QP

The next statement is a reformulation the Motzkin-Straus theorem for the maximum independent set problem. We show how it can be obtained from the formulation (**P2**).

Theorem 2.4. *The global optimal value of the following QP*

$$\max f(x) = \frac{1}{2} x^T A_{\bar{G}} x, \tag{2.17}$$

$$\tag{2.18}$$

$$s.t. \quad e^T x = 1,$$

$$x \geq 0.$$

is given by

$$\frac{1}{2}\left(1 - \frac{1}{\alpha(G)}\right),$$

where $\alpha(G)$ is the independence number of G.

Proof. Consider formulation (**P2**):

$$\alpha(G) = \max_{0 \leq x_i \leq 1, i=1,\ldots,n} \left(\sum_{i=1}^{n} x_i - \sum_{(i,j) \in E} x_i x_j \right) = \max_{0 \leq x_i \leq 1, i=1,\ldots,n} \left(e^T x - \frac{1}{2} x^T A_G x \right).$$

It is easy to see that changing the feasible region from $[0,1]^n$ in the last QP to the following

$$\{x \geq 0 : e^T x = \alpha(G)\}$$

does not change the optimal objective function value, thus, changing the variables to $y = \frac{1}{\alpha(G)} x$, we obtain:

$$\alpha(G) = \max \left(\alpha(G) e^T y - \frac{1}{2} \alpha(G)^2 y^T A_G y \right) \tag{2.19}$$

s. t.

$$e^T y = 1,$$

$$y \geq 0.$$

Let I is a maximum independent set of G. Then

$$y_i^* = \begin{cases} \frac{1}{\alpha(G)}, \text{ if } i \in I \\ 0, \text{otherwise,} \end{cases}$$

is an optimal solution for the last program.

We have:

$$A_G = O - J - A_{\bar{G}}$$

and

$$y^T A_G y = y^T O y - y^T J y - y^T A_{\bar{G}} y = 1 - y^T J y - y^T A_{\bar{G}} y,$$

where $A_{\bar{G}}$ is the adjacency matrix of the complement graph $\bar{G}$. Denote by $\mathcal{F} = \{y \geq 0 : e^T y = 1\}$. Then (2.19) is equivalent to the following:

$$\alpha(G) = \max_{y \in \mathcal{F}} \left(\alpha(G) + \frac{1}{2}\alpha(G)^2(-1 + y^T O y + y^T A_{\bar{G}} y \right)$$

which yields

$$1 = \max_{y \in \mathcal{F}} \left(y^T J y + y^T A_{\bar{G}} y \right)$$

Since the maximum is reached in y^*, we have:

$$1 - \frac{1}{\alpha(G)} = y^{*T} A_{\bar{G}} y^* \leq \max_{y \in \mathcal{F}} y^T A_{\bar{G}} y.$$

Now assume that for some $\hat{y}$:

$$\hat{y}^T A_{\bar{G}} \hat{y} = \max_{y \in \mathcal{F}} y^T A_{\bar{G}} y > 1 - \frac{1}{\alpha(G)}.$$

Then there exists $\tilde{y}$ with $|\{(i,j) \in E : \tilde{y}_i \tilde{y}_j > 0\}| = 0$ such that $\tilde{y}^T A_{\bar{G}} \hat{y} \geq \hat{y}^T A_{\bar{G}} \hat{y}$. For $\tilde{y}$ we have:

$$1 - \tilde{y}^T J \tilde{y} \geq \hat{y}^T A_{\bar{G}} \hat{y} > 1 - \frac{1}{\alpha(G)},$$

which yields the following:

$$\tilde{y}^T J \tilde{y} < \frac{1}{\alpha(G)},$$

Then, from the Lemma:

$$\frac{1}{|\{i : \tilde{y}_i > 0\}|} \leq \tilde{y}^T J \tilde{y} < \frac{1}{\alpha(G)}.$$

Note that $I = \{i : \tilde{y}_i > 0\}$ is an independent set and the last inequality implies $|I| > \alpha(G)$, which contradicts the definition of $\alpha(G)$. Thus, $\max_{y \in \mathcal{F}} y^T A_{\bar{G}} y = 1 - \frac{1}{\alpha(G)}$. □

2.4 A Generalization for Dominating Sets

For a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$, let $l = (k_1, \ldots, k_n)$ be a vector of integers such that $1 \leq k_i \leq d_i$ for $i \in V$, where d_i is the degree of vertex $i \in V$. An l-dominating set [113] is a set $D_l \subset V$ such that every vertex $i \in V \setminus D_l$ has at least k_i neighbors in D_l. The l-domination number $\gamma_l(G)$ of G is the cardinality of a smallest l-dominating set of G.

For $k_1 = \cdots = k_n = 1$, l-domination corresponds to the usual definition of domination. The domination number $\gamma(G)$ of G is the cardinality of a smallest dominating set of G. If $k_i = d_i$ for $i = 1, \ldots, n$, then $I = V \setminus D_l$ is an independent set and $\gamma_d(G) = n - \alpha(G)$ with $d = (d_1, \ldots, d_n)$.

The following theorem characterizes the domination number.

Theorem 2.5. *The domination number can be expressed by*

$$\gamma_l(G) = \min_{0 \leq x_i \leq 1, i=1,\ldots,n} f_l(x) = \min_{0 \leq x_i \leq 1, i=1,\ldots,n} \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} (1 - x_i) \times \left(\sum_{p=0}^{k_i - 1} \sum_{\{i_1, \ldots i_p\} \subset N(i)} \prod_{m \in \{i_1, \ldots, i_p\}} x_m \prod_{m \in N(i) \setminus \{i_1, \ldots, i_p\}} (1 - x_m) \right). \quad (2.20)$$

Proof. Denote the objective function by $g(G)$, *i.e.*,

$$g(G) = \min_{0\le x_i\le 1, i=1,\dots,n} \sum_{i=1}^{n} x_i \; + \; \sum_{i=1}^{n} (1-x_i) \times \left(\sum_{p=0}^{k_i-1} \sum_{\{i_1,\dots i_p\}\subset N(i)} \prod_{m\in\{i_1,\dots,i_p\}} x_m \prod_{m\in N(i)\setminus\{i_1,\dots,i_p\}} (1-x_m) \right). \quad (2.21)$$

We want to show that $\gamma_l(G) = g(G)$.

We first show that (2.21) always has an optimal 0-1 solution. Since $f_l(x)$ is a continuous function and $[0,1]^n = \{(x_1, x_2, \dots, x_n) : 0 \le x_i \le 1, i = 1, \dots, n\}$ is a compact set, there always exists $x^* \in [0,1]^n$ such that $f_l(x^*) = \min_{0\le x_i\le 1, i=1,\dots,n} f_l(x)$. The statement follows from linearity of $f_l(x)$ with respect to each variable.

Now we show that $g(G) \le \gamma_l(G)$. Assume $\gamma_l(G) = m$. Set

$$x_i^l = \begin{cases} 1, & \text{if } i \in D_l; \\ 0, & \text{otherwise.} \end{cases}$$

Then, $g(G) = \min_{0\le x_i\le 1, i=1,\dots,n} f_l(x) \le f_l(x^l) = m = \gamma_l(G)$.

Finally, we show that $g(G) \ge \gamma_l(G)$. Since (2.21) always has an optimal 0-1 solution, then $g(G)$ must be integer. Assume $g(G) = m$. Take an optimal 0-1 solution x^* of (2.20), such that the number of 1's is maximum among all 0-1 optimal solutions. Without loss of generality we can assume that this solution is $x_1^* = x_2^* = \cdots = x_r^* = 1$; $x_{r+1}^* = x_{r+2}^* = \cdots = x_n^* = 0$, for some r. Let

$$Q_i(x) = \sum_{p=0}^{k_i-1} \sum_{\{i_1,\dots i_p\}\subset N(i)} \prod_{m\in\{i_1,\dots,i_p\}} x_m \prod_{m\in N(i)\setminus\{i_1,\dots,i_p\}} (1-x_m);$$

$$Q(x) = \sum_{i=r+1}^{n} (1-x_i) \times \left(\sum_{p=0}^{k_i-1} \sum_{\{i_1,\dots i_p\}\subset N(i)} \prod_{m\in\{i_1,\dots,i_p\}} x_m \prod_{m\in N(i)\setminus\{i_1,\dots,i_p\}} (1-x_m) \right).$$

Let

$$D_l = \{i : x_i^* = 1\}.$$

We have

$$r + Q(x^*) = m.$$

From the last expression and nonnegativity of $Q(x)$ it follows that $|D_l| = r \leq m$.

We want to show that D_l is an l-dominating set. Assume that it is not. Let $S = \{i \in V \setminus D_l : |N(i) \cap D_l| < k_i\}$. Then $S \neq \emptyset$ and $\forall i \in S : \ Q_i(x^*) \geq 1$. Note, that changing $x_i^*, i \in S$ from 0 to 1 will increase $\sum_{i=1}^{n} x_i^*$ by 1 and will decrease $Q(x^*)$ by at least 1, thus it will not increase the objective function.

Consider $D_l^* = D_l \cup S$ and build x' as follows:

$$x_i' = \begin{cases} 1, & \text{if } i \in D_l^*; \\ 0, & \text{otherwise.} \end{cases}$$

Then $f_l(x') \leq f_l(x^*)$ and $|\{i : x_i' = 1\}| > |\{i : x_i^* = 1\}|$, which contradicts the assumption that x^* is an optimal solution with the maximum number of 1's. Thus, D_l is an l-dominating set with cardinality $r \leq m = g(G)$ and therefore $\gamma_l(G) \leq g(G)$. This concludes the proof of the theorem. □

Corollary 2.4. *For the case in which $k_1 = \cdots = k_n = 1$, we have*

$$\gamma(G) = \min_{0 \leq x_i \leq 1, i=1,\ldots,n} \sum_{i=1}^{n} \left(x_i + (1 - x_i) \prod_{(i,j) \in E} (1 - x_j) \right).$$

Corollary 2.5. *For the case $k_i = d_i, i = 1, \ldots, n$, the result of Theorem 2.2 follows.*

Proof. It can be shown by induction for $|N(i)|$, that

$$\sum_{p=0}^{d_i - 1} \sum_{\{i_1, \ldots i_p\} \subset N(i)} \prod_{m \in \{i_1, \ldots, i_p\}} x_m \prod_{m \in N(i) \setminus \{i_1, \ldots, i_p\}} (1 - x_m) = 1 - \prod_{j \in N(i)} x_j.$$

Thus,

$$\alpha(G) = n - \min_{0 \le x_i \le 1, i=1,\ldots,n} \sum_{i=1}^{n} \left(x_i + (1 - x_i)(1 - \prod_{(i,j) \in E} x_j) \right) =$$

$$\max_{0 \le x_i \le 1, i=1,\ldots,n} \sum_{i=1}^{n} (1 - x_i) \prod_{(i,j) \in E} x_j.$$

□

CHAPTER 3
HEURISTICS FOR THE MAXIMUM INDEPENDENT SET PROBLEM

In this chapter, we present several heuristics for the maximum independent set problem using continuous formulations. These results have previously appeared in several publications [3, 54].

3.1 Heuristic Based on Formulation (**P1**)

In this section, we discuss an algorithm for finding a maximal independent set based on formulation (**P1**) presented in Section 2.3. As pointed out before, the function $F(x)$ is linear with respect to each variable, so $A_i(x)$ and $B_i(x)$ can be computed for any $i \in \{1, 2, \ldots, n\}$. To produce a maximal independent set using $F(x)$, first let $x^0 \in [0, 1]^n$ be any starting point. The procedure described below produces a sequence of n points $x^1, x^2, \ldots, x^n$ such that x^n corresponds to a maximal independent set. Let $\mathcal{V} = \{1, 2, \ldots, n\}$ and consider some $i \in \mathcal{V}$. From (2.9–2.12) it follows that if we set

$$x_i^1 = \begin{cases} 0, & \text{if } A_i(x^0) > B_i(x^0); \\ 1, & \text{otherwise.} \end{cases}$$

and $x_j^1 = x_j^0$, if $j \neq i$, we obtain for the point $x^1 = (x_1^1, x_2^1, \ldots, x_n^1)$ that $F(x^1) \geq F(x^0)$.

If we update $\mathcal{V} = \mathcal{V} \setminus \{i\}$, we can construct the next point x^2 from x^1 in the same manner. Running this procedure n times, we obtain a point x^n which satisfies the inequality

$$F(x^n) \geq F(x^0).$$

The following theorem states that x^n has an independent set associated with it.

Theorem 3.1. *If $I = \{i \in \{1, 2, \ldots, n\} : x_i^n = 0\}$, then I is an independent set.*

Proof. Consider any $(i,j) \in E$. We need to show, that $\{i,j\}$ is not a subset of I. Without loss of generality, assume that we check x_i on the k-th iteration of the above procedure. If $x_i^k = 1$, then $i \notin I$. Alternatively, if $x_i^k = 0$, *i.e.*, $i \in I$, we need to show that $j \notin I$. Let $l > k$ be an iteration on which we check x_j. Then

$$A_j(x^{l-1}) = \prod_{(i,j)\in E} x_i = 0,$$

and therefore $A_j(x^{l-1}) \leq B_j(x^{l-1})$ and $x_j^l = 1$, which implies that $j \notin I$. □

From the discussion above, we have the following algorithm to find a maximal independent set.

Algorithm 1:

INPUT: $x^0 \in [0,1]^n$

OUTPUT: Maximal independent set I

0. $v := x^0$;
1. **for** $i = 1, \ldots, n$ **do if** $A_i(v) > B_i(v)$ **then** $v_i := 0$, **else** $v_i := 1$;
2. **for** $i = 1, \ldots, n$ **do if** $A_i(v) = 1$ **then** $v_i := 0$;
3. $I = \{i \in \{1, 2, \ldots, n\} : v_i = 0\}$;

END

Theorem 3.2. *Algorithm 1 is correct.*

Proof. We have already discussed steps 1 and 3 of the algorithm which guarantee that an independent set is produced. We need to show that step 2 guarantees that the independent set is maximal. Indeed, assume that after running step 1 we have, for some index i, $v_i = 1$ and

$$A_i(v) = \prod_{(i,j)\in E} v_j = 1.$$

This means that neither i nor any node from the neighborhood of i is included in the independent set that we obtained after step 1. Thus, we can increase the cardinality of this set including i in it by setting $v_i = 0$. □

The time complexity of the proposed algorithm is $O(\Delta^2 n)$, since $A_i(v)$ and $B_i(v)$ can be calculated in $O(\Delta^2)$ time.

3.2 Heuristic Based on Formulation (**P2**)

We now focus our attention on an algorithm, similar to Algorithm 1, based on formulation (**P2**).

Algorithm 2:

INPUT: $x^0 \in [0,1]^n$

OUTPUT: Maximal independent set I

0. $v := x^0$;
1. **for** $i = 1, \ldots, n$ **do if** $\sum_{(i,j)\in E} v_j < 1$ **then** $v_i := 1$, **else** $v_i := 0$;
2. **for** $i = 1, \ldots, n$ **do if** $\sum_{(i,j)\in E} v_j = 0$ **then** $v_i := 1$;
3. $I = \{i \in \{1, 2, \ldots, n\} : v_i = 1\}$;

END

Theorem 3.3. *Algorithm 2 is correct.*

Proof. Algorithm 2 is similar to Algorithm 1. In step 1 it finds an independent set $I_1 = \{i \in \{1, 2, \ldots, n\} : v_i = 1\}$. Set I_1 is independent because after step 1 we have that $\forall (i,j) \in E$ such that $i < j$, if $v_i = 1$ then $\sum_{(j,k)\in E} v_k \geq 1$ and $v_j = 0$. If I_1 is not a maximal independent set, then there exists i such that $v_i + \sum_{(i,j)\in E} v_j = 0$. We can increase the cardinality of I_1 by one, including i in it, by setting $v_i = 1$ in step 2. The resulting set is independent, because $\sum_{(i,j)\in E} v_j = 0$, which requires that $\forall j$ such that $(i,j) \in E : \ v_j = 0$. Thus no neighbors of i are included in the set. □

The time complexity of this algorithm is $O(\Delta n)$.

3.3 Examples

The algorithms presented build a maximal independent set from any given point $x^0 \in [0,1]^n$ in polynomial time. The output, however, depends on the choice of x^0. An interesting question that arises is how to choose such input point x^0, so that a maximum independent set can be found? The problem of finding such a point cannot be solved in polynomial time, unless P = NP. However, we can show on simple examples that in some cases even starting with a "bad" starting point x^0 ($F(x^0) \leq 1$ or $H(x^0) \leq 1$), we obtain a maximum independent set as the output.

3.3.1 Example 1

Consider the graph in Figure 3–1. For this example

$$
\begin{aligned}
x &= (x_1, x_2, x_3, x_4) \in [0,1]^4; \\
F(x) &= (1-x_1)x_2x_3x_4 + (1-x_2)x_1 + (1-x_3)x_1 + (1-x_4)x_1; \\
A_1(x) &= x_2x_3x_4; \\
B_1(x) &= (1-x_2) + (1-x_3) + (1-x_4); \\
A_2(x) &= A_3(x) = A_4(x) = x_1; \\
B_2(x) &= (1-x_1)x_3x_4; \\
B_3(x) &= (1-x_1)x_2x_4; \\
B_4(x) &= (1-x_1)x_2x_3; \\
H(x) &= x_1 + x_2 + x_3 + x_4 - x_1x_2 - x_1x_3 - x_1x_4.
\end{aligned}
$$

Consider Algorithm 1 with $x^0 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Since $A_1(x^0) = \frac{1}{8}$, $B_1(x^0) = \frac{3}{2}$, and since $\frac{1}{8} < \frac{3}{2}$, the next point is $x^1 = (1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$.

Next, $A_2(x^1) = 1$, $B_2(x^1) = 0$, and $x^2 = (1, 0, \frac{1}{2}, \frac{1}{2})$. After two more iterations we get $x^4 = (1,0,0,0)$ with $I = \{2,3,4\}$, which is the maximum independent set of the given graph. We have $|I| = 3$, $F(x^0) = \frac{13}{16}$, and the objective function increase is $|I| - F(x^0) = \frac{35}{16}$.

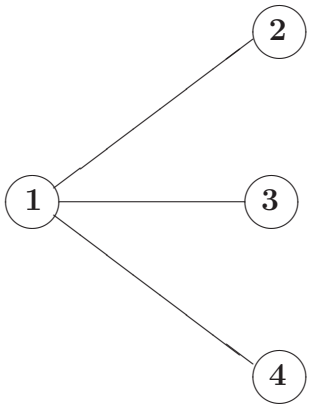

Figure 3–1: Illustration to Example 1

For Algorithm 2, starting with $x^0 = (1, 1, 1, 1)$, for which $H(x^0) = 1$, we obtain the maximum independent set after step 1. Note, that the Caro–Wei bound for this graph is $\frac{7}{4}$.

3.3.2 Example 2

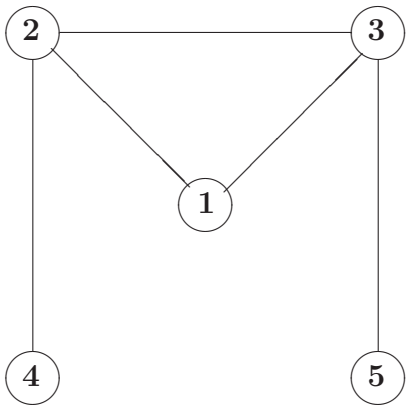

Figure 3–2: Illustration to Example 2

For the graph in Figure 3–2, we have $x = (x_1, x_2, x_3, x_4, x_5) \in [0, 1]^5$ and

$$F(x) = (1 - x_1)x_2x_3 + (1 - x_2)x_1x_3x_4 + (1 - x_3)x_1x_2x_5 + (1 - x_4)x_2 + (1 - x_5)x_3.$$

Applying Algorithm 1 for this graph with initial point $x^0 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, we obtain, at the end of step 1, the solution $x^5 = (1, 1, 1, 0, 0)$, which corresponds to the independent set $I = \{4, 5\}$. At the end of step 2, the solution is $(0, 1, 1, 0, 0)$, which corresponds to the maximum independent set $I = \{1, 4, 5\}$. For this case we have $|I| = 3$, $F(x^0) = \frac{7}{8}$, and the objective function improvement is $\frac{17}{8}$. With the initial point $x^0 = (0, 0, 0, 0, 0)$, $H(x^0) = 0$, and Algorithm 2 finds the maximum independent set after step 1. For this example, the Caro–Wei bound equals $\frac{11}{6}$.

3.3.3 Example 3

This example shows, that the output of Algorithm 1 and Algorithm 2 depends not only on initial point x^0, but also on the order in which we examine variables in steps 1 and 2. For example, if we consider the graph from Figure 3–1 with a different order of nodes (as in Figure 3–3), and run Algorithm 1 and Algorithm 2 for this graph with initial point $x^0 = (1, 1, 1, 1)$, we obtain $I = \{4\}$ as output for both algorithms. Note that, for the graph from Figure 3–1, both outputs would be the maximum independent set of the graph.

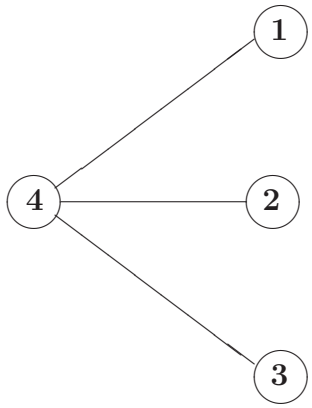

Figure 3–3: Illustration to Example 3

As Example 3 shows, we may be able to improve both algorithms by including two procedures (one for each step) which, given a set of remaining nodes, choose a node to be examined next. Consider Algorithm 2. Let `index1()` and `index2()` be

procedures for determining the order of examining nodes on step 1 and step 2 of the algorithm, respectively. Then we have the following algorithm.

Algorithm 3:

INPUT: $x^0 \in [0,1]^n$

OUTPUT: Maximal independent set I

0. $v := x^0$; $V_1 := V$; $V_2 := V$;
1. **while** $V_1 \neq \emptyset$ **do**
   (a) $k =$ `index1`(V_1);
   (b) **if** $\sum_{(k,j)\in E} v_j < 1$ **then** $v_k := 1$, **else** $v_k := 0$;
   (c) $V_1 := V_1 \setminus \{v_k\}$;
2. **while** $V_2 \neq \emptyset$ **do**
   (a) $k =$ `index2`(V_2);
   (b) **if** $\sum_{(k,j)\in E} v_j = 0$ **then** $v_k := 1$;
   (c) $V_2 := V_2 \setminus \{v_k\}$;
3. $I = \{i \in \{1, 2, \ldots, n\} : v_i = 1\}$;

END

In general, procedures `index1()` and `index2()` can be different. We propose the same procedure `index()` for `index1()` and `index2()`:

$$\texttt{index}(V_0) = \text{argmax}_{k\in V_0}\{\sum_{(k,j)\in E} v_j\}$$

breaking ties in favor of the node with the smallest neighborhood in $V \setminus V_0$ and at random if any nodes remain tied.

3.3.4 Computational Experiments

This section presents computational results of the algorithms described in the previous section. We have tested the algorithms on some of the DIMACS clique instances which can be downloaded from the URL

Table 3–1: Results on benchmark instances: Algorithm 1, random x^0.

Name	n	Density	$\omega(G)$	Sol. Found	Average Sol.	Time(sec.)
MANN_a9	45	0.927	16	15	13.05	0.01
MANN_a27	378	0.990	126	113	68.16	8.14
MANN_a45	1035	0.996	345	283	198.76	243.94
c-fat200-1	200	0.077	12	12	10.80	33.72
c-fat200-2	200	0.163	24	24	22.22	31.43
c-fat200-5	200	0.426	58	58	57.14	17.60
hamming6-2	64	0.905	32	32	30.17	0.06
hamming6-4	64	0.349	4	4	3.72	0.37
hamming8-2	256	0.969	128	128	119.54	6.45
hamming8-4	256	0.639	16	16	10.87	38.22
hamming10-2	1024	0.990	512	503	471.17	233.45
johnson8-2-4	28	0.556	4	4	4.00	0.01
johnson8-4-4	70	0.768	14	14	13.02	0.17
johnson16-2-4	120	0.765	8	8	8.00	1.13
johnson32-2-4	496	0.879	16	16	16.00	186.77
keller4	171	0.649	11	7	7.00	9.72
san200_0.9_1	200	0.900	70	53	39.08	0.14
san200_0.9_2	200	0.900	60	34	29.41	0.14
san200_0.9_3	200	0.900	44	31	27.58	0.23
san400_0.9_1	400	0.900	100	53	46.18	4.54

http://dimacs.rutgers.edu/Challenges/. All algorithms are programmed in C and compiled and executed on an Intel Pentium III 600 Mhz PC under MS Windows NT.

First, each algorithm was executed 100 times with random initial solutions uniformly distributed in the unit hypercube. The results of these experiments are summarized in Tables 3–1, 3–2, and 3–3. The columns "Name," "n," "Density," and "$\omega(G)$" represent the name of the graph, the number of its vertices, its density, and its clique number, respectively. This information is available from the DIMACS web site. The column "Sol. Found" contains the size of the largest clique found after 100 runs. The columns "Average Sol." and "Time(sec.)" contain average solution and average CPU time (in seconds) taken over 100 runs of an algorithm, respectively.

Table 3–2: Results on benchmark instances: Algorithm 2, random x^0.

Name	n	Density	$\omega(G)$	Sol. Found	Average Sol.	Time(sec.)
MANN_a9	45	0.927	16	16	14.45	0.01
MANN_a27	378	0.990	126	120	119.16	1.89
MANN_a45	1035	0.996	345	334	331.66	36.20
c-fat200-1	200	0.077	12	12	12.00	0.33
c-fat200-2	200	0.163	24	24	22.59	0.32
c-fat200-5	200	0.426	58	58	57.85	0.28
hamming6-2	64	0.905	32	32	21.44	0.01
hamming6-4	64	0.349	4	4	2.38	0.01
hamming8-2	256	0.969	128	121	90.95	1.07
hamming8-4	256	0.639	16	16	9.71	1.43
hamming10-2	1024	0.990	512	494	410.92	46.46
johnson8-2-4	28	0.556	4	4	4.00	0.01
johnson8-4-4	70	0.768	14	14	9.99	0.01
johnson16-2-4	120	0.765	8	8	8.00	0.04
johnson32-2-4	496	0.879	16	16	16.00	4.57
keller4	171	0.649	11	7	7.00	0.13
san200_0.9_1	200	0.900	70	61	38.86	0.14
san200_0.9_2	200	0.900	60	32	29.09	0.13
san200_0.9_3	200	0.900	44	30	26.93	0.18
san400_0.9_1	400	0.900	100	54	44.20	2.68

Table 3–4 contains the results of computations for all the algorithms with the initial solution x^0, such that $x^0_i = 0$, $i = 1, \ldots, n$. In this table, "A3" stands for Algorithm 3. As can be seen from the tables, the best solutions for almost all instances obtained during the experiments can be found among the results for Algorithm 3 with $x^0_i = 0$, $i = 1, \ldots, n$ (see Table 3–4).

In Table 3–5 we compare these results with results for some other continuous based heuristics for the maximum clique problem taken from the paper of Bomze et al. [39]. The columns "ARH", "PRD($\frac{1}{2}$)", "PRD(0)" and "CBH" contain the size of a clique found using the annealed replication heuristic of Bomze et al. [39], the plain replicator dynamics applied for two different parameterizations (with parameters $\frac{1}{2}$ and 0) of the Motzkin–Straus formulation [40], and the continuous-based heuristic

Table 3–3: Results on benchmark instances: Algorithm 3, random x^0.

Name	n	Dens.	$\omega(G)$	Sol. Found	Average Sol.	Time(sec.)
MANN_a9	45	0.927	16	16	14.98	0.01
MANN_a27	378	0.990	126	121	119.21	4.32
MANN_a45	1035	0.996	345	334	331.57	87.78
c-fat200-1	200	0.077	12	12	11.64	0.48
c-fat200-2	200	0.163	24	24	22.47	0.47
c-fat200-5	200	0.426	58	58	57.25	0.42
hamming6-2	64	0.905	32	32	27.49	0.02
hamming6-4	64	0.349	4	4	4.00	0.02
hamming8-2	256	0.969	128	128	100.78	0.80
hamming8-4	256	0.639	16	16	12.49	1.13
hamming10-2	1024	0.990	512	512	359.53	90.11
johnson8-2-4	28	0.556	4	4	4.00	0.01
johnson8-4-4	70	0.768	14	14	11.22	0.02
johnson16-2-4	120	0.765	8	8	8.00	0.09
johnson32-2-4	496	0.879	16	16	16.00	10.20
keller4	171	0.649	11	9	7.54	0.28
san200_0.9_1	200	0.900	70	46	45.03	0.37
san200_0.9_2	200	0.900	60	37	34.94	0.38
san200_0.9_3	200	0.900	44	32	26.86	0.37
san400_0.9_1	400	0.900	100	51	50.01	2.54

of Gibbons et al. [97], respectively. The column "A3(0)" represents the results for Algorithm 3 with $x_i^0 = 0$, $i = 1, \ldots, n$.

3.4 Heuristic Based on Optimization of a Quadratic Over a Sphere

Recall the Motzkin-Strauss formulation of the maximum clique problem. Let A_G be the adjacency matrix of G and let e be the n-dimensional vector with all components equal to 1.

Theorem 3.4 (Motzkin-Straus). *The optimal value of the following quadratic program*

$$\max f(x) = \frac{1}{2} x^T A_G x, \tag{3.1}$$

Table 3–4: Results on benchmark instances: Algorithms 1–3, $x_i^0 = 0$, for $i = 1, \ldots, n$.

Name	n	Dens.	$\omega(G)$	Sol. Found			Time(sec.)		
				A1	A2	A3	A1	A2	A3
MANN_a9	45	0.927	16	16	9	16	0.01	0.01	0.01
MANN_a27	378	0.990	126	125	27	125	8.17	2.01	3.72
MANN_a45	1035	0.996	345	340	45	342	170.31	36.42	79.15
c-fat200-1	200	0.077	12	12	12	12	34.13	0.48	1.59
c-fat200-2	200	0.163	24	24	24	24	33.15	0.79	0.92
c-fat200-5	200	0.426	58	58	58	58	21.63	0.48	12.02
hamming6-2	64	0.905	32	32	32	32	0.23	0.03	0.05
hamming6-4	64	0.349	4	2	4	4	0.42	0.03	0.06
hamming8-2	256	0.969	128	128	128	128	4.57	0.89	1.32
hamming8-4	256	0.639	16	16	16	16	39.07	1.98	1.79
hamming10-2	1024	0.990	512	512	512	512	508.30	35.97	82.19
johnson8-2-4	28	0.556	4	4	4	4	0.01	0.01	0.01
johnson8-4-4	70	0.768	14	14	14	14	0.33	0.02	0.03
johnson16-2-4	120	0.765	8	8	8	8	1.56	0.09	0.19
johnson32-2-4	496	0.879	16	16	16	16	191.24	5.97	9.81
keller4	171	0.649	11	7	7	11	7.99	0.18	0.43
san200_0.9_1	200	0.900	70	42	43	47	3.19	0.27	0.56
san200_0.9_2	200	0.900	60	29	36	40	3.21	0.27	0.54
san200_0.9_3	200	0.900	44	29	21	34	3.05	1.04	0.48
san400_0.9_1	400	0.900	100	52	35	75	88.47	3.28	6.55

Table 3–5: Results on benchmark instances: comparison with other continuous based approaches. $x_i^0 = 0, i = 1, \ldots, n$.

Name	n	Dens.	$\omega(G)$	Sol. Found				
				ARH	PRD($\frac{1}{2}$)	PRD(0)	CBH	A3(0)
MANN_a9	45	0.927	16	16	12	12	16	16
MANN_a27	378	0.990	126	117	117	117	121	125
keller4	171	0.649	11	8	7	7	10	11
san200_0.9_1	200	0.900	70	45	45	45	46	47
san200_0.9_2	200	0.900	60	39	36	35	36	40
san200_0.9_3	200	0.900	44	31	32	33	30	34
san400_0.9_1	400	0.900	100	50	40	55	50	75

subject to

$$e^T x = 1, \tag{3.1a}$$

$$x \geq 0. \tag{3.1b}$$

is given by

$$\frac{1}{2}\left(1-\frac{1}{\omega(G)}\right),$$

where $\omega(G)$ is the clique number of G.

Gibbons et al. [98] extended this result by providing a characterization of maximal cliques in terms of local solutions. Moreover, they studied optimality conditions of the Motzkin-Straus program and investigated the properties of a newly introduced parametrization of the corresponding quadratic program.

Gibbons et al. [97] proposed another, more tractable continuous formulation of the maximum independent set problem.

Theorem 3.5 (Gibbons et al.). *Consider the following optimization problem:*

$$V(k) = \min \frac{1}{2}x^T A_G x + (\sum_{i=1}^{n} x_i - 1)^2 \tag{3.2}$$

subject to

$$\sum_{i=1}^{n} x_i^2 \leq \frac{1}{k}, \tag{3.2a}$$

$$x \geq 0. \tag{3.2b}$$

If $\bar{x}$ is a solution to (3.2) then $V(k) = 0$ iff there exists an independent set I in G such that $|I| \geq k$.

Based on this theorem the authors proposed a heuristic for the maximum clique problem, which uses the global minimum of the problem (3.2)-(3.2a) to extract a maximal independent set from the graph. The approach that we propose in this section is based on the same formulation as in Theorem 3.5. Moreover, we also adopt the idea of using the problem of optimization of the same quadratic over a sphere to find a large independent set, but in our approach we use information about all stationary points of the problem (3.2)-(3.2a). We will use the computational results of Gibbons et al. [97] to estimate the efficiency of our approach in subsection 3.4.3.

3.4.1 Optimization of a Quadratic Function Over a Sphere

Although optimization of a quadratic function subject to linear constraints is NP-hard, quadratic programming subject to an ellipsoid constraint is polynomially solvable. In fact, Ye [203] proved that it can be solved in $O(log(log(1/\epsilon)))$ iterations, where ϵ is the error tolerance. Due to its application in trust region methods of nonlinear optimization, the problem of optimization of a quadratic function over a sphere is a well studied one [89, 110, 153, 164]. In this section we will discuss some results related to this problem.

Consider the following problem:

$$\min f(x) = 2c^T x + x^T Q x, \tag{3.3}$$

subject to

$$||x||_2 = r, \tag{3.3a}$$

where $x \in \mathbb{R}^n$ is the vector of variables, Q is a symmetric $n \times n$ matrix, $c \in \mathbb{R}^n$ is an arbitrary fixed vector, $r > 0$ is some number, and $|| \cdot ||_2$ is the second norm in $\mathbb{R}^n$, *i.e.*, for any $x \in \mathbb{R}^n$: $||x||_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$.

We are interested in all stationary points of the problem (3.3), which can be found from the Lagrangian $\mathcal{L}(x, \lambda)$ of $f(x)$. We have

$$\mathcal{L}(x, \lambda) = f(x) - \lambda(||x||_2^2 - r^2), \tag{3.4}$$

where λ is the Lagrange multiplier of the constraint of (3.3), and the stationarity conditions:

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial x_i} = 0, i = 1, \ldots, n; \tag{3.5}$$

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial \lambda} = 0. \tag{3.6}$$

The set of conditions (3.5) is equivalent to the system

$$(Q - \lambda I)x = -c, \tag{3.7}$$

where I stands for the $n \times n$ identity matrix. The condition (3.6) is simply the constraint (3.3a). Therefore, x is a stationary point of problem (3.3)-(3.3a) if and only if there exists a real number $\lambda = \lambda(x')$ such that

$$(Q - \lambda I)x = -c; \tag{3.8}$$

$$\sum_{i=1}^{n} x_i^2 = r^2. \tag{3.9}$$

The set Λ of all real λ for which there exists an x such that (3.8) and (3.9) are satisfied, is called the spectrum of the problem (3.3) - (3.3a).

Applying the eigenvalue decomposition for the matrix Q we have

$$Q = R\,\mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)R^T,$$

where $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ are the eigenvalues of the matrix Q sorted in nondecreasing order, R is the matrix having a corresponding orthonormal set of eigenvectors as its columns. Transforming the vectors x and c to the basis R of the eigenvectors of Q, we obtain

$$\bar{x} = R^T x, \;\; \bar{c} = R^T c.$$

Then (3.7) can be rewritten as

$$(\lambda_i - \lambda)\bar{x}_i = -\bar{c}_i, \;\; i = 1, \ldots, n. \tag{3.10}$$

Assuming that $\lambda \neq \lambda_i$, we have

$$\bar{x}_i = -\frac{\bar{c}_i}{\lambda_i - \lambda}. \tag{3.11}$$

Rewriting expression (3.9) in the basis associated with the rows of matrix R and substituting (3.11) into it, we obtain the so-called secular equation

$$\sum_{i=1}^{n} \frac{\bar{c}_i^2}{(\lambda_i - \lambda)^2} - r^2 = 0. \tag{3.12}$$

The left-hand side of the last expression is a univariate function consisting of $n+1$ pieces each of which is continuous and convex. This implies that it can have at most two roots at any interval $(\lambda_i, \lambda_{i+1}),\ i = 0, \ldots, n$ corresponding to the i^{th} piece, where by λ_0 and λ_{n+1} we mean $-\infty$ and $+\infty$ respectively. Moreover, the spectrum Λ includes the set Λ^0 of all solutions to the secular equation (3.12) and $\Lambda \setminus \Lambda^0 = \{\lambda_i : \bar{c}_i = 0\}$ (so, $\Lambda = \Lambda^0$ if $\{\lambda_i : \bar{c}_i = 0\} = \emptyset$).

Forsythe and Golub [89] have shown that the global minimum of problem (3.3) corresponds to the smallest element of the spectrum $\lambda^* = \min \Lambda$. It implies that if $\bar{c}_i \neq 0$ then $\lambda^* = \min \Lambda^0$.

We mention one more result, which relates the maximum independent set problem to the problem of optimization of a quadratic over a sphere. Recently, de Angelis et al. [70] have shown that in the problem of optimizing a quadratic over a sphere of radius r a global solution $x^*(r)$ can be chosen, which depends continuously upon r. The last fact gives us a new point of view on the maximum independent set problem. Namely, it can be formulated as a problem of optimization of a quadratic function over a sphere with an unknown radius:

Lemma 3.1. *There exists* $r \in \left[\frac{1}{2}, \frac{\sqrt{n}}{2}\right]$ *such that*

$$\alpha(G) = \max f_\alpha(x) = \sum_{i=1}^{n} x_i - \sum_{(i,j)\in E} x_i x_j \tag{3.13}$$

$$\textit{subject to} \quad ||x - \tfrac{1}{2}e||_2 = r,$$

To show the validity of the lemma, consider formulation (**P2**) and take two spheres S_1 (of radius $\frac{1}{2}$) and S_2 (of radius $\frac{\sqrt{n}}{2}$) inscribed in and circumscribed over

the unit hypercube respectively. We have

$$\max_{x \in S_1} f_\alpha(x) \leq \alpha(G) = \max_{x \in [0,1]^n} f_\alpha(x) \leq \max_{x \in S_2} f_\alpha(x)$$

and by the Intermediate Value Theorem the lemma is correct.

3.4.2 The Heuristic

In this subsection, we present the heuristic for the maximum independent set problem which employs the formulation mentioned in Theorem 3.5, and utilizes the methods similar to those in the so-called QUALEX heuristic [53]. In QUALEX, the feasible region of the formulation (**P2**) is changed from the unit hypercube to the sphere with the center $x^0 = 0$ and a certain radius r, depending on the input graph, resulting in the following problem:

$$\max f_\alpha(x) = \sum_{i=1}^{n} x_i - \sum_{(i,j) \in E} x_i x_j \tag{3.14}$$

subject to

$$||x - x^0||_2^2 = r^2. \tag{3.14a}$$

In short, QUALEX can be regarded as a sophisticated greedy heuristic, which first finds a local solution using a straightforward greedy approach, and then attempts to find a better solution using information provided by the stationary points of the above problem.

We apply similar techniques for the problem which is obtained from (3.2)-(3.2b) by changing the inequality to the equality in (3.2a) and relaxing nonnegativity constraints (3.2b):

$$\min \frac{1}{2} x^T A_G x + (\sum_{i=1}^{n} x_i - 1)^2 \tag{3.15}$$

subject to

$$\sum_{i=1}^{n} x_i^2 = \frac{1}{k}. \tag{3.15a}$$

As the value of k we use the cardinality of a maximal independent set found by a simple greedy heuristic (numerical experiments suggest that varying k does not make results much better, but, naturally, requires more computational time). For each stationary point x of the problem (3.15)-(3.15a) we apply a greedy heuristic to find a maximal independent set I_x, which uses order of the vertices in the graph corresponding to the nonincreasing order of the components of x. The largest found I_x is reported as the output maximal independent set.

Below we present an outline of our algorithm.

Algorithm 4 [QSH]:

INPUT: Graph G with adjacency matrix A_G; $\alpha = 0$

OUTPUT: A maximal independent set I^* and its cardinality α

0. Apply the following greedy heuristic:
   (a) $G_0 = G$;
   (b) In G_0 recursively remove the neighborhood of a vertex with the minimum degree in the current graph until an empty graph with vertex set I is obtained;
   (c) **If** $|I| > \alpha$ **then begin** $I^* = I,\ \alpha = |I|$; **end**
   (d) **If** $G_0 \setminus I \neq \emptyset$ **then begin** $G_0 = G_0 \setminus I$; **go to** step (b); **end**.
1. **For** $k = \alpha$ **begin**

   For each interval $(\lambda_i, \lambda_{i+1})$ **begin**

   (a) Apply the techniques discussed in Section 3.4.1 for the problem (3.15)-(3.15a) to find the set of all stationary points S corresponding to $(\lambda_i, \lambda_{i+1})$;
   (b) For each $x \in S$ find I_x by the following procedure:

i. Order the components of x in nonincreasing order

$x_{i_1} \geq x_{i_2} \geq \ldots \geq x_{i_n}$;

ii. $I_x = \emptyset$;

iii. **For** $j = 1$ **to** n **begin**

If $N(i_j) \bigcap I_x = \emptyset$ then $I_x = I_x \bigcup \{i_j\}$

end

(c) $x^* = \text{argmax}\{|I_x| : x \in S\}$, $I = I_{x^*}$;

(d) If $|I| > \alpha$ then $I^* = I$ and $\alpha = |I|$;

end

end

The results of numerical experiments with the algorithm are reported in the next subsection.

3.4.3 Computational Experiments

This section presents results of computational experiments for the proposed heuristic. The algorithm has been tested on the complement graphs of some of the DIMACS clique instances which can be downloaded from the URL `http://dimacs.rutgers.edu/Challenges/`. The performance of the algorithm (denoted by QSH) on the benchmarks is summarized in Tables 3–6 and 3–7. The columns "Name," "n," "Density," and "$\omega(G)$" represent the name of the graph, the number of its vertices, its density, and its clique number, respectively. This information is available from the DIMACS web site. Recall that we are working with complements of the considered graphs, and the densities are specified for the original (maximum clique) instances. The column "Sol. Found" contains the size of the clique found by each algorithm. In Table 3–6, sub-columns "CBH" and "QSH" represent the continuous based heuristic [97] and the heuristic proposed in this section, correspondingly. Finally, the column "Time" reports computational times for QSH in seconds obtained using the C function `time`. The results for CBH were taken from

Table 3–6: Results on benchmark instances, part I.

Name	n	Density	$\omega(G)$	Sol. Found		Time
				CBH	QSH	(sec.)
MANN_a9	45	0.927	16	16	16	0
MANN_a27	378	0.990	126	121	125	8
brock200_1	200	0.745	21	20	21	1
brock200_2	200	0.496	12	12	12	1
brock200_3	200	0.605	15	14	15	1
brock200_4	200	0.658	17	16	17	2
brock400_1	400	0.748	27	23	27	21
brock400_2	400	0.749	29	24	29	21
brock400_3	400	0.748	31	23	31	20
brock400_4	400	0.749	33	24	33	21
brock800_1	800	0.649	23	20	17	245
brock800_2	800	0.651	24	19	24	243
brock800_3	800	0.649	25	20	25	228
brock800_4	800	0.650	26	19	26	233
c-fat200-1	200	0.077	12	12	12	1
c-fat200-2	200	0.163	24	24	24	1
c-fat200-5	200	0.426	58	58	58	1
c-fat500-1	500	0.036	14	14	14	33
c-fat500-10	500	0.374	≥ 126	126	126	20
c-fat500-2	200	0.073	26	26	26	31
c-fat500-5	500	0.186	64	64	64	23
hamming6-2	64	0.905	32	32	32	0
hamming6-4	64	0.349	4	4	4	0
hamming8-2	256	0.969	128	128	128	2
hamming8-4	256	0.639	16	16	16	1
johnson8-2-4	28	0.556	4	4	4	0
johnson8-4-4	70	0.768	14	14	14	0
johnson16-2-4	120	0.765	8	8	8	0
johnson32-2-4	496	0.879	16	16	16	24

Gibbons et al. [97]. A modification of the C++ QUALEX code available on-line from Busygin's webpage [53] was used to obtain the results for QSH. The code was compiled and executed on an IBM computer with 2 PowerPC processors, 333 Mhz each.

In Table 3–8 we compare the performance of CBH and QSH on different types of graphs. In each row of this table the first column contains a name of the family of

Table 3–7: Results on benchmark instances, part II.

Name	n	Density	$\omega(G)$	Sol. Found		Time
				CBH	QSH	(sec.)
keller4	171	0.649	11	10	11	1
keller5	776	0.751	27	21	24	149
p_hat300-1	300	0.244	8	8	7	6
p_hat300-2	300	0.489	25	25	24	6
p_hat300-3	300	0.744	36	36	33	6
p_hat500-1	500	0.253	9	9	9	48
p_hat500-2	500	0.505	36	35	33	49
p_hat500-3	500	0.752	≥ 49	49	46	48
p_hat700-1	700	0.249	11	11	8	143
p_hat700-2	700	0.498	44	44	42	143
p_hat700-3	700	0.748	≥ 62	60	59	143
san200_0.7_1	200	0.700	30	15	30	1
san200_0.7_2	200	0.700	18	12	18	1
san200_0.9_1	200	0.900	70	46	70	1
san200_0.9_2	200	0.900	60	36	60	1
san200_0.9_3	200	0.900	44	30	35	1
san400_0.5_1	400	0.500	13	8	9	20
san400_0.7_1	400	0.700	40	20	40	21
san400_0.7_2	400	0.700	30	15	30	19
san400_0.7_3	400	0.700	22	14	16	21
san400_0.9_1	400	0.900	100	50	100	23
sanr200_0.7	200	0.700	18	18	15	1
sanr200_0.9	200	0.900	≥ 42	41	37	1
sanr400_0.5	400	0.500	13	12	11	20
sanr400_0.7_1	400	0.700	≥ 21	20	18	19

graphs, where each family consists of a group of graphs with the same letter names (in Tables 3–6 and 3–7 such families are separated by horizontal lines). In the next three columns "+" represents the number of instances from each family for which CBH found a better solution than QSH; "–" represents the number of instances for which QSH found a better solution than CBH; finally, "=" stands for the number of instances for which cardinalities of the independent sets found by the two algorithms were equal.

Table 3–8: Comparison of the results on benchmark instances.

Graphs family name	CBH vs QSH +	=	−
MANN_a	0	1	1
brock	1	1	10
c-fat	0	7	0
hamming	0	4	0
johnson	0	4	0
keller	0	0	2
p_hat	8	1	0
san	0	0	10
sanr	4	0	0
TOTAL	13	18	23

As one can see QSH performed better on four families (MANN_a, brock, keller, san) and did worse on two (p_hat and sanr). Analyzing the difference in quality of the results, one can notice a huge gap between the solutions obtained for san group. The results of numerical experiments suggest that QSH is superior to CBH and many other heuristics presented in the literature.

3.5 Concluding Remarks

In this chapter, we discussed several heuristics for the maximum independent set problem based on continuous formulations. In Sections 3.1 and 3.2, we offer three syntactically related algorithms for finding large maximal independent sets. A computational investigation of these algorithms shows their competitiveness.

In Section 3.4, we present a heuristic for the maximum independent set problem based on techniques used in optimization of a quadratic over a sphere. We show the validity of this approach by demonstrating the results of numerical experiments. We want to stress that the information provided by the stationary points other than points of global optimality may help us to find better independent sets in the corresponding graph.

In Lemma 3.1 we formulated the maximum independent set problem as a problem of optimization of a quadratic over a sphere with an unknown radius. In future research, when using similar techniques for combinatorial optimization problems, one should try to address the following issues:

- How to optimally choose the parameters of the quadratic objective and of the sphere used as the feasible set?
- What is the optimal way to extract the sought combinatorial object from the information provided by stationary (or optimal) points of the considered quadratic problem?

To answer these questions both theoretical and empirical studies are needed.

CHAPTER 4
APPLICATIONS IN MASSIVE DATA SETS

Massive data sets arise in a broad spectrum of scientific, engineering and commercial applications. These include government and military systems, telecommunications, medicine and biotechnology, astrophysics, finance, ecology, geographical information systems, etc [5]. Some of the wide range of problems associated with massive data sets are data warehousing, compression and visualization, information retrieval, clustering and pattern recognition, and nearest neighbor search. Handling these problems requires special interdisciplinary efforts in developing novel sophisticated techniques. The pervasiveness and complexity of the problems brought by massive data sets make it one of the most challenging and exciting areas of research for years to come.

In many cases, a massive data set can be represented as a very large graph with certain attributes associated with its vertices and edges. These attributes may contain specific information characterizing the given application. Studying the structure of this graph is important for understanding the structural properties of the application it represents, as well as for improving storage organization and information retrieval.

In this chapter, we first review current development in studying massive graphs. Then, in Section 4.2 we present our research concerning the market graph representing the U.S. stock markets. We will conclude with some remarks in Section 4.4. This chapter is based on joint publications with Boginski and Pardalos [32, 33].

4.1 Modeling and Optimization in Massive Graphs

In this section we discuss recent advances in modeling and optimization for massive graphs. As examples, call, Internet, and Web graphs will be used.

As before, by $G = (V, E)$ we will denote a simple undirected graph with the set of n vertices V and the set of edges E. A multi-graph is an undirected graph with multiple edges.

The *distance* between two vertices is the number of edges in the shortest path between them (it is equal to infinity for vertices representing different connected components). The *diameter* of a graph G is usually defined as the maximal distance between pairs of vertices of G. Note, that in the case of a disconnected graph the usual definition of the diameter would result in the infinite diameter, therefore the following definition is in order. By the diameter of a disconnected graph we will mean the maximum finite shortest path length in the graph (which is the same as the largest of diameters of the graph's connected components).

4.1.1 Examples of Massive Graphs

The call graph

Here we discuss an example of a massive graph representing telecommunications traffic data presented by Abello, Pardalos and Resende [4]. In this *call graph* the vertices are telephone numbers, and two vertices are connected by an edge if a call was made from one number to another.

Abello et al. [4] experimented with data from AT&T telephone billing records. To give an idea of how large a call graph can be we mention that a graph based on one 20-day period had 290 million vertices and 4 billion edges. The analyzed one-day call graph had 53,767,087 vertices and over 170 millions of edges. This graph appeared to have 3,667,448 connected components, most of them tiny; only 302,468 (or 8%) components had more than 3 vertices. A giant connected component with 44,989,297 vertices was computed. It was observed that the existence of a giant component resembles a behavior suggested by the random graphs theory of Erdös and Rényi, but by the pattern of connections the call graph obviously does not fit into this theory. This will be discussed in more detail in Subsection 4.1.3. The

maximum clique problem and problem of finding large quasi-cliques with prespecified density were considered in this giant component. These problems were attacked using a greedy randomized adaptive search procedure (GRASP) [85, 86]. In short, GRASP is an iterative method that at each iteration constructs, using a greedy function, a randomized solution and then finds a locally optimal solution by searching the neighborhood of the constructed solution. This is a heuristic approach which gives no guarantee about quality of the solutions found, but proved to be practically efficient for many combinatorial optimization problems. To make application of optimization algorithms in the considered large component possible, the authors use some suitable graph decomposition techniques employing external memory algorithms (see Subsection 4.1.2).

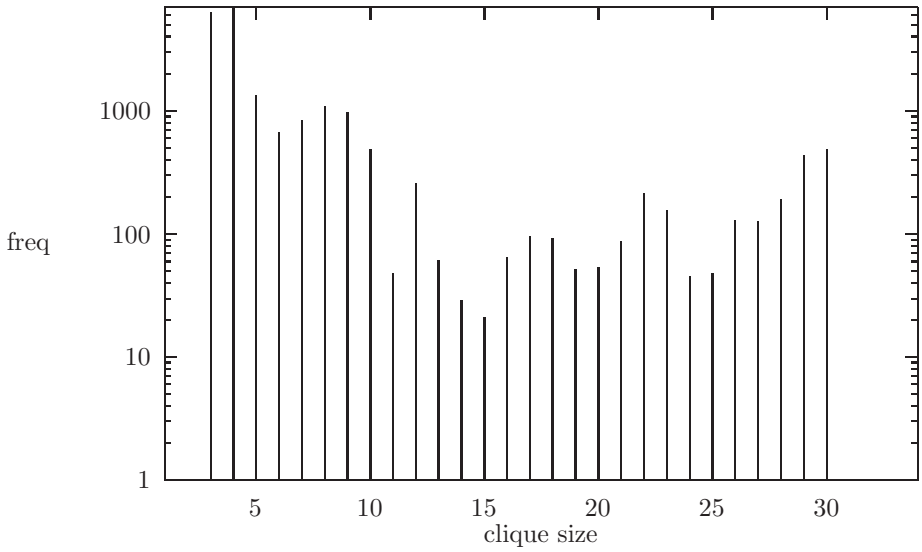

Figure 4–1: Frequencies of clique sizes in the call graph found by Abello et al. [4].

Abello et al. ran 100,000 GRASP iterations taking 10 parallel processors about one and a half days to finish. Of the 100,000 cliques generated, 14,141 appeared to be distinct, although many of them had vertices in common. The authors suggested that the graph contains no clique of a size greater than 32. Figure 4–1 shows the number of detected cliques of various sizes. Finally, large quasi-cliques with density parameters

$\gamma = 0.9, 0.8, 0.7$, and 0.5 for the giant connected component were computed. The sizes of the largest quasi-cliques found were 44, 57, 65, and 98, respectively.

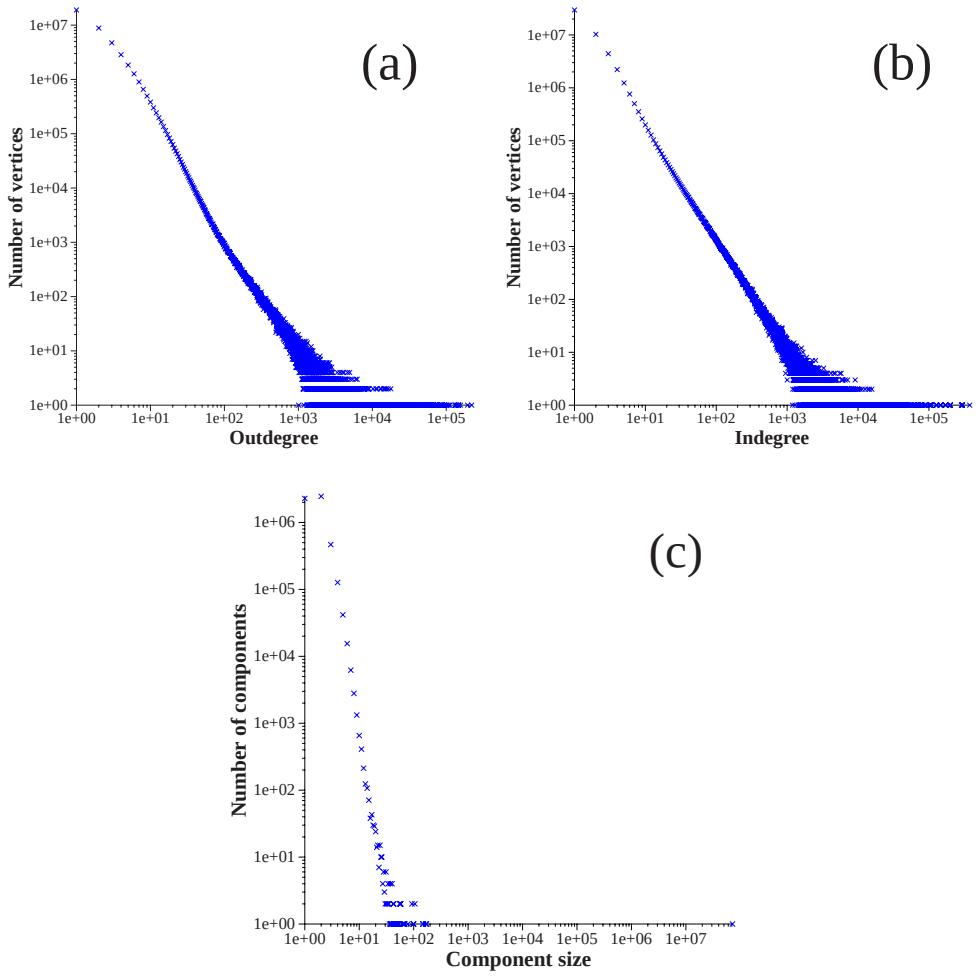

Figure 4–2: Number of vertices with various out-degrees (a) and in-degrees (b); the number of connected components of various sizes (c) in the call graph, due to Aiello et al. [11].

Aiello et al. [11] used the same data as Abello et al. [4] to show that the considered call graph fits to their random graph model which will be discussed in Subsection 4.1.3. The plots in Figure 4–2 demonstrate some connectivity properties of the call graph.

Figure 4–2(a,b) shows plots of the number of vertices for every out-degree and in-degree, respectively. Frequencies of the sizes of connected components are represented in Figure 4–2(c).

The Internet and Web graphs

The role of the Internet in the modern world is difficult to overestimate; its invention changed the way people interact, learn, and communicate like nothing before. Alongside with increasing significance, the Internet itself continues to grow at an overwhelming rate. Figure 4–3 shows the dynamics of growth of the number of Internet hosts for the last 13 years. As of January 2002 this number was estimated to be close to 150 million. The number of web pages indexed by large search engines exceeds 2 billion, and the number of web sites is growing by thousands daily.

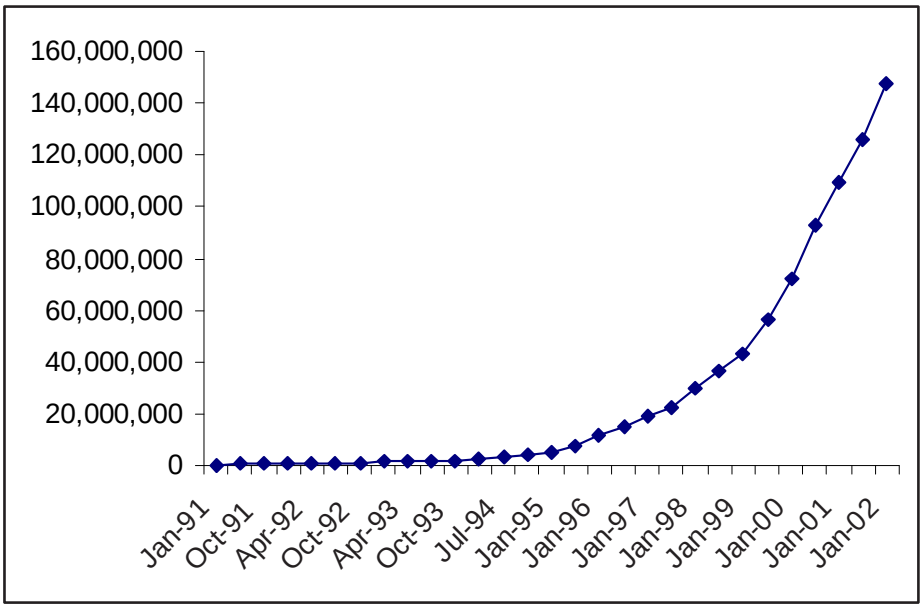

Figure 4–3: Number of Internet hosts for the period 01/1991-01/2002. Data by Internet Software Consortium [126].

The highly dynamic and seemingly unpredictable structure of the World Wide Web attracts more and more attention of scientists representing many diverse disciplines, including graph theory. In a graph representation of the World Wide Web, the vertices are documents and the edges are hyperlinks pointing from one document to another. Similarly to the call graph, the Web is a directed multigraph, although often it is treated as an undirected graph to simplify the analysis. Another graph is associated with the physical network of the Internet, where the vertices are

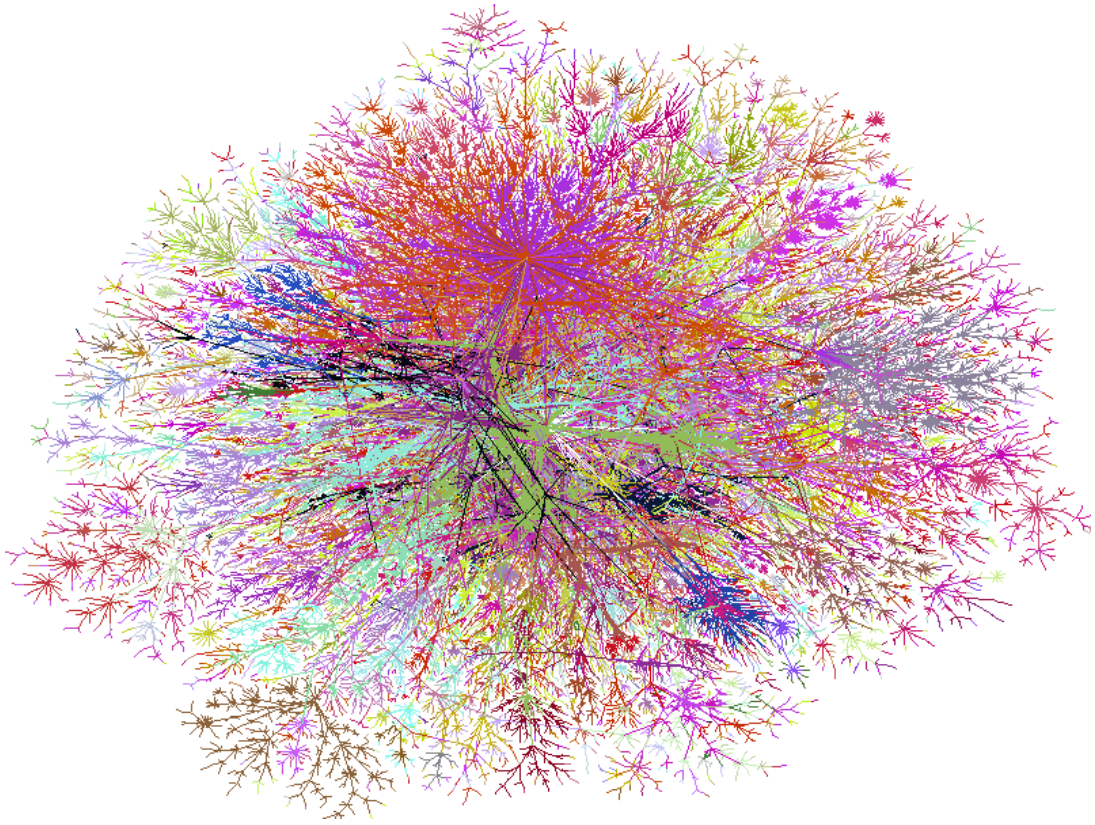

Figure 4–4: A sample of paths of the physical network of Internet cables created by W. Cheswick and H. Burch [60]. Courtesy of Lumeta Corporation. Patent(s) Pending. Copyright ©Lumeta Corporation 2003. All Rights Reserved.

routers navigating packets of data or groups of routers (*domains*). The edges in this graph represent wires or cables in the physical network. Figure 4–4 illustrates the tremendous complexity of this network. This and many other maps aiming to visualize the Internet topology are products of the "Internet Mapping Project" [60]. They are created from data obtained by tracing routes from one terminal to a set of other Internet domains.

Graph theory has been applied for web search [42, 59, 141], web mining [160, 161] and other problems arising in the Internet and World Wide Web. In several recent studies, there were attempts to understand some structural properties of the Web graph by investigating large Web crawls. Adamic and Huberman [8, 124] used

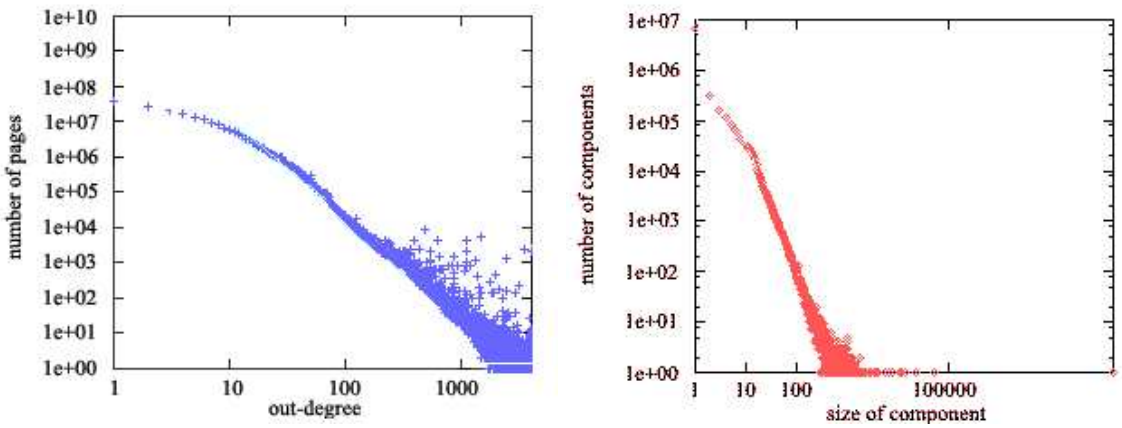

Figure 4–5: Number of vertices with various out-degrees (left) and distribution of sizes of strongly connected components (right) in Web graph [43].

crawls which covered almost 260,000 pages in their studies. Barabási and Albert [26] analyzed a subgraph of the Web graph approximately 325,000 nodes representing nd.edu pages. In another experiment, Kumar et al. [146] examined a data set containing about 40 million pages. In a recent study, Broder et al. [43] used two Altavista crawls, each with about 200 million pages and 1.5 billion links, thus significantly exceeding the scale of the preceding experiments. This work yielded several remarkable observations about local and global properties of the Web graph. All of the properties observed in one of the two crawls were validated for the other as well. Below, by the Web graph we will mean one of the crawls, which has 203,549,046 nodes and 2130 million arcs.

The first observation made by Broder et al. confirms a property of the Web graph suggested in earlier works [26, 146] claiming that the distribution of degrees follows a *power law*. That is, the number of vertices of degree k is proportional to $k^{-\gamma}$ for some $\gamma > 1$. Interestingly, the degree distribution of the Web graph resembles the power-law relationship of the Internet graph topology, which was first discovered by Faloutsos et al. [82]. Broder et al. [43] computed the in- and out-degree distributions for both considered crawls and showed that these distributions agree with power laws. Moreover, they observed that in the case of in-degrees the constant $\gamma \approx 2.1$ is the

same as the exponent of power laws discovered in earlier studies [26, 146]. In another set of experiments conducted by Broder et al., directed and undirected connected components were investigated. It was noticed that the distribution of sizes of these connected components also obeys a power law. Figure 4–5 illustrates the experiments with distributions of out-degrees and connected component sizes.

The last series of experiments discussed by Broder et al. [43] aimed to explore the global connectivity structure of the Web. This led to the discovery of the so-called *Bow-Tie* model of the Web [44]. Similarly to the call graph, the considered Web graph appeared to have a giant connected component, containing 186,771,290 nodes, or over 90% of the total number of nodes. Taking into account the directed nature of the edges, this connected component can be subdivided into four pieces: *strongly connected component* (*SCC*), *In* and *Out* components, and *"Tendrils"*. Overall, the Web graph in the Bow-Tie model is divided into the following pieces:

- The *strongly connected component* is the part of the giant connected component in which all nodes are reachable from one another by a directed path.
- The *In component* consists of nodes which can reach any node in the SCC but cannot be reached from the SCC.
- The *Out component* contains the nodes that are reachable from the SCC, but cannot access the SCC through directed links.
- The *Tendrils component* accumulates the remaining nodes of the giant connected component, *i.e.*, the nodes which are not connected with the SCC.
- The *Disconnected component* is the part of the Web which is not connected with the giant connected component.

Figure 4–6 shows the connectivity structure of the Web, as well as sizes of the considered components. As one can see from the figure, the sizes of SCC, In, Out and Tendrils components are roughly equal, and the Disconnected component is significantly smaller.

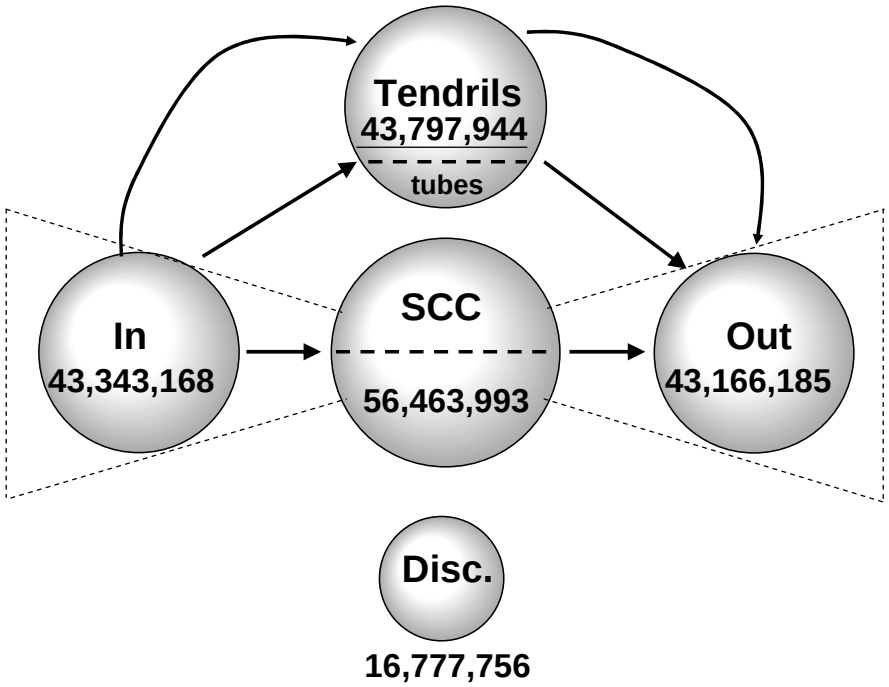

Figure 4–6: Connectivity of the Web due to Broder et al. [43].

Broder et al. [43] have also computed the diameters of the SCC and of the whole graph. It was shown that the diameter of the SCC is at least 28, and the diameter of the whole graph is at least 503. The *average connected distance* is defined as the pairwise distance averaged over those directed pairs (i, j) of nodes for which there exists a path from i to j. The average connected distance of the whole graph was estimated as 16.12 for in-links, 16.18 for out-links, and 6.83 for undirected links. Interestingly, it was also found that for a randomly chosen directed pair of nodes, the chance that there is a directed path between them is only about 24%. These results are not in agreement with the prediction of Albert et al. [12], who suggested that the average distance between two randomly chosen documents of the Web is 18.59. Let us mention that the property of a large network to have a small diameter has been observed in many real-life networks and is frequently referred to as the *small world phenomenon* [15, 198, 199].

4.1.2 External Memory Algorithms

In many cases, the data associated with massive graphs is too large to fit entirely inside the fast computer's internal memory, therefore a slower external memory (for example disks) needs to be used. The input/output communication (I/O) between these memories can result in an algorithm's slow performance. *External memory* (EM) algorithms and data structures are designed with aim to reduce the I/O cost by exploiting the locality. Recently, external memory algorithms have been successfully applied for solving batched problems involving graphs, including connected components, topological sorting, and shortest paths.

The first EM graph algorithm was developed by Ullman and Yannakakis [194] in 1991 and dealt with the problem of transitive closure. Many other researchers contributed to the progress in this area ever since [2, 18, 19, 49, 61, 147, 197]. Chiang et al. [61] proposed several new techniques for design and analysis of efficient EM graph algorithms and discussed applications of these techniques to specific problems, including minimum spanning tree verification, connected and biconnected components, graph drawing, and visibility representation. Abello et al. [2] proposed a functional approach for EM graph algorithms and used their methodology to develop deterministic and randomized algorithms for computing connected components, maximal independent sets, maximal matchings, and other structures in the graph. In this approach each algorithm is defined as a sequence of functions, and the computation continues in a series of scan operations over the data. If the produced output data, once written, cannot be changed, then the function is said to have no side effects. The lack of side effects enables the application of standard checkpointing techniques, thus increasing the reliability. Abello et al. presented a *semi-external* model for graph problems, which assumes that only the vertices fit in the computer's internal memory. This is quite common in practice, and in fact this was the case for the call graph described in Subsection 4.1.1, for which efficient EM

algorithms developed by Abello et al. [2] were used in order to compute its connected components [4].

For more detail on external memory algorithms see the book [6] and the extensive review by Vitter [197] of EM algorithms and data structures.

4.1.3 Modeling Massive Graphs

The size of real-life massive graphs, many of which cannot be held even by a computer with several gigabytes of main memory, vanishes the power of classical algorithms and makes one look for novel approaches. External memory algorithms and data structures discussed in the previous subsection represent one of the research directions aiming to overcome difficulties created by data sizes. But in some cases not only is the amount of data huge, but the data itself is not completely available. For instance, one can hardly expect to collect complete information about the Web graph; in fact, the largest search engines are estimated to cover only 38% of the Web [148].

Some approaches were developed for studying the properties of real-life massive graphs using only the information about a small part of the graph. For instance, Goldreich [105] proposes two randomized algorithms for testing if a given massive graph has some predefined property. These algorithms analyze a part of the graph and with some probability give the answer if this graph has a given property or not, based on a certain criterion.

Another methodology of investigating real-life massive graphs is to use the available information in order to construct proper theoretical models of these graphs. One of the earliest attempts to model real networks theoretically goes back to the late 1950's, when the foundations of random graph theory had been developed. In this subsection we will present some of the results produced by this and other (more realistic) graph models.

Uniform random graphs

The classical theory of random graphs founded by Erdös and Rényi [78, 79] deals with several standard models of the so-called *uniform* random graphs. Two of such models are $G(n,m)$ and $\mathcal{G}(n,p)$ [35]. The first model assigns the same probability to all graphs with n vertices and m edges, while in the second model each pair of vertices is chosen to be linked by an edge randomly and independently with probability p.

In most cases for each natural n a probability space consisting of graphs with exactly n vertices is considered, and the properties of this space as $n \to \infty$ are studied. It is said that a typical element of the space or *almost every* (*a.e.*) graph has property Q when the probability that a random graph on n vertices has this property tends to 1 as $n \to \infty$. We will also say that the property Q holds *asymptotically almost surely (a.a.s.)*. Erdös and Rényi discovered that in many cases either almost every graph has property Q or almost every graph does not have this property.

Many properties of uniform random graphs have been well studied [34, 35, 129, 143]. Below we will summarize some known results in this field.

Probably the simplest property to be considered in any graph is its *connectivity*. It was shown that for a uniform random graph $G(n,p) \in \mathcal{G}(n,p)$ there is a "threshold" value of p that determines whether a graph is almost surely connected or not. More specifically, a graph $G(n,p)$ is *a.a.s.* disconnected if $p < \frac{\log n}{n}$. Furthermore, it turns out that if p is in the range $\frac{1}{n} < p < \frac{\log n}{n}$, the graph $G(n,p)$ *a.a.s.* has a unique *giant connected component* [35]. The emergence of a giant connected component in a random graph is very often referred to as the "phase transition".

The next subject of our discussion is the diameter of a uniform random graph $G(n,p)$. Recall that the diameter of a disconnected graph is defined as the maximum diameter of its connected components. When dealing with random graphs, one usually speaks not about a certain diameter, but rather about the distribution of the possible values of the diameter. Intuitively, one can say that this distribution

depends on the interrelationship of the parameters of the model n and p. However, this dependency turns out to be rather complicated. It was discussed in many papers, and the corresponding results are summarized below.

It was proved by Klee and Larman [140] that a random graph asymptotically almost surely has the diameter d, where d is a certain integer value, if the following conditions are satisfied

$$\frac{p^{d-1}}{n} \to 0 \text{ and } \frac{p^d}{n} \to \infty, n \to \infty.$$

Bollobás [35] proved that if $np - \log n \to \infty$ then the diameter of a random graph is *a.a.s.* concentrated on no more than four values.

Łuczak [151] considered the case $np < 1$, when a uniform random graph *a.a.s.* is disconnected and has no giant connected component. Let $diam_T(G)$ denote the maximum diameter of all connected components of $G(n,p)$ which are trees. Then if $(1-np)n^{1/3} \to \infty$ the diameter of $G(n,p)$ is *a.a.s.* equal to $diam_T(G)$.

Chung and Lu [62] investigated another extreme case: $np \to \infty$. They showed that in this case the diameter of a random graph $G(n,p)$ is *a.a.s.* equal to

$$(1+o(1))\frac{\log n}{\log(np)}.$$

Moreover, they considered the case when $np \geq c > 1$ for some constant c and got a generalization of the above result:

$$(1+o(1))\frac{\log n}{\log(np)} \leq diam(G(n,p)) \leq \frac{\log n}{\log(np)} + 2\frac{\left(\frac{10c}{(\sqrt{c}-1)^2}+1\right)}{c-\log(2c)}\frac{\log n}{np} + 1.$$

Also, they explored the distribution of the diameter of a random graph with respect to different ranges of the ratio $np/\log n$. They obtained the following results:

- For $np/\log n = c > 8$ the diameter of $G(n,p)$ is *a.a.s.* concentrated on at most two values at $\log n/\log(np)$.

- For $8 \geq np/\log n = c > 2$ the diameter of $G(n,p)$ is *a.a.s.* concentrated on at most three values at $\log n/\log(np)$.

- For $2 \geq np/\log n = c > 1$ the diameter of $G(n,p)$ is *a.a.s.* concentrated on at most four values at $\log n/\log(np)$.

- For $1 \geq np/\log n = c > c_0$ the diameter of $G(n,p)$ is *a.a.s.* concentrated on a finite number of values, and this number is at most $2\left\lfloor \frac{1}{c_0} \right\rfloor + 4$. More specifically, in this case the following formula can be proved:

$$\left\lceil \frac{\log(c_0 n/11)}{\log(np)} \right\rceil \leq diam(G(n,p)) \leq \left\lceil \frac{\log\left(\frac{33c_0^2}{400} n \log n\right)}{\log(np)} \right\rceil + 2\left\lfloor \frac{1}{c_0} \right\rfloor + 2.$$

As pointed out above, a graph $G(n,p)$ *a.a.s.* has a giant connected component for $1 < np < \log n$. It is natural to assume that in this case the diameter of $G(n,p)$ is equal to the diameter of this giant connected component. However, it was strictly proved by Chung and Lu [62] that it is *a.a.s.* true only if $np > 3.5128$.

Potential drawbacks of the uniform random graph model

There were some attempts to model the real-life massive graphs by the uniform random graphs and to compare their behavior. However, the results of these experiments demonstrated a significant discrepancy between the properties of real graphs and corresponding uniform random graphs.

The further discussion analyzes the potential drawbacks of applying the uniform random graph model to the real-life massive graphs.

Though the uniform random graphs demonstrate some properties similar to the real-life massive graphs, many problems arise when one tries to describe the real graphs using the uniform random graph model. As it was mentioned above, a giant connected component *a.a.s.* emerges in a uniform random graph at a certain threshold. It looks very similar to the properties of the real massive graphs discussed in Subsection 4.1.3. However, after deeper insight, it can be seen that the giant

connected components in the uniform random graphs and the real-life massive graphs have different structures. The fundamental difference between them is as follows: it was noticed that in almost all the real massive graphs the property of so-called *clustering* takes place [198, 199]. It means that the probability of the event that two given vertices are connected by an edge is higher if these vertices have a common neighbor (*i.e.*, a vertex which is connected by an edge with both of these vertices). The probability that two neighbors of a given vertex are connected by an edge is called the *clustering coefficient*. It can be easily seen that in the case of the uniform random graphs, the clustering coefficient is equal to the parameter p, since the probability that each pair of vertices is connected by an edge is independent of all other vertices. In real-life massive graphs, the value of the clustering coefficient turns out to be much higher than the value of the parameter p of the uniform random graphs with the same number of vertices and edges. Adamic [7] found that the value of the clustering coefficient for some part of the Web graph was approximately 0.1078, while the clustering coefficient for the corresponding uniform random graph was 0.00023. Pastor-Satorras et al. [172] got similar results for the part of the Internet graph. The values of the clustering coefficients for the real graph and the corresponding uniform random graph were 0.24 and 0.0006 respectively.

Another significant problem arising in modeling massive graphs using the uniform random graph model is the difference in degree distributions. It can be shown that as the number of vertices in a uniform random graph increases, the distribution of the degrees of the vertices tends to the well-known *Poisson distribution* with the parameter np which represents the average degree of a vertex. However, as it was pointed out in Subsection 4.1.3, the experiments show that in the real massive graphs degree distributions obey a power law. These facts demonstrate that some other models are needed to better describe the properties of real massive graphs. Next,

we discuss two of such models; namely, the random graph model with a given degree sequence and its most important special case - the power-law model.

Random graphs with a given degree sequence

Besides the uniform random graphs, there are more general ways of modeling massive graphs. These models deal with *random graphs with a given degree sequence*. The main idea of how to construct these graphs is as follows. For all the vertices $i = 1 \dots n$ the set of the degrees $\{k_i\}$ is specified. This set is chosen so that the fraction of vertices that have degree k tends to the desired degree distribution p_k as n increases.

It turns out that some properties of the uniform random graphs can be generalized for the model of a random graph with a given degree sequence.

Recall the notation of so-called "phase transition" (*i.e.*, the phenomenon when at a certain point a giant connected component emerges in a random graph) which happens in the uniform random graphs. It turns out that a similar thing takes place in the case of a random graph with a given degree sequence. This result was obtained by Molloy and Reed [162]. The essence of their findings is as follows.

Consider a sequence of non-negative real numbers $p_0, p_1, \dots,$ such that $\sum_k p_k = 1$. Assume that a graph G with n vertices has approximately $p_k n$ vertices of degree k. If we define $Q = \sum_{k \geq 1} k(k-2)p_k$ then it can be proved that G *a.a.s.* has a giant connected component if $Q > 0$ and there is *a.a.s.* no giant connected component if $Q < 0$.

As a development of the analysis of random graphs with a given degree sequence, the work of Cooper and Frieze [66] should be mentioned. They considered a sparse directed random graph with a given degree sequence and analyzed its *strong* connectivity. In the study, the size of the giant strongly connected component, as well as the conditions of its existence, were discussed.

The results obtained for the model of random graphs with a given degree sequence are especially useful because they can be implemented for some important special cases of this model. For instance, the classical results on the size of a connected component in uniform random graphs follow from the aforementioned fact presented by Molloy and Reed. Next, we present another example of applying this general result to one of the most practically used random graph models - the power-law model.

Power-law random graphs

One of the most important special cases of the model of random graphs with a given degree sequence is the *power-law* random graph model. The power-law random graphs are also sometimes referred to as (α, β)-graphs. This model was recently applied to describe some real-life massive graphs such as the call graph, the Internet graph and the Web graph mentioned above. Some fundamental results for this model were obtained by Aiello, Chung and Lu [10, 11].

The basic idea of the power-law random graph model *P(α, β)* is as follows. If we define y to be the number of nodes with degree x, then according to this model

$$y = e^{\alpha}/x^{\beta}. \tag{4.1}$$

Equivalently, we can write

$$\log y = \alpha - \beta \log x. \tag{4.2}$$

This representation is more convenient in the sense that the relationship between y and x can be plotted as a straight line on a log-log scale, so that *(-β)* is the slope, and α is the intercept. This implies the following properties of a graph described by the power law model [11]:

- The maximum degree of the graph is $e^{\alpha/\beta}$.

- The number of vertices is

$$n = \sum_{x=1}^{e^{\frac{\alpha}{\beta}}} \frac{e^{\alpha}}{x^{\beta}} \approx \begin{cases} \zeta(\beta)e^{\alpha}, \beta > 1, \\ \alpha e^{\alpha}, \beta = 1, \\ e^{\alpha/\beta}/(1-\beta), 0 < \beta < 1, \end{cases} \tag{4.3}$$

  where $\zeta(t) = \sum_{n=1}^{\infty} \frac{1}{n^t}$ is the Riemann Zeta function.

- The number of edges is

$$E = \frac{1}{2}\sum_{x=1}^{e^{\frac{\alpha}{\beta}}} x\frac{e^{\alpha}}{x^{\beta}} \approx \begin{cases} \frac{1}{2}\zeta(\beta-1)e^{\alpha}, \beta > 2, \\ \frac{1}{4}\alpha e^{\alpha}, \beta = 2, \\ \frac{1}{2}e^{2\alpha/\beta}\Big/(2-\beta), 0 < \beta < 2. \end{cases} \tag{4.4}$$

Since the power-law random graph model is a special case of the model of a random graph with a given degree sequence, the results discussed above can be applied to the power law graphs. We need to find the threshold value of β in which the "phase transition" (*i.e.*, the emergence of a giant connected component) occurs. In this case $Q = \sum_{x\geq 1} x(x-2)p_x$ is defined as

$$Q = \sum_{x=1}^{e^{\frac{\alpha}{\beta}}} x(x-2)\left\lfloor \frac{e^{\alpha}}{x^{\beta}} \right\rfloor \approx \sum_{x=1}^{e^{\frac{\alpha}{\beta}}} \frac{e^{\alpha}}{x^{\beta-2}} - 2\sum_{x=1}^{e^{\frac{\alpha}{\beta}}} \frac{e^{\alpha}}{x^{\beta-1}} \approx [\zeta(\beta-2) - 2\zeta(\beta-1)]e^{\alpha} \text{ for } \beta > 3.$$

Hence, the threshold value β_0 can be found from the equation

$$\zeta(\beta-2) - 2\zeta(\beta-1) = 0,$$

which yields $\beta_0 \simeq 3.47875$.

The results on the size of the connected component of a power-law graph were presented by Aiello, Chung and Lu [11]. These results are summarized below.

- If $0 < \beta < 1$, then a power-law graph is *a.a.s.* connected (*i.e.*, there is only one connected component of size n).

- If $1 \leq \beta < 2$, then a power-law graph *a.a.s.* has a giant connected component (the component size is $\Theta(n)$), and the second largest connected component

a.a.s. has a size $\Theta(1)$.

- If $2 < \beta < \beta_0 = 3.47875$, then a giant connected component *a.a.s.* exists, and the size of the second largest component *a.a.s.* is $\Theta(\log n)$.
- $\beta = 2$ is a special case when there is *a.a.s.* a giant connected component, and the size of the second largest connected component is $\Theta(\log n / \log\log n)$.
- If $\beta > \beta_0 = 3.47875$, then there is *a.a.s.* no giant connected component.

The power-law random graph model was developed for describing real-life massive graphs. So the natural question is how well it reflects the properties of these graphs.

Though this model certainly does not reflect all the properties of real massive graphs, it turns out that the massive graphs such as the call graph or the Internet graph can be fairly well described by the power-law model. The following example demonstrates it.

Aiello, Chung and Lu [11] investigated the same call graph that was analyzed by Abello et al. [4]. This massive graph was already discussed in Subsection 4.1.3, so it is interesting to compare the experimental results presented by Abello et al. [4] with the theoretical results obtained in [11] using the power-law random graph model.

Figure 4–2 shows the number of vertices in the call graph with certain in-degrees and out-degrees. Recall that according to the power-law model the dependency between the number of vertices and the corresponding degrees can be plotted as a straight line on a log-log scale, so one can approximate the real data shown in Figure 4–2 by a straight line and evaluate the parameter α and β using the values of the intercept and the slope of the line. The value of β for the in-degree data was estimated to be approximately 2.1, and the value of e^{α} was approximately 30×10^6. The total number of nodes can be estimated using formula (4.3) as $\zeta(2.1) \times e^{\alpha} = 1.56 \times e^{\alpha} \approx 47 \times 10^6$ (compare with Subsection 4.1.3).

According to the results for the size of the largest connected component presented above, a power-law graph with $1 \leq \beta < 3.47875$ *a.a.s.* has a giant connected

component. Since $\beta \approx 2.1$ falls in this range, this result exactly coincides with the real observations for the call graph (see Subsection 4.1.3).

Another aspect that is worth mentioning is how to generate power-law graphs. The methodology for doing it was discussed in detail in the literature [10, 48, 65]. These papers use a similar approach, which is referred to as a *random graph evolution process*. The main idea is to construct a power-law massive graph "step-by-step": at each time step, a node and an edge are added to a graph in accordance with certain rules in order to obtain a graph with a specified in-degree and out-degree power-law distribution. The in-degree and out-degree parameters of the resulting power-law graph are functions of the input parameters of the model. A simple evolution model was presented by Kumar et al. [145]. Aiello, Chung and Lu [10] developed four more advanced models for generating both directed and undirected power-law graphs with different distributions of in-degrees and out-degrees. As an example, we will briefly describe one of their models. It was the basic model developed in the paper, and the other three models actually were improvements and generalizations of this model.

The main idea of the considered model is as follows. At the first time moment a vertex is added to the graph, and it is assigned two parameters - the *in-weight* and the *out-weight*, both equal to 1. Then at each time step $t + 1$ a new vertex with in-weight 1 and out-weight 1 is added to the graph with probability $1 - \alpha$, and a new directed edge is added to the graph with probability α. The origin and destination vertices are chosen according to the current values of the in-weights and out-weights. More specifically, a vertex u is chosen as the origin of this edge with the probability proportional to its current out-weight which is defined as $w^{out}_{u,t} = 1 + \delta^{out}_{u,t}$ where $\delta^{out}_{u,t}$ is the out-degree of the vertex u at time t. Similarly, a vertex v is chosen as the destination with the probability proportional to its current in-weight $w^{in}_{v,t} = 1 + \delta^{in}_{v,t}$ where $\delta^{in}_{v,t}$ is the in-degree of v at time t. From the above description it can be seen that at time t the total in-weight and the total out-weight are both equal to t. So for

each particular pair of vertices u and v, the probability that an edge going from u to v is added to the graph at time t is equal to

$$\alpha \frac{(1+\delta_{u,t}^{out})(1+\delta_{v,t}^{in})}{t^2}.$$

In the above notations, the parameter α is the input parameter of the model. The output of this model is a power-law random graph with the parameter of the degree distribution being a function of the input parameter. In the case of the considered model, it was shown that it generates a power-law graph with the distribution of in-degrees and out-degrees having the parameter $1 + \frac{1}{\alpha}$.

The notion of the so-called *scale invariance* [26, 27] must also be mentioned. This concept arises from the following considerations. The evolution of massive graphs can be treated as the process of growing the graph at a time unit. Now, if we replace all the nodes that were added to the graph at the same unit of time by only one node, then we will get another graph of a smaller size. The bigger the time unit is, the smaller the new graph size will be. The evolution model is called *scale-free* (*scale-invariant*) if with high probability the new (scaled) graph has the same power-law distribution of in-degrees and out-degrees as the original graph, for any choice of the time unit length. It turns out that most of the random evolution models have this property. For instance, the models of Aiello et al. [10] were proved to be scale-invariant.

4.1.4 Optimization in Random Massive Graphs

Recent random graph models of real-life massive networks, some of which were mentioned in Subsection 4.1.3 increased interest in various properties of random graphs and methods used to discover these properties. Indeed, numerical characteristics of graphs, such as clique and chromatic numbers, could be used as one of the steps in validation of the proposed models. In this regard, the expected clique number of power-law random graphs is of special interest due to the results by Abello et al. [4] and Aiello et al. [10] mentioned in Subsections 4.1.1 and 4.1.3. If computed,

it could be used as one of the points in verifying the validity of the model for the call graph proposed by Aiello et al. [10].

In this subsection we present some well-known facts regarding the clique and chromatic numbers in uniform random graphs.

Clique number

The earliest results describing the properties of cliques in uniform random graphs are due to Matula [159], who noticed that for a fixed p almost all graphs $G \in \mathcal{G}(n,p)$ have about the same clique number, if n is sufficiently large. Bollobás and Erdös [37] further developed these remarkable results by proving some more specific facts about the clique number of a random graph. Let us discuss these results in more detail by presenting not only the facts but also some reasoning behind them. For more detail see books by Bollobás [34, 35] and Janson et al. [129].

Assume that $0 < p < 1$ is fixed. Then instead of the sequence of spaces $\{\mathcal{G}(n,p), n \geq 1\}$ one can work with the single probability space $\mathcal{G}(\mathbf{N},p)$ containing graphs on $\mathbf{N}$ with the edges chosen independently with probability p. In this way, $\mathcal{G}(n,p)$ becomes an image of $\mathcal{G}(\mathbf{N},p)$, and the term "almost every" is used in its usual measure-theory sense. For a graph $G \in \mathcal{G}(\mathbf{N},p)$ we denote by G_n the subgraph of G induced by the first n vertices $\{1,2,\ldots,n\}$. Then the sequence $\omega(G_n)$ appears to be almost completely determined for *a.e.* $G \in \mathcal{G}(\mathbf{N},p)$.

For a natural l, let us denote by $k_l(G_n)$ the number of cliques spanning l vertices of G_n. Then, obviously,

$$\omega(G_n) = \max\{l : k_l(G_n) > 0\}.$$

When l is small, the random variable $k_l(G_n)$ has a large expectation and a rather small variance. If l is increased, then for most values of n there exists some number l_0 for which the expectation of $k_{l_0}(G_n)$ is fairly large (> 1) and $k_{l_0+1}(G_n)$ is much smaller than 1. Therefore, if we find this value l_0 then $\omega(G_n) = l_0$ with a high probability.

The expectation of $k_l(G_n)$ can be calculated as

$$E(k_l(G_n)) = \binom{n}{l} p^{\binom{l}{2}}.$$

Denoting by $f(l) = E(k_l(G_n))$ and replacing $\binom{n}{l}$ by its Stirling approximation we obtain

$$f(l) \approx \frac{n^{n+1/2}}{\sqrt{2\pi}(n-l)^{n-l+1/2} l^{l+1/2}} p^{l(l-1)/2}.$$

Solving the equation $f(l) = 1$ we get the following approximation l_0 of the root:

$$\begin{aligned} l_0 &= 2\log_{1/p} n - 2\log_{1/p}\log_{1/p} n + 2\log_{1/p}(e/2) + 1 + o(1) \\ &= 2\log_{1/p} n + O(\log\log n). \end{aligned} \tag{4.5}$$

Using this observation and the second moment method, Bollobás and Erdös [37] proved that if $p = p(n)$ satisfies $n^{-\epsilon} < p \leq c$ for every ϵ and some $c < 1$, then there exists a function $cl : \mathbf{N} \to \mathbf{N}$ such that *a.a.s.*

$$cl(n) \leq \omega(G_n) \leq cl(n) + 1,$$

i.e., the clique number is asymptotically distributed on at most two values. The sequence $cl(n)$ appears to be close to $l_0(n)$ computed in (4.5). Namely, it can be shown that for *a.e.* $G \in \mathcal{G}(\mathbf{N}, p)$ if n is large enough then

$$\lfloor l_0(n) - 2\log\log n/\log n \rfloor \leq \omega(G_n) \leq \lfloor l_0(n) + 2\log\log n/\log n \rfloor$$

and

$$\left|\omega(G_n) - 2\log_{1/p} n + 2\log_{1/p}\log_{1/p} n - 2\log_{1/p}(e/2) - 1\right| < \frac{3}{2}.$$

Frieze [91] and Janson et al. [129] extended these results by showing that for $\epsilon > 0$ there exists a constant c_ϵ, such that for $\frac{c_\epsilon}{n} \leq p(n) \leq \log^{-2} n$ *a.a.s.*

$$\lfloor 2\log_{1/p} n - 2\log_{1/p}\log_{1/p} n + 2\log_{1/p}(e/2) + 1 - \epsilon/p \rfloor \leq \omega(G_n) \leq$$

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

Chromatic number

Grimmett and McDiarmid [106] were the first to study the problem of coloring random graphs. Many other researchers contributed to solving this problem [13, 36]. We will mention some facts emerged from these studies.

Łuczak [150] improved the results about the concentration of $\chi(G(n,p))$ previously proved by Shamir and Spencer [182], proving that for every sequence $p = p(n)$ such that $p \leq n^{-6/7}$ there is a function $ch(n)$ such that $a.a.s.$

$$ch(n) \leq \chi(G(n,p)) \leq ch(n) + 1.$$

Alon and Krivelevich [13] proved that for any positive constant δ the chromatic number of a uniform random graph $G(n,p)$, where $p = n^{\frac{1}{2}-\delta}$, is $a.a.s.$ concentrated in two consecutive values. Moreover, they proved that a proper choice of $p(n)$ may result in a one-point distribution. The function $ch(n)$ is difficult to find, but in some cases it can be characterized. For example, Janson et al. [129] proved that there exists a constant c_0 such that for any $p = p(n)$ satisfying $\frac{c_0}{n} \leq p \leq \log^{-7} n$ $a.a.s.$

$$\frac{np}{2\log np - 2\log\log np + 1} \leq \chi(G(n,p)) \leq \frac{np}{2\log np - 40\log\log np}.$$

In the case when p is constant Bollobás' method utilizing martingales [35] yields the following estimate:

$$\chi(G(n,p)) = \frac{n}{2\log_b n - 2\log_b\log_b n + O_c(1)},$$

where $b = 1/(1-p)$.

4.1.5 Remarks

We discussed advances in several research directions dealing with massive graphs, such as external memory algorithms and modeling of massive networks as random

graphs with power-law degree distributions. Despite the evidence that uniform random graphs are hardly suitable for modeling the considered real-life graphs, the classical random graphs theory still may serve as a great source of ideas in studying properties of massive graphs and their models. We recalled some well-known results produced by the classical random graphs theory. These include results for concentration of clique number and chromatic number of random graphs, which would be interesting to extend to more complicated random graph models (*i.e.*, power-law graphs and graphs with arbitrary degree distributions). External memory algorithms and numerical optimization techniques could be applied to find an approximate value of the clique number (as it was discussed in Subsection 4.1.1). On the other hand, probabilistic methods similar to those discussed in Subsection 4.1.4 could be utilized in order to find the asymptotical distribution of the clique number in the same network's random graph model, and therefore verify this model.

4.2 The Market Graph

Although not so obviously as in the examples in Subsection 4.1.1, financial markets can also be represented as graphs. For a stock market one natural representation is based on the cross correlations of stock price fluctuations. A market graph can be constructed as follows: each stock is represented by a vertex, and two vertices are connected by an edge if the correlation coefficient of the corresponding pair of stocks (calculated for a certain period of time) is above a prespecified threshold $\theta, -1 \leq \theta \leq 1$.

4.2.1 Constructing the Market Graph

The market graph that we study in this chapter represents the set of financial instruments traded in the U.S. stock markets. More specifically, we consider 6546 instruments and analyze daily changes of their prices over a period of 500 consecutive trading days in 2000-2002. Based on this information, we calculate the cross-correlations between each pair of stocks using the following formula [156]:

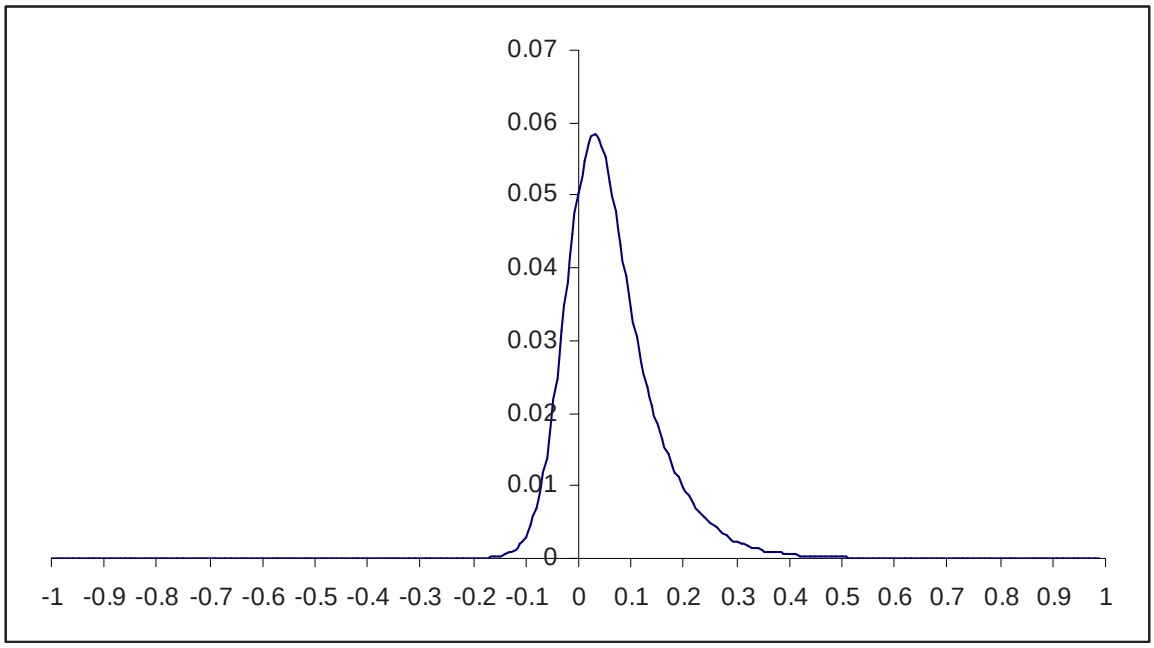

Figure 4–7: Distribution of correlation coefficients in the stock market

$$C_{ij} = \frac{\langle R_i R_j \rangle - \langle R_i \rangle \langle R_j \rangle}{\sqrt{\langle R_i^2 - \langle R_i \rangle^2 \rangle \langle R_j^2 - \langle R_j \rangle^2 \rangle}},$$

where $R_i(t) = \ln \frac{P_i(t)}{P_i(t-1)}$ defines the return of the stock i for day t. $P_i(t)$ denotes the price of the stock i on day t.

The correlation coefficients C_{ij} can vary from -1 to 1. Figure 4–7 shows the distribution of the correlation coefficients based on the prices data for the years 2000-2002. It can be seen that this distribution is nearly symmetric around the mean, which is approximately equal to 0.05.

The main idea of constructing a market graph is as follows. Let the set of financial instruments represent the set of vertices of the graph. Also, we specify a certain threshold value $\theta, -1 \leq \theta \leq 1$ and add an undirected edge connecting the vertices i and j if the corresponding correlation coefficient C_{ij} is greater than or equal to θ. Obviously, different values of θ define the market graphs with the same set of vertices, but different sets of edges.

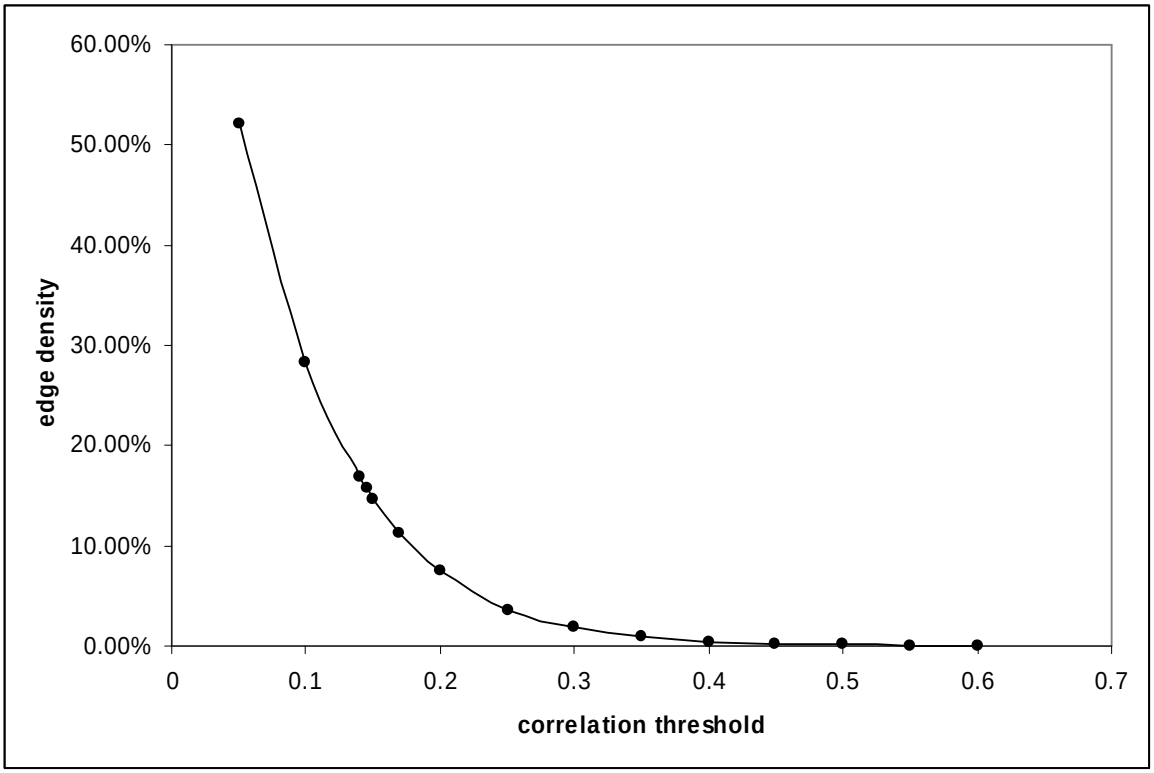

Figure 4–8: Edge density of the market graph for different values of the correlation threshold.

It is easy to see that the number of edges in the market graph decreases as the threshold value θ increases. In fact, our experiments show that the edge density of the market graph decreases exponentially w.r.t. θ. The corresponding graph is presented on Figure 4–8.

4.2.2 Connectivity of the Market Graph

In Subsection 4.1.3 we mentioned the connectivity thresholds in random graphs. The main idea of this concept is finding a threshold value of the parameter of the model (p in the case of uniform random graphs, and β in the case of power-law graphs) that will define if the graph is connected or not. Moreover, if the graph is disconnected, another threshold value can be defined to determine if the graph has a giant connected component or all of its connected components have a small size.

For instance, in the case of the power-law model $\beta = 1$ is a threshold value that determines the connectivity of the power-law graph, *i.e.*, the graph is a.a.s. connected

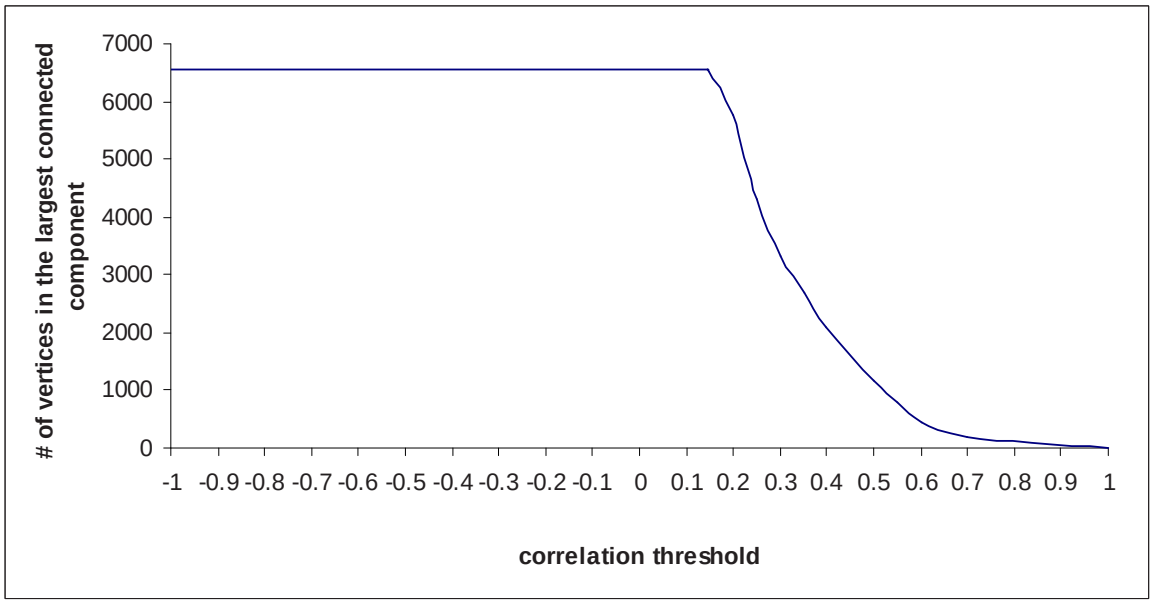

Figure 4–9: Plot of the size of the largest connected component in the market graph as a function of correlation threshold θ.

if $\beta < 1$, and it is a.a.s. disconnected otherwise. Similarly, $\beta \approx 3.47875$ defines the existence of a giant connected component in the power-law graph.

Now a natural question arises: what is the connectivity threshold for the market graph? Since the number of edges in the market graph depends on the chosen correlation threshold θ, we should find a value θ_0 that determines the connectivity of the graph. As it was mentioned above, the smaller value of θ we choose, the more edges the market graph will have. So, if we decrease θ, after a certain point, the graph will become connected. We have conducted a series of computational experiments for checking the connectivity of the market graph using the breadth-first search technique, and we obtained a relatively accurate approximation of the connectivity threshold: $\theta_0 \simeq 0.14382$. Moreover, we investigated the dependency of the size of the largest connected component in the market graph w.r.t. θ. The corresponding plot is shown on Figure 4–9.

4.2.3 Degree Distributions in the Market Graph

As it was shown in the previous section, the power-law model fairly well describes some of the real-life massive graphs, such as the Web graph and the call graph. In this subsection, we will show that the market graph also obeys the power-law model.

It should be noted that since we consider a set of market graphs, where each graph corresponds to a certain value of θ, the degree distributions will be different for each θ.

The results of our experiments turned out to be rather interesting.

If we specify a small value of the correlation threshold θ, such as $\theta = 0$, $\theta = 0.05$, $\theta = 0.1$, $\theta = 0.15$, the distribution of the degrees of the vertices is very "noisy" and does not have any well-defined structure. Note that for these values of θ the market graph is connected and has a high edge density. The market graph structure seems to be very difficult to analyze in these cases.

However, the situation changes drastically if a higher correlation threshold is chosen. As the edge density of the graph decreases, the degree distribution more and more resembles a power law. In fact, for $\theta \geq 0.2$ this distribution is approximately a straight line in the log-log scale, which is exactly the power law distribution, as it was shown in Section 4.1. Figure 4–10 demonstrates the degree distributions of the market graphs for some values of the correlation threshold.

An interesting observation is that the slope of the lines (which is equal to the parameter β of the power-law model) is rather small. It can be seen from formula (4.1) that in this case the graph will contain many vertices with a high degree. This fact is important for the next subject of our interest - finding maximum cliques in the market graph. Intuitively, one can expect a large clique in a graph with a small value of the parameter β. As we will see next, this assumption is true for the market graph.

Another combinatorial optimization problem associated with the market graph is finding maximum independent sets in the graphs with a negative correlation threshold

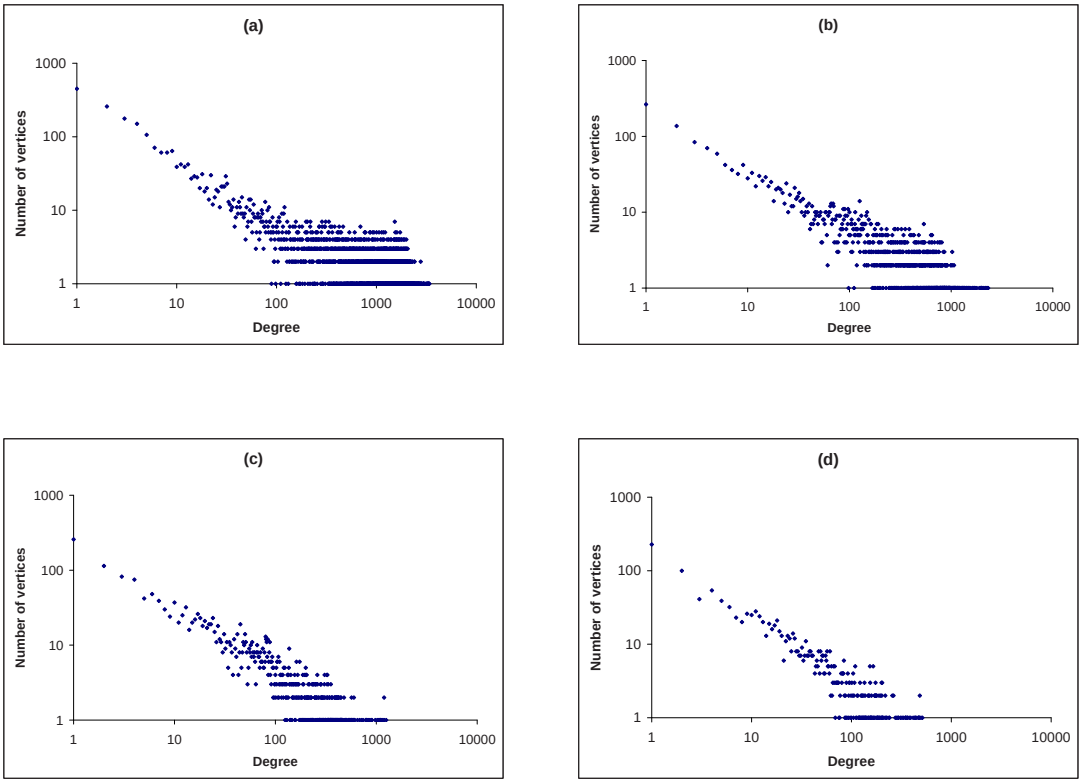

Figure 4–10: Degree distribution of the market graph for (a) $\theta = 0.2$; (b) $\theta = 0.3$; (c) $\theta = 0.4$; (d) $\theta = 0.5$

θ. Clearly, instruments in an independent set will be negatively correlated with each other, and therefore form a diversified portfolio.

However, we can consider a *complementary graph* for a market graph with a negative value of θ. In this graph, an edge will connect instruments i and j if the correlation between them $C_{ij} < \theta$. Recall that a maximum independent set in the initial graph is a maximum clique in the complementary graph, so the maximum independent set problem can be reduced to the maximum clique problem in the complementary graph.

Therefore, it is also useful to investigate the degree distributions of these complementary graphs. As it can be seen from Figure 4–7, the distribution of the correlation coefficients is almost symmetric around $\theta = 0.05$, so for the values of θ

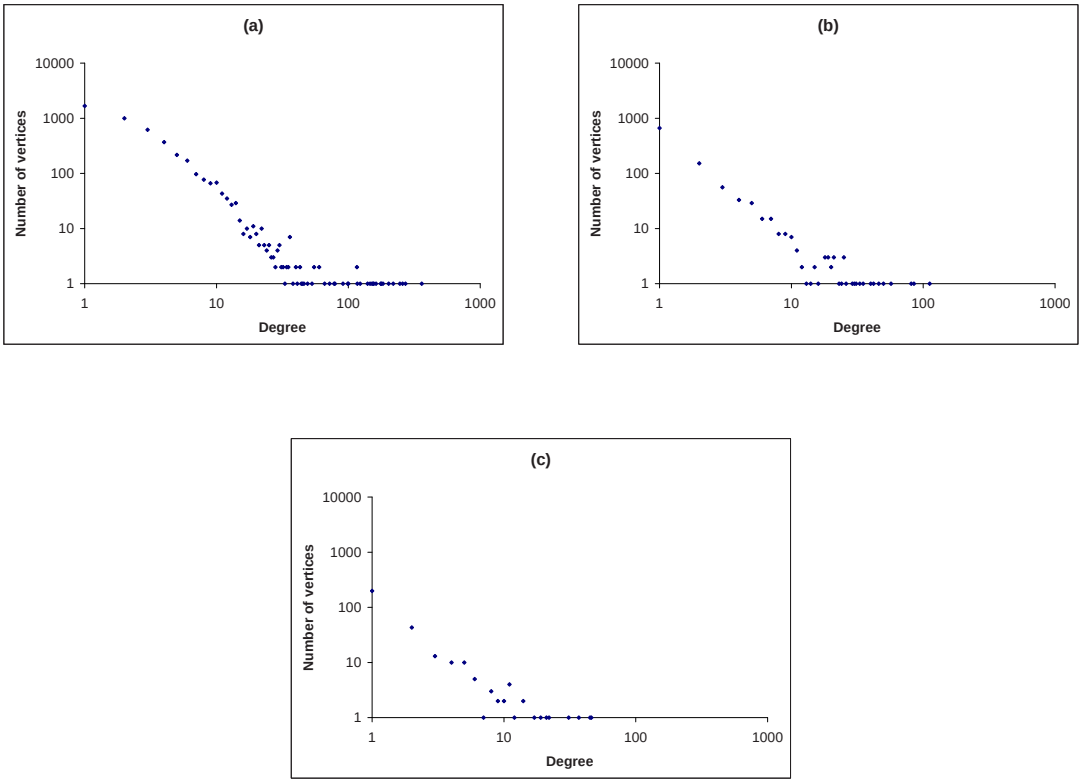

Figure 4–11: Degree distribution of the complementary market graph for (a) $\theta = -0.15$; (b) $\theta = -0.2$; (c) $\theta = -0.25$

close to 0 the edge density of both the initial and the complementary graph is high enough. So, for these values of θ the degree distribution of a complementary graph is also "noisy" as in the case of the corresponding initial graph.

As θ decreases (*i.e.*, increases in the absolute value), the degree distribution of a complementary graph tends to the power law. The corresponding graphs are shown on Figure 4–11. However, in this case, the slope of the line in the log-log scale (the value of the parameter β) is higher than in the case of positive values of θ. It means that there are not many vertices with a high degree in these graphs, so the size of a maximum clique should be significantly smaller than in the case of the market graphs with a positive correlation threshold.

Table 4–1: Clustering coefficients of the market graph (* - complementary graph)

θ	clustering coef.
-0.15*	2.64×10^{-5}
-0.1*	0.0012
0.3	0.4885
0.4	0.4458
0.5	0.4522
0.6	0.4872
0.7	0.4886

It is also interesting to compare the difference in clustering coefficients of a market graph (with positive values of θ) or its complement (with negative values of θ) (see Table 4–1).

Intuitively, the large clustering coefficients should correspond to graphs with larger cliques, therefore, from Table 4–1 one should expect that the cliques in market graph with positive θ are much larger than the independent sets in market graph with negative θ. This prediction will be confirmed in the next subsection, where we present the computational results of solving the maximum clique and maximum independent set problems in the market graph.

4.2.4 Cliques and Independent Sets in the Market Graph

As it was mentioned above, the maximum clique and the maximum independent set problems are NP-hard. It makes these problems especially challenging in large graphs. The maximum clique problem admits an integer programming formulation, however, in the case of the graph with 6546 vertices this integer programming problem cannot be solved in a reasonable time. Therefore, we used a greedy heuristic for finding a lower bound of the clique number, and a special preprocessing technique which reduces a problem size.

To find a large clique, we apply the following greedy algorithm. Starting with an empty set, we recursively add to the clique a vertex from the neighborhood of the clique adjacent to the most vertices in the neighborhood of the clique. If we denote by

$N(i) = \{j|(i,j) \in E\}$ the set of neighbors of i in $G = (V, E)$, then the neighborhood of a clique C is $\bigcap_{i \in C} N(i)$, and we obtain the following algorithm:

$C = \emptyset, G_0 = G$;

do

$G_0 = \bigcap_{i \in C} N(i) \setminus C$;

$C = C \bigcup j$, where j is a vertex of largest degree in G_0;

until $G_0 = \emptyset$.

After running this algorithm, we applied the following preprocessing procedure [4]. We recursively remove from the graph all of the vertices which are not in C and whose degree is less than $|C|$, where C is the clique found by the above algorithm. This simple procedure enabled us to significantly reduce the size of the maximum clique search space. Let us denote by $G'(V', E')$ the graph induced by remaining vertices. Table 4–2 presents the sizes of the cliques found using the greedy algorithm, and sizes of the graphs remaining after applying the preprocessing procedure.

Table 4–2: Sizes of cliques found using the greedy algorithm and sizes of graphs remaining after applying the preprocessing technique

θ	edge density	clique size	$\|V'\|$	edge dens. in G'
0.35	0.0090	168	535	0.6494
0.4	0.0047	104	405	0.6142
0.45	0.0024	109	213	0.8162
0.5	0.0013	84	146	0.8436
0.55	0.0007	61	102	0.8701
0.6	0.0004	45	70	0.8758
0.65	0.0002	23	80	0.5231
0.7	0.0001	21	33	0.7557

In order to find the maximum clique of G' (which is also the maximum clique in the original graph G), we used the following integer programming formulation of the maximum clique problem (see Section 2.1):

$$\text{maximize} \quad \sum x_i$$

Table 4–3: Sizes of the maximum cliques in the market graph with different values of the correlation threshold

θ	edge density	clique size
0.35	0.0090	193
0.4	0.0047	144
0.45	0.0024	109
0.5	0.0013	85
0.55	0.0007	63
0.6	0.0004	45
0.65	0.0002	27
0.7	0.0001	22

s.t.

$$x_i + x_j \leq 1, (i,j) \notin E';$$

$$x_i \in \{0,1\}, i = 1, \ldots, n.$$

We used CPLEX 7.0 [125] to solve this integer program for some of the considered instances.

Table 4–3 summarizes the sizes of the maximum cliques found in the graph for different values of θ. It turns out that these cliques are rather large. In fact, even for $\theta = 0.6$, which is a very high correlation threshold, the clique of size 45 was found.

These results are in agreement with the above discussion, where we analyzed the degree distributions of the market graphs with positive values of θ and came to the conclusion that the cliques in these graphs should be large.

The financial interpretation of the clique in the market graph is that it defines the set of stocks whose price fluctuations exhibit a similar behavior. Our results show that in the modern stock market there are large groups of instruments that are correlated with each other.

Next, we consider the maximum independent set problem in the market graphs with nonpositive values of the correlation threshold θ. This problem can be easily represented as a maximum clique problem in a complementary graph. Interestingly, the preprocessing procedure that was very helpful for finding maximum cliques in

Table 4–4: Sizes of independent sets found using the greedy algorithm

θ	edge density	indep. set size
0.05	0.4794	36
0.0	0.2001	12
-0.05	0.0431	5
-0.1	0.005	3
-0.15	0.0005	2

original graphs was absolutely useless in the case with their complements. Therefore, we conclude that the maximum independent set appears to be more difficult to compute than the maximum clique in the market graph. Table 4–4 presents the results obtained using the greedy algorithm described above.

As one can see, the sizes of the computed independent sets are very small, which coincides with the prediction that was made before based on the analysis of the degree distributions.

From the financial point of view, the independent set in the market graph represents "completely diversified" portfolio, where all instruments are negatively correlated with each other. It turns out that choosing such a portfolio is not an easy task, and one cannot expect to easily find a large group of negatively correlated instruments.

4.2.5 Instruments Corresponding to High-Degree Vertices

Up to this point, we studied the properties of the market graph as one big system, and did not consider the characteristics of every vertex in this graph. However, a very important practical issue is to investigate the degree of each vertex in the market graph and to find the vertices with high degrees, *i.e.*, the instruments that are highly correlated with many other instruments in the market. Clearly, this information will help us to answer a very important question: which instruments most accurately reflect the behavior of the market?

For this purpose, we chose the market graph with a high correlation threshold ($\theta = 0.6$), calculated the degrees of each vertex in this graph and sorted the vertices in the decreasing order of their degrees.

Interestingly, even though the edge density of the considered graph is only 0.04% (only highly correlated instruments are connected by an edge), there are many vertices with degrees greater than 100.

According to our calculations, the vertex with the highest degree in this market graph corresponds to the NASDAQ 100 Index Tracking Stock. The degree of this vertex is 216, which means that there are 216 instruments that are highly correlated with it. An interesting observation is that the degree of this vertex is twice higher than the number of companies whose stock prices the NASDAQ index reflects, which means that these 100 companies greatly influence the market.

In Table 4–5 we present the "top 25" instruments in the U.S. stock market, according to their degrees in the considered market graph. The corresponding symbols definitions can be found on several web sites, for example http://www.nasdaq.com. Note that most of them are indices that incorporate a number of different stocks of companies in different industries. Although this result is not surprising from the financial point of view, it is important as a practical justification of the market graph model.

4.3 Evolution of the Market Graph

In order to investigate the dynamics of the market graph structure, we chose the period of 1000 trading days in 1998 - 2002 and considered eleven 500-day shifts within this period. The starting points of every two consecutive shifts are separated by the interval of 50 days. Therefore, every pair of consecutive shifts had 450 days in common and 50 days different (see Figure 4–12). Dates corresponding to each shift are summarized in Table 4–6.

Table 4–5: Top 25 instruments with highest degrees ($\theta = 0.6$).

symbol	vertex degree
QQQ	216
IWF	193
IWO	193
IYW	193
XLK	181
IVV	175
MDY	171
SPY	162
IJH	159
IWV	158
IVW	156
IAH	155
IYY	154
IWB	153
IYV	150
BDH	144
MKH	143
IWM	142
IJR	134
SMH	130
STM	118
IIH	116
IVE	113
DIA	106
IWD	106

The advantage of this procedure is that it allows us to accurately reflect the structural changes of the market graph using relatively small intervals between shifts, but at the same time we can maintain large sample sizes of the stock prices data for calculating cross-correlations for each shift. We should note that in our analysis we considered only stocks which were among those traded as of the last of the 1000 trading days. For practical reasons we did not take into account stocks which had been withdrawn from the market. However, these could be included in a more detailed analysis to obtain a better global picture of the market evolution.

The first subject of our consideration is the distribution of correlation coefficients between all pairs of stocks in the market. Recall that for the market graph considered

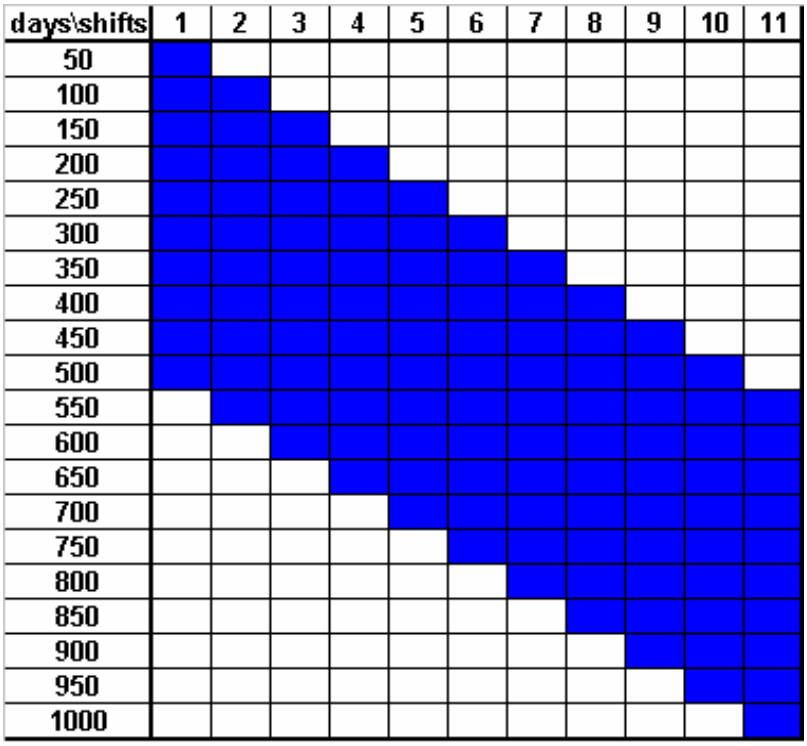

Figure 4–12: Time shifts used for studying the evolution of the market graph structure.

in the previous section this distribution was nearly symmetric around 0.05 and like a normal distribution it was concentrated around its mean. An interpretation of this fact is that the correlation of most pairs of stocks is close to zero, therefore, the structure of the stock market is substantially random, and one can make a reasonable assumption that the prices of most stocks change independently. As we consider the evolution of the correlation distribution over time, it turns out that this distribution remains almost unchanged for all time intervals, which is illustrated by Figure 4–13.

The stability of the correlation coefficients distribution of the market graph intuitively motivates the hypothesis that the degree distribution should also remain stable. To verify this assumption, we have calculated the degree distribution of the graphs constructed for all considered time shifts. The correlation threshold $\theta = 0.5$ was chosen. Our experiments show that the degree distribution is similar for all intervals, and in all cases it is well described by a power law. Figure 4–14 shows

Table 4–6: Dates corresponding to each 500-day shift.

Period #	Starting date	Ending date
1	09/24/1998	09/15/2000
2	12/04/1998	11/27/2000
3	02/18/1999	02/08/2001
4	04/30/1999	04/23/2001
5	07/13/1999	07/03/2001
6	09/22/1999	09/19/2001
7	12/02/1999	11/29/2001
8	02/14/2000	02/12/2002
9	04/26/2000	04/25/2002
10	07/07/2000	07/08/2002
11	09/18/2000	09/17/2002

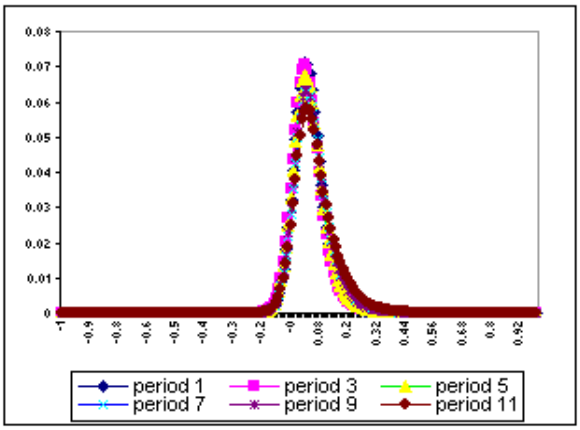

Figure 4–13: Distribution of the correlation coefficients between all considered pairs of stocks in the market, for odd-numbered time shifts.

the degree distributions (in the logarithmic scale) for some instances of the market graph corresponding to different intervals. As one can see, all these plots can be well approximated by straight lines, which means that they represent the power-law distribution.

The cross-correlation distribution and the degree distribution of the market graph represent the global characteristics of the market, and the aforementioned results lead us to the conclusion that the general structure of the market is stable over time. However, as we will see now, some global changes in the stock market structure do

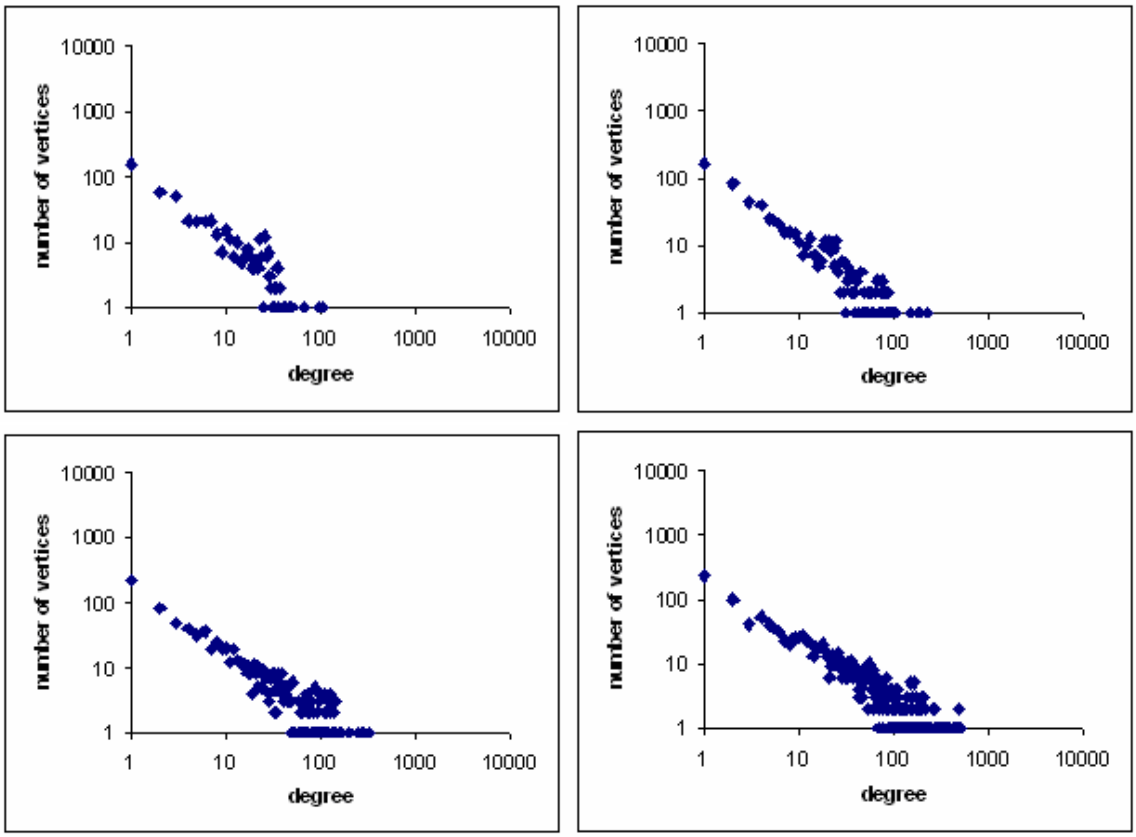

Figure 4–14: Degree distribution of the market graph for periods (from left to right, from top to bottom) 1, 4, 7, and 11 (logarithmic scale).

take place. In order to demonstrate it, we look at another characteristic of the market graph, its edge density.

For studying the edge density of the market graph, we chose a relatively high correlation threshold $\theta = 0.5$ that would ensure that we consider only the edges corresponding to the pairs of stocks, which are significantly correlated with each other. In this case, the edge density of the market graph would represent the proportion of those pairs of stocks in the market, whose price fluctuations are somewhat similar and correspondingly influence each other. The subject of our interest is to study how this proportion changes during the considered period of time. Table 4–7 summarizes the obtained results.

Table 4–7: Number of vertices and number of edges in the market graph ($\theta = 0.5$) for different periods.

Period	$\|V\|$	$\|E\|$	Edge density
1	5430	2258	0.015%
2	5507	2614	0.017%
3	5593	3772	0.024%
4	5666	5276	0.033%
5	5768	6841	0.041%
6	5866	7770	0.045%
7	6013	10428	0.058%
8	6104	12457	0.067%
9	6262	12911	0.066%
10	6399	19707	0.096%
11	6556	27885	0.130%

As it can be seen from this table, both the number of vertices and the number of edges in the market graph increase as time goes. Obviously, the number of vertices grows since new stocks appear in the market, and we don't consider those stocks which ceased to exist by the last of 1000 trading days used in our analysis, so the maximum possible number of edges in the graph increases as well. However, it turns out that the number of edges grows faster; therefore, the edge density of the market graph increases from period to period. As one can see from Figure 4–15, the greatest increase of the edge density corresponds to the last two periods. In fact, the edge density for the latest interval is approximately 8.5 times higher than for the first interval!

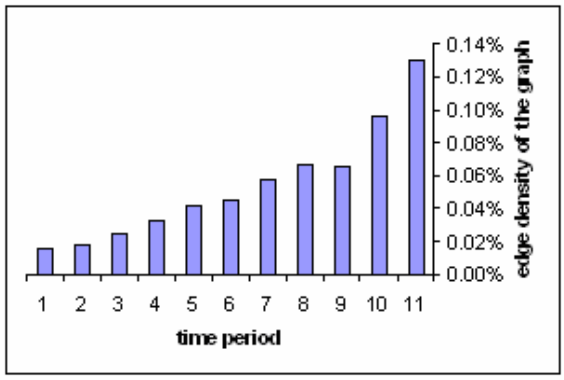

Figure 4–15: Growth dynamics of the edge density of the market graph over time.

This dramatic jump suggests that there is a trend to the "globalization" of the modern stock market, which means that nowadays more and more stocks significantly affect the behavior of the others, and the structure of the market becomes not purely random.

It should be noted that the increase of the edge density could be predicted from the analysis of the distribution of the cross-correlations between all pairs of stocks, which was mentioned above. From Figure 4–13, one can observe that even though the distributions corresponding to different periods have a similar shape and the same mean, the "tail" of the distribution corresponding to the latest period (period 11) is significantly "heavier" than for the earlier periods, which means that there are more pairs of stocks with higher values of the correlation coefficient.

Next, we find the stock with the highest degree in the market graph. Table 4–8 presents the stocks with the highest degrees in the market graph with the correlation threshold $\theta = 0.5$, for different time intervals. As one would expect, these stocks are exchange traded funds: NASDAQ-100 index tracking stock (QQQ), MidCap SPDR Trust Series I (MDY), and SPDR Trust Series I (SPY). Note, that the values of the highest degree in the market graph increase from period to period, which is another confirmation of the fact that the "globalization" of the stock market actually takes place.

4.4 Conclusion

A great number of problems arising in massive graphs exhibit a challenging and attractive field of research for representatives of a variety of diverse fields of science and engineering, including probability, graph theory, and computer science among many others.

In this chapter we have given a brief review of the recent progress in selected topics associated with massive graphs. Although in examples we restricted our attention to graphs arising in telecommunications, the Internet and finance only,

Table 4–8: Vertices with the highest degrees in the market graph for different periods ($\theta = 0.5$).

Period	Stock with the highest degree	Degree
1	SPY	106
2	QQQ	130
3	QQQ	181
4	QQQ	224
5	QQQ	252
6	QQQ	267
7	QQQ	318
8	QQQ	338
9	QQQ	337
10	MDY	402
11	MDY	514

there are many other real-life networks with similar problems which have attracted considerable research efforts in the past few years [117, 130, 131, 173].

We presented a detailed study of the properties of the market graph, including the analysis of its degree distribution, connectivity, as well as computing of cliques and independent sets. Finding cliques and independent sets in the market graph gives us a new tool of the analysis of the market structure by classifying the stocks into different groups.

Moreover, we proved that the power-law model, which well describes the massive graphs arising in telecommunications and the Internet, is also applicable in finance. It confirms an amazing observation that many real-life massive graphs have a similar power-law structure.

In Section 4.3, we have studied the evolution of different characteristics of the market graph over time and came to several interesting conclusions based on our analysis. It turns out that the power-law structure of the market graph is quite stable over the considered time intervals. Another interesting result is the fact that the edge density of the market graph and the highest degrees of the vertices steadily

increase during the last several years, which supports the well-known idea about the globalization of economy which has been widely discussed recently.

Although we addressed many issues in our analysis of the market graph, there are still a lot of open problems. For instance, since the independent sets in the market graph turned out to be very small, there is a possibility to consider quasi-cliques instead of cliques in the complementary graph. This will allow us to find larger diversified portfolios which is important from the practical point of view. Also, one can consider another type of the market graph based on the data of the liquidity of different instruments, instead of considering the returns. It would be very interesting to study the properties of this graph and compare it with the market graph considered in this chapter. Therefore, this research direction is very promising and important for a deeper understanding of market behavior.

CHAPTER 5
APPLICATIONS IN CODING THEORY

Error correcting codes lie in the heart of digital technology, making cell phones, compact disk players and modems possible. They are also of special significance due to increasing importance of reliability issues in internet transmissions. A fundamental problem of interest is to send a message across a noisy channel with a maximum possible reliability. This problem is related to the problem of finding the largest error correcting codes for certain types of errors. Computing estimates of the size of correcting codes is important from both theoretical and practical perspectives.

In this chapter we use the observation that the problem of finding the largest correcting codes and their estimates can be formulated in terms of maximum independent set and graph coloring problems in specially constructed graphs. We solve the problem of finding the largest correcting codes using efficient algorithms for optimization problems in graphs. We report new exact solutions and estimates. This chapter is based on publications by Butenko et al. [55, 56].

5.1 Introduction

Let a positive integer n be given. For a binary vector $u \in B^n$ denote by $F_e(u)$ the set of all vectors (not necessary of dimension n) which can be obtained from u as a consequence of certain error e, such as deletion or transposition of bits. A subset $C \subseteq B^n$ is said to be an e-correcting code if $F_e(u) \bigcap F_e(v) = \emptyset$ for all $u, v \in C,\ u \neq v$. In this paper we consider the following cases for the error e.

- Single deletion ($e = 1d$): $F_{1d}(u) \subseteq B^{n-1}$ and all elements of $F_{1d}(u)$ are obtained by deletion of one of the components of u. For example, if $n = 4$, $u = 0101$ then $F_{1d}(u) = \{101, 001, 011, 010\}$. Sloane [188] has published a survey of single-deletion-correcting codes.

- Two-deletion ($e = 2d$): $F_{2d}(u) \subseteq B^{n-2}$ and all elements of $F_{2d}(u)$ are obtained by deletion of two of the components of u. For $u = 0101$ we have $F_{2d}(u) = \{00, 01, 10, 11\}$.

- Single transposition, excluding the end-around transposition ($e = 1t$): $F_{1t}(u) \subseteq B^n$ and all elements of $F_{1t}(u)$ are obtained by transposition of a neighboring pair of components in u. For example, if $n = 5$, $u = 11100$ then $F_{1t}(u) = \{11100, 11010\}$.

- Single transposition, including the end-around transposition ($e = 1et$): $F_{1et}(u) \subseteq B^n$ and all elements of $F_{1et}(u)$ are obtained by transposition of a neighboring pair of components in u, where the first and the last components are also considered as neighbors. For $n = 5$, $u = 11100$ we obtain $F_{1et}(u) = \{11100, 11010, 01100\}$.

- One error on the Z-Channel ($e = 1z$): $F_{1z}(u) \subseteq B^n$ and all elements of $F_{1z}(u)$ are obtained by possibly changing one of the nonzero components of u from 1 to 0. If $n = 5$, $u = 11100$ then $F_{1z}(u) = \{11100, 01100, 10100, 11000\}$. The codes correcting one error on the Z-channel represent the simplest case of *asymmetric codes*.

The problem of our interest is to find the largest correcting codes. It appears that this problem can be formulated in terms of extremal graph problems as follows.

Consider a graph G_n having a vertex for every vector $u \in B^n$, with an edge joining the vertices corresponding to $u, v \in B^n,\ u \neq v$ if and only if $F_e(u) \bigcap F_e(v) \neq \emptyset$. Then a correcting code corresponds to an independent set in G_n. Hence, the largest e-correcting code can be found by solving the maximum independent set problem in the considered graph. Note, that this problem could be equivalently formulated as the maximum clique problem in the complement graph of G.

In this chapter, using efficient approaches to the maximum independent set and graph coloring problems we improved some of the previously known lower bounds for asymmetric codes and found the exact solutions for some of the instances.

The remainder of this chapter is organized as follows. In Section 5.2, we find lower bounds and exact solutions for the largest codes using efficient algorithms for the maximum independent set problem. In Section 5.3, a graph coloring heuristic

and the partitioning method are utilized in order to obtain better lower bounds for some asymmetric codes. Finally, concluding remarks are made in Section 5.4.

5.2 Finding Lower Bounds and Exact Sizes of the Largest Codes

In this section, we summarize some previously obtained results [56, 181]. We start with the following global optimization formulation for the maximum independent set problem, which is very similar to the formulation (**P1**) from Chapter 2.

Theorem 5.1. *The independence number of G satisfies the following equality:*

$$\alpha(G) = \max_{x\in[0,1]^n} \sum_{i=1}^{n} x_i \prod_{(i,j)\in E} (1 - x_j). \tag{5.1}$$

This formulation is valid if instead of $[0,1]^n$ we will use $\{0,1\}^n$ as the feasible region, thus obtaining an integer 0-1 programming problem. In problem (5.1), to each vertex i corresponds a Boolean expression:

$$i \longleftrightarrow r_i = x_i \bigwedge \left[\bigwedge_{(i,j)\in E} \overline{x_j} \right].$$

Therefore the problem of finding a maximum independent set can be reduced to the problem of finding a Boolean vector x^* which maximizes the number of "true" values among $r_i, i = 1, \ldots, n$.

$$x^* = \text{argmax} \sum_{i=1}^{n} \left| x_i \bigwedge \left[\bigwedge_{(i,j)\in E} \overline{x_j} \right] \right|. \tag{5.2}$$

To apply local search techniques for the above problem one needs to define a proper neighborhood. We define the neighborhood on the set of all maximal independent sets:

$$List_j = \left\{ i \notin I : r_i = \left[x_i \bigwedge_{(i,k)\in E} \bar{x}_k \right] \begin{array}{l} \text{has exactly 2 lite-} \\ \text{rals with value 0} \end{array} \right\}.$$

If the set $List_j$ is not empty, then the vertex j is added to this set.

If $I(G(List_{j_q}))$ is an arbitrary maximal independent set in $G(List_{j_q})$, then the sets of the form

$$(I - \{x_{j_q}\}) \cup I(G(List_{j_q})), q = 1, \ldots, |I|$$

are maximal independent sets in G. Therefore, the neighborhood of a maximal independent set I in G can be defined as follows:

$$O(I) = (I - \{x_{j_q}\}) \cup I(G(List_{j_q})),$$

$$j_q \in I, q = 1, \ldots |I|\}.$$

We have the following algorithm to find maximal independent sets:

1. Given a randomly generated Boolean vector x, find an appropriate initial maximal independent set I.

2. Find a maximal independent set from the neighborhood (defined for maximal independent sets) of I, which has the largest cardinality.

We tested the proposed algorithm with the following graphs arising from coding theory. These graphs are constructed as discussed in Section 5.1 and can be downloaded from Neil Sloane's website [187].

- Graphs From Single-Deletion-Correcting Codes (1dc);
- Graphs From Two-Deletion-Correcting Codes (2dc);
- Graphs From Codes For Correcting a Single Transposition, Excluding the End-Around Transposition (1tc);
- Graphs From Codes For Correcting a Single Transposition, Including the End-Around Transposition (1et);
- Graphs From Codes For Correcting One Error on the Z-Channel (1zc).

The results of the experiments are summarized in Table 5–1. In this table, the columns "Graph", "n", and "$|E|$" represent the name of the graph, the number of its vertices, and its number of edges. This information is available from the website [187]. The column "Solution found" contains the size of largest independent sets found by

Table 5–1: Lower bounds obtained.

Graph	n	$\|E\|$	Solution found
1dc128	128	1471	16
1dc256	256	3839	30
1dc512	512	9727	52
1dc1024	1024	24063	94
1dc2048	2048	58367	172
2dc128	128	5173	5
2dc256	256	17183	7
2dc512	512	54895	11
2dc1024	1024	169162	16
1tc64	64	192	20
1tc128	128	512	38
1tc256	256	1312	63
1tc512	512	3264	110
1tc1024	1024	7936	196
1tc2048	2048	18944	352
1et64	64	264	18
1et128	128	672	28
1et256	256	1664	50
1et512	512	4032	100
1et1024	1024	9600	171
1et2048	2048	220528	316
1zc128	128	1120	18
1zc256	256	2816	36
1zc512	512	6912	62
1zc1024	1024	16140	112
1zc2048	2048	39424	198
1zc4096	4096	92160	379

the algorithm over 10 runs. As one can see, the results are very encouraging. In fact, for all of the considered instances, they were at least as good as the best previously known estimates.

5.2.1 Finding the Largest Correcting Codes

The proposed exact algorithm consists of the following steps.

- Preprocessing: finding and removing the set of isolated cliques;
- Finding a partition of the graph into disjoint cliques;
- Finding an approximate solution;

- Finding an upper bound;
- Branch-and-Bound algorithm.

Below we give more detail on each of these steps.

0. **Preprocessing: finding and removing the set of isolated cliques.**

   We will call a clique C isolated if it contains a vertex i with the property $|N(i)| = |C| - 1$. Using the fact that if C is an isolated clique, then $\alpha(G) = \alpha(G - G(C)) + 1$, we iteratively find and remove all isolated cliques in the graph. After that, we consider each connected component of the obtained graph separately.

1. **Finding a partition of the graph into disjoint cliques.**

   We partition the set of vertices V of G as follows:

$$V = \bigcup_{i=1}^{k} C_i,$$

   where $C_i,\ i = 1, 2, \ldots, k$, are cliques such that $C_i \cap C_j = \emptyset,\ i \neq j$.

   The cliques are found using a simple greedy algorithm. Starting with $C_1 = \emptyset$, we pick vertex $j \notin C_1$ that has the maximal number of neighbors among those vertices outside of C_1 which are in neighborhood of every vertex from C_1. Set $C_1 = C_1 \bigcup \{j\}$, and repeat recursively, until there is no vertex to add. Then remove C_1 from the graph, and repeat the above procedure to obtain C_2. Continue in this way until the vertex set in the graph is empty.

2. **Finding an approximate solution.**

   An approximate solution is found using the approach described above.

3. **Finding an upper bound.**

   To obtain an upper bound for $\alpha(G)$ we can solve the following linear program:

$$O_{\mathcal{C}}(G) = \max \sum_{i=1}^{n} x_i, \tag{5.3}$$

$$\text{s. t.} \quad \sum_{i \in C_j} x_i \leq 1, j = 1, \ldots, m, \tag{5.4}$$

$$x \geq 0, \tag{5.5}$$

where $C_j \in \mathcal{C}$ is a maximal clique, $\mathcal{C}$ - a set of maximal cliques, $|\mathcal{C}|$ = m. For a general graph the last constraint should read $0 \leq x_i \leq 1,\ i = 1, \ldots, n$. But, since an isolated vertex is an isolated clique as well, after the preprocessing step our graph does not contain isolated vertices and the above inequalities are implied by the set of clique constraints (5.4) along with nonnegativity constraints (5.5). We call $O_{\mathcal{C}}(G)$ the linear clique estimate.

In order to find a tight bound $O_{\mathcal{C}}(G)$ one normally needs to consider a large number of clique constraints. Therefore, one deals with linear programs in which the number of constraints may be much larger than the number of variables. In this case it makes sense to consider the linear program which is dual to problem (5.3) - (5.5). The dual problem can be written as follows:

$$O_{\mathcal{C}}(G) = \min \sum_{j=1}^{m} y_j, \tag{5.6}$$

$$\text{s. t.} \quad \sum_{j=1}^{m} a_{ij} y_j \geq 1, i = 1, \ldots, n, \tag{5.7}$$

$$y \geq 0, \tag{5.8}$$

where

$$a_{ij} = \begin{cases} 1, \text{ if } i \in C_j, \\ 0, \text{ otherwise.} \end{cases}$$

The number of constraints in the last LP is always equal to the number of vertices in G. This gives us some advantages comparing to problem (5.3) - (5.5). If $m > n$ the dual problem is more suitable for solving with simplex method and interior point methods. Increasing the number of clique constraints in problem (5.3) - (5.5) leads only to increasing the number of variables in

problem (5.6) - (5.8). This provides a convenient "restart" scheme (start from an optimal solution to the previous problem) when additional clique constraints are generated.

To solve problem (5.6) - (5.8) we used a variation of interior point method proposed by Dikin [75, 76]. We will call this version of interior point method Dikin's Interior Point Method, or DIPM. We present a computational scheme of DIPM for an LP problem in the following form:

$$\min_{y \in R^{m+n}} \sum_{i=1}^{m+n} c_i y_i, \tag{5.9}$$

$$\text{s. t.} \qquad Ay = e, \tag{5.10}$$

$$y_i \geq 0, i = 1, \ldots, m+n. \tag{5.11}$$

Here A is $(m+n) \times n$ matrix in which first m columns are determined by coefficients a_{ij} and columns $a_{m+i} = -e_i$ for $i = 1, \ldots, n$, where e_i is the i^{th} orth. Vector $c \in R^{m+n}$ has the first m components equal to one and the rest n components equal to zero; $e \in R^n$ is the vector of all ones. Problem (5.6) - (5.8) can be reduced to this form if inequality constraints in (5.7) are replaced by equality constraints. As the initial point for the DIPM method we choose y^0, such that

$$y_i^0 = \begin{cases} 2, & \text{for } i = 1, \ldots, m; \\ 2\sum_{j=1}^{m} a_{ij} - 1, & \text{for } i = m+1, \ldots, m+n. \end{cases}$$

Now, let y^k is a feasible point for problem (5.9)- (5.11). In the DIPM method the next point y^{k+1} is obtained by the following scheme:

- Determine D_k of dimension $(m+n) \times (m+n)$ as $D_k = \text{diag}\{y^k\}$.

- Compute vector

$$c^p = (I - (AD_k)^T(AD_k^2A^T)^{-1}AD_k)D_kc.$$

- Find $\rho_k = \max\limits_{i=1,\ldots,m+n} c_i^p$.
- Compute $y_i^{k+1} = y_i^k\left(1 - \alpha\frac{c_i^p}{\rho_k}\right),\ i = 1,\ldots,m+n$, where $\alpha = 0.9$.

As the stopping criterion we used the condition

$$\sum_{j=1}^{m+n} c_j y_j^k - \sum_{j=1}^{m+n} c_j y_j^{k+1} < \varepsilon, \text{ where } \varepsilon = 10^{-3}.$$

The most labor-consuming operation of the method is the computation of the vector c^p. This part was implemented using subroutines DPPFA and DPPSL of LINPACK [77] for solving the following system of linear equations:

$$AD_k^2A^Tu_k = AD_k^2c, u_k \in R^n.$$

In this implementation the time complexity of one iteration of DIPM can be estimated as $O(n^3)$.

The values of vector $u_k = (AD_k^2A^T)^{-1}AD_k^2c$ found from the last system define dual variables in problem (5.6) - (5.8) (Lagrange multipliers for constraints (5.7)). The optimal values of dual variables were then used as weight coefficients for finding additional clique constraints, which help to reduce the linear clique estimate $O_{\mathcal{C}}(G)$. The problem of finding weighted cliques was solved using an approximation algorithm; the maximum of 1000 cliques were added to the constraints.

4. **Branch-and-Bound algorithm.**
   (a) Branching: Based on the fact that the number of vertices from a clique that can be included in an independent set is always equal to 0 or 1.
   (b) Bounding: We use the approximate solution found as a lower bound and the linear clique estimate $O_{\mathcal{C}}(G)$ as an upper bound.

Table 5–2: Exact algorithm: computational results.

Graph	#	1	2	3
1tc128	1	5	4	4.0002
	2	5	5	5.0002
	3	5	5	5.0002
	4	5	4	4.0001
1tc256	1	6	5	5.0002
	2	10	9	9.2501
	3	19	13	13.7501
	4	10	9	9.5003
	5	6	5	5.0003
1tc512	1	10	7	7.0003
	2	18	14	14. 9221
	3	29	22	23.6836
	4	29	22	23.6811
	5	18	14	14.9232
	6	10	7	7.0002
1dc512	1	75	50	51.3167
2dc512	1	16	9	10.9674
1et128	1	3	2	3.0004
	2	6	4	5.0003
	3	9	7	7.0002
	4	9	7	7.0002
	5	6	4	5.0003
	6	3	2	3.0004
1et256	1	3	2	3.0002
	2	8	6	6.0006
	3	14	10	12.0001
	4	22	12	14.4002
	5	14	10	12.0004
	6	8	6	6.0005
	7	3	2	3.0002
1et512	1	3	3	3.0000
	2	10	7	8.2502
	3	27	18	18.0006
	4	29	21	23.0626
	5	29	21	23.1029
	6	27	18	18.0009
	7	10	7	8.2501
	8	3	3	3.0000

Tables 5–2 and 5–3 contain a summary of the numerical experiments with the exact algorithm. In Table 5–2 column "#" contains a number assigned to each connected component of a graph after the preprocessing. Columns "1", "2" and

Table 5–3: Exact solutions found.

Graph	n	$\|E\|$	$\alpha(G)$	Time (sec)
1dc512	512	9727	52	2118
2dc512	512	54895	11	2618
1tc128	128	512	38	7
1tc256	256	1312	63	39
1tc512	512	3264	110	141
1et128	128	672	28	25
1et256	256	1664	50	72
1et512	512	4032	100	143

"3" stand for number of cliques in the partition, solution found by the approximation algorithm and the value of the upper bound $O_{\mathcal{C}}(G)$, respectively. In Table 5–3 column "$\alpha(G)$" contains the independence number of the corresponding instance found by the exact algorithm; column "Time" summarizes the total time needed to find $\alpha(G)$.

Among the exact solutions presented in Table 5–3, only two were previously known, for 2dc512 and 1et128. The rest were either unknown or were not proved to be exact.

5.3 Lower Bounds for Codes Correcting One Error on the Z-Channel

The error-correcting codes for Z-channel have very important practical applications. The Z-channel shown in Figure 5–1 is an asymmetric binary channel, in which the probability of transformation of 1 into 0 is p, and the probability of transformation of 0 into 1 is 0.

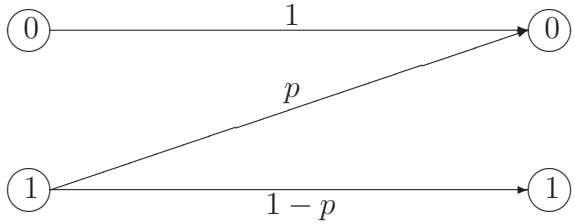

Figure 5–1: A scheme of the Z-channel.

The problem of our interest is to find good estimates for the size of the largest codes correcting one error on the Z-channel.

Let us introduce some background information related to *asymmetric codes*.

The asymmetric distance $d_A(x, y)$ between vectors $x, y \in B^n$ is defined as follows [177]:

$$d_A(x, y) = \max\{N(x, y), N(y, x)\}, \tag{5.12}$$

where $N(x, y) = |\{i : (x_i = 0) \wedge (y_i = 1)\}|$. It is related to the Hamming distance $d_H(x, y) = \sum_{i=1}^{n} |x_i - y_i| = N(x, y) + N(y, x)$ by the expression

$$2d_A(x, y) = d_H(x, y) + |w(x) - w(y)|, \tag{5.13}$$

where $w(x) = \sum_{i=1}^{n} x_i = |\{i : x_i = 1\}|$ is the weight of x. Let us define the minimum asymmetric distance Δ for a code $C \subset B^n$ as

$$\Delta = \min\{d_A(x, y) | x, y \in C, x \neq y\}.$$

Rao and Chawla [177] have shown that a code C with the minimum asymmetric distance Δ can correct at most $(\Delta - 1)$ asymmetric errors (transitions of 1 to 0). In this subsection we present new lower bounds for codes with the minimum asymmetric distance $\Delta = 2$.

Let us define the graph $G = (V(n), E(n))$, where the set of vertices $V(n) = B^n$ consists of all binary vectors of length n, and $(v_i, v_j) \in E(n)$ if and only if $d_A(v_i, v_j) < \Delta$. Then the problem of finding the size of the code with minimal asymmetric distance Δ is reduced to the maximum independent set problem in this graph. Table 5–4 contains the lower bounds obtained using the algorithm presented above in this section (some of them were mentioned in Table 5–1).

5.3.1 The Partitioning Method

The partitioning method [47, 81, 195] uses independent set partitions of the vertices of graph G in order to obtain a lower bound for the code size. An independent set partition is a partition of vertices into independent sets such that each vertex

Table 5–4: Lower bounds obtained by: a - Varshamov [196]; b - Constantin and Rao [64]; c - Delsarte and Piret [71]; d - Etzion and Östergård [81]; e - Butenko et al. ([55, 56], this chapter).

n	Lower bound	Upper bound
4	4	4
5	6a	6
6	12b	12
7	18c	18
8	36c	36
9	62c	62
10	112d	117
11	198d	210
12	379e (378d)	410

belongs to exactly one independent set, that is

$$V(n) = \bigcup_{i=1}^{m} I_i,\ I_i \text{ is an independent set},\ \ I_i \bigcap I_j = \emptyset,\ i \neq j. \tag{5.14}$$

Recall, that the problem of finding the smallest m for which a partition of the vertices into m disjoint independent sets exists is the well known graph coloring problem.

The independent set partition (5.14) can be identified by the vector

$$\Pi(n) = (I_1, I_2, \ldots, I_m).$$

With this vector, we associate the vector

$$\pi(n) = (|I_1|, |I_2|, \ldots, |I_m|),$$

which is called the *index vector* of partition $\Pi(n)$. Its norm is defined as

$$\pi(n) \cdot \pi(n) = \sum_{i=1}^{m} |I_i|^2.$$

We will assume that $|I_1| \geq |I_2| \geq \ldots \geq |I_m|$.

In terms of the codes, the independent set partition is a partition of words (binary vectors) into a set of codes, where each code corresponds to an independent set in the graph.

Similarly, for the set of all binary vectors of weight w we can construct a graph $G(n, w)$, in which the set of vertices is the set of the $\binom{n}{w}$ vectors, and two vertices are adjacent *iff* the Hamming distance between the corresponding vectors is less than 4. Then an independent set partition

$$\Pi(n, w) = (I_1^w, I_2^w, \ldots, I_m^w)$$

can be considered in which each independent set will correspond to a subcode with minimum Hamming distance 4. The index vector and its norm are defined in the same way as for $\Pi(n)$.

By the *direct product* $\Pi(n_1) \times \Pi(n_2, w)$ of a partition of asymmetric codes $\Pi(n_1) = (I_1, I_2, \ldots, I_{m_1})$ and a partition of constant weight codes $\Pi(n_2, w) = (I_1^w, I_2^w, \ldots, I_{m_2}^w)$ we will mean the set of vectors

$$C = \{(u, v) : \ u \in I_i, \ v \in I_i^w, \ 1 \leq i \leq m\},$$

where $m = \min\{m_1, m_2\}$. It appears that C is a code of length $n = n_1 + n_2$ with minimum asymmetric distance 2, *i.e.*, a code correcting one error on the Z-channel of length $n = n_1 + n_2$ [81].

In order to find a code C of length n and minimum asymmetric distance 2 by the partitioning method, we can use the following construction procedure.

1. Choose n_1 and n_2 such that $n_1 + n_2 = n$.
2. Choose $\epsilon = 0$ or 1.
3. Set

$$C = \bigcup_{i=0}^{\lfloor n_2/2 \rfloor} \left(\Pi(n_1) \times \Pi(n_2, 2i + \epsilon)\right). \tag{5.15}$$

5.3.2 The Partitioning Algorithm

One of the popular heuristic approaches to the independent set partitioning (graph coloring) problem is the following. Suppose that a graph $G = (V, E)$ is given.

INPUT: $G = (V, E)$;

OUTPUT: $I_1, I_2, \ldots, I_m$.

0. i=1;

1. **while** $G \neq \emptyset$

Find a maximal independent set I; set $I_i = I$; $i = i + 1$;

$G = G - G(I)$, where $G(I)$ is the subgraph induced by I;

end

Shylo and Boyarchuk [184, 185] proposed to improve this approach by finding at each step a specified number of maximal independent sets. Then a new graph $\mathcal{G}$ is constructed, in which a vertex corresponds to a maximal independent set, and two vertices are adjacent *iff* the corresponding independent sets have common vertices. In the graph $\mathcal{G}$, a few maximal independent sets are found, and the best of them (say, the one with the least number of adjacent edges in the corresponding independent sets of G) is chosen. This approach is formally described in Figure 5–2.

INPUT: $G = (V, E)$;
OUTPUT: $I_1, I_2, \ldots, I_m$.
0. i=0;
1. **while** $G \neq \emptyset$
for $j = 1$ **to** k
Find a maximal independent set IS_j;
if $|IS_j| < |IS_{j-1}|$ **break**
end
Construct graph $\mathcal{G}$;
Find a maximal independent set $MIS = \{IS_{i_1}, \ldots, IS_{i_p}\}$ of $\mathcal{G}$;
$I_{i+q} = IS_{i_q},\ q = 1, \ldots, p$;
$G = G - \bigcup_{q=1}^{p} G(I_{i+q});\ \ i = i + p$;
end

Figure 5–2: Algorithm for finding independent set partitions.

5.3.3 Improved Lower Bounds for Code Sizes

The partitions obtained using the described partition algorithm are given in Tables 5–5 and 5–6. These partitions, together with the facts that [80]

Table 5–5: Partitions of asymmetric codes found.

n	#	Partition index vector	Norm	m
8	1	36,34, 34, 33, 30, 29, 26, 25, 9	7820	9
9	1	62, 62, 62, 61, 58, 56, 53, 46, 29, 18, 5	27868	11
	2	62, 62, 62, 62, 58, 56, 53, 43, 32, 16, 6	27850	11
	3	62, 62, 62, 61, 58, 56, 52, 46, 31, 17, 5	27848	11
	4	62, 62, 62, 62, 58, 56, 52, 43, 33, 17, 5	27832	11
	5	62, 62, 62, 62, 58, 56, 54, 42, 31, 15, 8	27806	11
	6	62, 62, 62, 60, 57, 55, 52, 45, 31, 18, 8	27794	11
	7	62, 62, 62, 60, 58, 55, 51, 45, 37, 16, 4	27788	11
	8	62, 62, 62, 60, 58, 56, 53, 45, 32, 16, 6	27782	11
	9	62, 62, 62, 62, 58, 56, 52, 43, 32, 17, 6	27778	11
	10	62, 62, 62, 60, 58, 56, 53, 45, 31, 18, 5	27776	11
	11	62, 62, 62, 62, 58, 56, 50, 45, 32, 18, 5	27774	11
	12	62, 62, 62, 61, 58, 56, 51, 45, 30, 22, 3	27772	11
	13	62, 62, 62, 62, 58, 56, 50, 44, 34, 16, 6	27760	11
	14	62, 62, 62, 62, 58, 55, 51, 44, 32, 20, 4	27742	11
10	1	112, 110, 110, 109, 105, 100, 99, 88, 75, 59, 37, 16, 4	97942	13
	2	112, 110, 110, 109, 105, 101, 96, 87, 77, 60, 38, 15, 4	97850	13
	3	112, 110, 110, 108, 106, 99, 95, 89, 76, 60, 43, 15, 1	97842	13
	4	112, 110, 110, 108, 105, 100, 96, 88, 74, 65, 38, 17, 1	97828	13
	5	112, 110, 110, 108, 106, 103, 95, 85, 76, 60, 40, 15, 4	97720	13
	6	112, 110, 110, 108, 106, 101, 95, 87, 75, 61, 40, 17, 2	97678	13
	7	112, 110, 109, 108, 105, 101, 96, 86, 78, 63, 36, 17, 3	97674	13

$\Pi(n, 0)$ consists of one (zero) codeword,

$\Pi(n, 1)$ consists of n codes of size 1,

$\Pi(n, 2)$ consists of $n - 1$ codes of size $n/2$ for even n,

Index vectors of $\Pi(n, w)$ and $\Pi(n, n - w)$ are equal;

were used in (5.15), with $\epsilon = 0$, to obtain new lower bounds for asymmetric codes presented in Table 5–7. To illustrate how the lower bounds were computed, let us show how the code for $n = 18$ was constructed. We use $n_1 = 8,\ n_2 = 10$.

$|\Pi(8) \times \Pi(10, 0)| = 36 \cdot 1 = 36$;

$|\Pi(8) \times \Pi(10, 2)| = 256 \cdot 5 = 1280$;

$|\Pi(8) \times \Pi(10, 4)| = 36 \cdot 30 + 34 \cdot 30 + 34 \cdot 30 + 33 \cdot 30 + 30 \cdot 26 + 29 \cdot 25 + 26 \cdot 22 + 25 \cdot 15 + 9 \cdot 2 = 6580$;

Table 5–6: Partitions of constant weight codes obtained in: a-this chapter; b-Brouwer et al. [47]; c-Etzion and Östergård [81].

k	w	#	Partition index-vector	Norm	m
10	4	1a	30, 30, 30, 30, 26, 25, 22, 15, 2	5614	9
12	4	1a	51, 51, 51, 51, 49, 48, 48, 42, 42, 37, 23, 2	22843	12
12	4	2a	51, 51, 51, 51, 49, 48, 48, 45, 39, 36, 22, 4	22755	12
12	4	3a	51, 51, 51, 51, 49, 48, 48, 45, 41, 32, 22, 6	22663	12
12	6	1a	132, 132, 120, 120, 110, 94, 90, 76, 36, 14	99952	10
14	4	1c	91, 91, 88, 87, 84, 82, 81, 79, 76, 73, 66, 54, 38, 11	78399	14
14	4	2c	91, 90, 88, 85, 84, 83, 81, 79, 76, 72, 67, 59, 34, 11, 1	78305	15
14	6	1b	278, 273, 265, 257, 250, 231, 229, 219, 211, 203, 184, 156, 127, 81, 35, 4	672203	16

Table 5–7: New lower bounds. Previous lower bounds were found by: a-Etzion [80]; b-Etzion and Östergård [81].

	Lower bound	
n	new	previous
18	15792	15762a
19	29478	29334b
20	56196	56144b
21	107862	107648b
22	202130	201508b
24	678860	678098b

$|\Pi(8) \times \Pi(10,6)| = |\Pi(8) \times \Pi(10,4)| = 6580;$

$|\Pi(8) \times \Pi(10,8)| = |\Pi(8) \times \Pi(10,2)| = 1280;$

$|\Pi(8) \times \Pi(10,10)| = |\Pi(8) \times \Pi(10,0)| = 36;$

The total is $2(36 + 1280 + 6580) = 15792$ codewords.

5.4 Conclusion

In this chapter we deal with binary codes of given length correcting certain types of errors. For such codes, a graph can be constructed, in which each vertex corresponds to a binary vector and the edges are built in the way, that each independent set corresponds to a correcting code. The problem of finding the largest code is thus reduced to the maximum independent set problem in the corresponding graph. For asymmetric codes, we also applied the partitioning method, which utilizes

independent set partitions (or graph colorings) in order to obtain lower bounds for the maximum code sizes.

We use efficient approaches to the maximum independent set and graph coloring problems to deal with the problem of estimating the largest code sizes. As a result, some improved lower bounds and exact solutions for the size of largest error-correcting codes were obtained.

CHAPTER 6
APPLICATIONS IN WIRELESS NETWORKS

6.1 Introduction

Ad hoc wireless networks have applications in emergency search-and-rescue operations, decision making in the battlefield, data acquisition operations in inhospitable terrain, etc. They are featured by dynamic topology (infrastructureless), multihop communication, limited resources (bandwidth, CPU, battery, etc) and limited security. These characteristics put special challenges in routing protocol design.

Existing routing protocols can be classified into three categories: *proactive*, *reactive* and the combination of the two [132, 134, 175, 176]. Proactive routing protocols ask each host to maintain global topology information, thus a route can be provided immediately when requested. Protocols in this category suffer from lower scalability and high protocol overhead. Reactive routing protocols have the feature *on-demand*. Each host computes route for a specific destination only when necessary. Topology changes that do not influence active routes do not trigger any route maintenance function. Thus, communication overhead is lower compared to proactive routing protocol. The third category maintains partial topology information in some hosts. Routing decisions are made either proactively or reactively. One important observation on these protocols is that none of them can avoid the involvement of *flooding*. For example, proactive protocols rely on flooding for the dissemination of topology update packets, and reactive protocols rely on flooding for route discovery.

Flooding suffers from the notorious *broadcast storm problem* [166]. Broadcast storm problem refers to the fact that flooding may result in excessive *redundancy*, *contention*, and *collision*. This causes high protocol overhead and interference to

ongoing traffic. On the other hand, flooding is very *unreliable* [133], which means that *not all* hosts get all the broadcast messages when free from collisions. Sinha et. al. [186] claimed that "in moderately sparse graphs the expected number of nodes in the network that will receive a broadcast message was shown to be as low as 80%." In reactive protocols, the unreliability of flooding may obstruct the detection of the shortest path, or simply cannot detect any path at all, even though there exists a path. In proactive protocols, the unreliability of flooding may cause the global topology information to become obsolete, thus causing the newly-computed path obsolescence.

Recently an approach based on overlaying a virtual infrastructure on an ad hoc network was proposed by Sinha et al.[186]. Routing protocols are operated over this infrastructure, which is termed *core*. All core hosts form a dominating set. The key feature in this approach is the new *core broadcast mechanism* which uses unicast to replace the flooding mechanism used by most on-demand routing protocols. The unicast of route requests packets to be restricted to core nodes and a (small) subset of non-core nodes. Simulation results when running DSR (Dynamic Source Routing [134]) and AODV (Ad hoc On-demand Distance Vector routing [176]) over the core indicate that the core structure is *effective* in enhancing the performance of the routing protocols. Prior to this work, inspired by the *physical backbone* in a wired network, many researchers proposed the concept of *virtual backbone* for unicast, multicast/broadcast in ad hoc wireless networks [17, 67, 202].

In this chapter, we will study the problem of efficiently constructing a virtual backbone for ad hoc wireless networks. The number of hosts forming the virtual backbone must be as small as possible to decrease protocol overhead. The algorithm must be time/message efficient due to resource scarcity. We use a connected dominating set (CDS) to approximate the virtual backbone (see Figure 6–1). We assume a given ad hoc network instance contains n hosts. Each host is in the ground

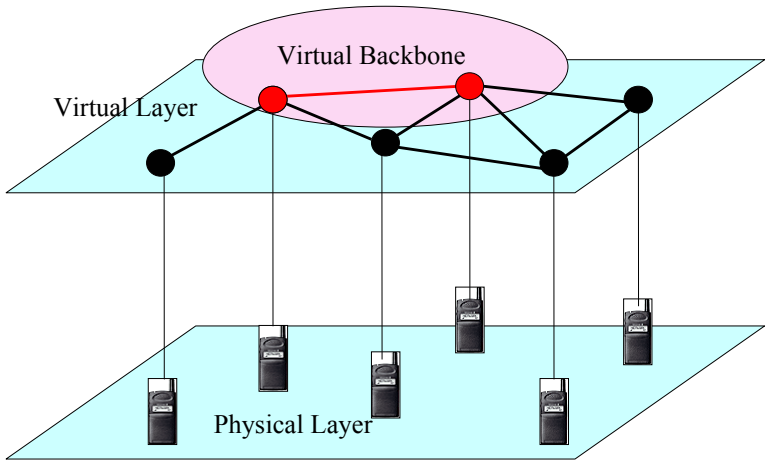

Figure 6–1: Approximating the virtual backbone with a connected dominating set in a unit-disk graph

and is mounted by an omni-directional antenna. Thus the transmission range of a host is a disk. We further assume that each transceiver has the same communication range R. Thus the footprint of an ad hoc wireless network is a unit-disk graph. In graph-theoretic terminology, the network topology we are interested in is a graph $G = (V, E)$ where V contains all hosts and E is the set of links. A link between u and v exists if they are separated by the distance of at most R. In a real world ad hoc wireless network, sometimes even when v is located in u's transmission range, v is not reachable from u due to *hidden/exposed terminal problems*. But in this chapter we only consider bidirectional links. From now on, we use host and node interchangeably to represent a wireless mobile.

There exist several distributed algorithms for MCDS computation in the context of ad hoc wireless networking [14, 67, 202]. The algorithm of Alzoubi et al. [14] first builds a rooted tree in a distributed fashion. Then the status (inside or outside of the CDS) is assigned for each host based on the level of the host in the tree. Das and Bharghavan [67] provide the distributed implementation of the two centralized algorithms given by Guha and Khuller [109]. Both implementations suffer from high message complexities. The one given by Wu and Li in [202] has no performance

analysis, but it needs at least two-hop neighborhood information. The status of each host is assigned based on the connectivity of its neighbors. We will compare our algorithm with the other approaches [14, 67, 202] in Sections 6.2 and 6.3.

We adopt the following definitions and notations. A *dominating set (DS)* D of G is a subset of V such that any node not in D has at least one neighbor in D. If the induced subgraph of D is connected, then D is a *connected dominating set (CDS)*. Among all CDSs of graph G, the one with minimum cardinality is called a *minimum connected dominating set (MCDS)*. Computing an MCDS in a unit graph is NP-hard [63]. Note that the problem of finding an MCDS in a graph is equivalent to the problem of finding a spanning tree (ST) with maximum number of leaves. All non-leaf nodes in the spanning tree form the MCDS. It is easy to prove that any maximal independent set (MIS) is a dominating set.

For a graph G, if $e = (u, v) \in E \ \ iff \ \ length(e) \leq 1$, then G is called a *unit-disk graph*. We will only consider unit-disk graphs in this chapter. From now on, when we say a "graph G", we mean a "unit-disk graph G". The following lemma was proved by Alzoubi et al. [14]. This lemma relates the size of any MIS of unit-disk graph G to the size of its optimal CDS.

Lemma 6.1. *[14] The size of any MIS of G is at most $4 \times opt + 1$, where opt is the size of any MCDS of G.*

For a minimization problem $\mathcal{P}$, the *performance ratio* of an approximation algorithm A is defined to be $\rho_A = sup_{i \in I} \frac{A_i}{opt_i}$, where I is the set of instances of $\mathcal{P}$, A_i is the output from A for instance i and opt_i is the optimal solution for instance i.

The remainder of this chapter is organized as follows. Our algorithm and its theoretic performance analysis are presented in Section 6.2. Results of numerical experiments are demonstrated in Section 6.3. Section 6.4 concludes the chapter.

6.2 An 8-Approximate Algorithm to Compute CDS

In this section, we propose a distributed algorithm to compute CDS. This algorithm contains two phases. First, we compute a maximal independent set (MIS); then we use a tree to connect all vertices in the MIS. We will show that our algorithm has performance ratio at most 8 and is message and time efficient.

6.2.1 Algorithm Description

Initially each host is colored *white*. A dominator is colored *black*, while a dominatee is colored *gray*. The *effective degree* of a vertex is the total number of white neighbors. We assume that each vertex knows its distance-one neighbors and their effective degrees d^*. This information can be collected by periodic or event-driven hello messages.

We also designate one host as the *leader*. This is a realistic assumption. For example, the leader can be the commander's mobile for a platoon of soldiers in a mission. If it is impossible to designate any leader, a distributed leader-election algorithm can be applied to find out a leader. This adds message and time complexity. The best leader-election algorithm [24] takes time $O(n)$ and message $O(n\ log\ n)$ and these are the best-achievable results. Assume host s is the leader.

Phase 1. Host s first colors itself black and broadcasts message DOMINATOR. Any white host u receiving DOMINATOR message the first time from v colors itself gray and broadcasts message DOMINATEE. u selects v as its dominator. A white host receiving at least one DOMINATEE message becomes active. An active white host with highest (d^*, id) among all of its active white neighbors will color itself black and broadcast message DOMINATOR. A white host decreases its effective degree by 1 and broadcasts message DEGREE whenever it receives a DOMINATEE message. Message DEGREE contains the sender's current effective degree. A white vertex receiving a DEGREE message will update its neighborhood information accordingly. Each gray vertex will broadcast message

NUMOFBLACKNEIGHBORS when it detects that none of its neighbors is white. Phase 1 terminates when no white vertex left.

Phase 2. When s receives message NUMOFBLACKNEIGHBORS from all of its gray neighbors, it starts phase 2 by broadcasting message M. A host is "ready" to be explored if it has no white neighbors. We will use a tree to connect all black hosts generated in Phase 1. The idea is to pick those gray vertices which connect to many black neighbors. We will modify the classical distributed depth first search spanning tree algorithm of Attiya and Welch [22] to compute the tree.

A black vertex without any dominator is *active*. Initially no black vertex has a dominator and all hosts are *unexplored*. Message M contains a field *next* which specifies the next host to be explored. A gray vertex with at least one active black neighbor is *effective*. If M is built by a black vertex, its *next* field contains the *id* of the unexplored gray neighbor which connects to maximum number of active black hosts. If M is built by a gray vertex, its *next* field contains the *id* of any unexplored black neighbor. Any black host u receiving a M message the first time from a gray host v sets its dominator to v by broadcasting message PARENT. When a gray host u receives message M from v that specifies u to be explored next, u then colors itself black, sets its dominator to v and broadcasts its own M message. Any gray vertex receiving message PARENT from a black neighbor will broadcast message NUMOFBLACKNEIGHBORS, which contains the number of active black neighbors. A black vertex becomes *inactive* after its dominator is set. A gray vertex becomes *ineffective* if none of its black neighbors is active. A gray vertex without active black neighbor, or a black vertex without effective gray neighbor, will send message DONE to the host which activates its exploration or to its dominator. When s gets message DONE and it has no effective gray neighbors, the algorithm terminates.

Note that phase 1 sets the dominators for all gray vertices. Phase 2 may modify the dominator of some gray vertex. The main job for phase 2 is to set a dominator for each black vertex. All black vertices form a CDS.

In Phase 1, each host broadcasts each of the messages DOMINATOR and DOMINATEE at most once. The message complexity is dominated by message DEGREE, since it may be broadcasted Δ times by a host, where Δ is the maximum degree. Thus the message complexity of Phase 1 is $O(n \cdot \Delta)$. The time complexity of Phase 1 is $O(n)$.

In phase 2, vertices are explored one by one. The total number of vertices explored is the size of the output CDS. Thus the time complexity is at most $O(n)$. The message complexity is dominated by message NUMOFBLACKNEIGHBORS, which is broadcasted at most 5 times by each gray vertex because a gray vertex has at most 5 black neighbors in a unit-disk graph. Thus the message complexity is also $O(n)$.

From the above analysis, we have

Theorem 6.1. *The distributed algorithm has time complexity $O(n)$ and message complexity $O(n \cdot \Delta)$.*

Note that in phase 1 if we use (id) instead of (d^*, id) as the parameter to select a white vertex to color it black, the message complexity will be $O(n)$ because no DEGREE messages will be broadcasted. $O(n \cdot \Delta)$ is the best result we can achieve if effective degree is taken into consideration.

6.2.2 Performance Analysis

In this subsection, we study the performance of our algorithm.

Lemma 6.2. *Phase 1 computes an MIS which contains all black nodes.*

Proof. A node is colored black only from white. No two white neighbors can be colored black at the same time since they must have different (d^*, id). When a node

is colored black, all of its neighbors are colored gray. Once a node is colored gray, it remains in color gray during Phase 1. □

From the proof of Lemma 6.2, it is clear that if (id) instead of (d^*, id) is used, we still get an MIS. Intuitively, this would yield an MIS of a larger size.

Lemma 6.3. *In phase 2, at least one gray vertex which connects to maximum number of black vertices will be selected.*

Proof. Let u be a gray vertex with maximum number of black neighbors. At some step in phase 2, one of u's black neighbors v will be explored. In the following step, u will be explored. This exploration is triggered by v. □

Lemma 6.4. *If there are c black hosts after phase 1, then at most $c-1$ gray hosts will be colored black in phase 2.*

Proof. In phase 2, the first gray vertex selected will connect to at least 2 black vertices. In the following steps, any newly selected gray vertex will connect to at least one new black vertex. □

Lemma 6.5. *If there exists a gray vertex which connects to at least 3 black vertices, then the number of gray vertices which are colored black in phase 2 will be at most $c-2$, where c is the number of black vertices after phase 1.*

Proof. From Lemma 6.3, at least one gray vertex with maximum black neighbors will be colored black in phase 2. Denote this vertex by u. If u is colored black, then all of its black neighbors will choose u as its dominator. Thus, the selection of u causes more than 1 black hosts to be connected. □

Theorem 6.2. *Our algorithm has performance ratio at most 8.*

Proof. From Lemma 6.2, phase 1 computes an MIS. We will consider two cases here.

If there exists a gray vertex which has at least 3 black neighbors after phase 1, from Lemma 6.1, the size of the MIS is at most $4 \cdot opt + 1$. From lemma 6.5, we know

the total number of black vertices after phase 2 is at most $4 \cdot opt + 1 + ((4 \cdot opt + 1) - 2) = 8 \cdot opt$.

If the maximum number of black neighbors a gray vertex has is 2, then the size of the MIS computed in phase 1 is at most $2 \cdot opt$ since any vertex in MCDS connects to at most 2 vertices in the MIS. Thus from Lemma 6.4, the total number of black hosts will be $2 \cdot opt + 2 \cdot opt - 1 < 4 \cdot opt$. □

Note that from the proof of Theorem 6.2, if (id) instead of (d^*, id) is used in phase 1, our algorithm still has performance ratio at most 8.

We compare the theoretical performance of our algorithm with other algorithms proposed in the literature [14, 67, 202] in Table 6–1. The parameters used for comparison include the (upper bound of the) cardinality of the generated CDS (CDSC), the message and time complexities (MC and TC, respectively), the message length (ML) and neighborhood information (NI).

Table 6–1: Performance comparison of the algorithms of Das and Bharghavan [67], Wu and Li [202], and Alzoubi et al. [14] and the one proposed in this chapter. Here *opt* is the size of the MCDS; Δ is the maximum degree; $|C|$ is the size of the generated connected dominating set; m is the number of edges; n is the number of hosts.

	[67]-I	[67]-II	[202]	[14]	A
CDSC	$\leq (2ln\Delta + 3)opt$	$\leq (2ln\Delta + 2)opt$	N/A	$\leq 8\ opt + 1$	$< 8\ opt$
MC	$O(n\|C\| + m + nlog\ n)$	$O(n\|C\|)$	$O(n\Delta)$	$O(nlog\ n)$	$O(n)$
TC	$O((n + \|C\|)\Delta)$	$O(\|C\|(\Delta + \|C\|))$	$O(\Delta^2)$	$O(n\Delta)$	$O(n\Delta)$
ML	$O(\Delta)$	$O(\Delta)$	$O(\Delta)$	$O(1)$	$O(1)$
NI	2-hop	2-hop	2-hop	1-hop	1-hop

Note that the last column (labelled by A) in Table 6–1 corresponds to our algorithm. It can be seen that our algorithm is superior to the two algorithms of Das and Bharghavan [67] with respect to all parameters. Algorithm of Wu and Li [202] takes less time than our algorithm but it has much higher message complexity and it uses more complicated message information. The algorithm of Alzoubi et al. [14] is comparable with our algorithms in many parameters. But the simulation

results in Section 6.3 show that our algorithm computes smaller on average connected dominating sets for both random and unform graphs.

6.3 Numerical Experiments

Table 6–1 in the previous section compares our algorithms with others theoretically. In this section, we will compare the size of the CDSs computed by different algorithms. As mentioned earlier, the virtual backbone is mainly used to disseminate control packets. Thus the most important parameter is the number of hosts in the virtual backbone after it is constructed. The larger the size of a virtual backbone, the larger number of transmissions to broadcast a message to the whole network is needed. Note that the message complexities of the algorithms of Das and Bharghavan [67] and Wu and Li [202] are too high compared to other algorithms and they need 2-hop neighborhood information. Thus we will not consider them in the simulation study. We will compare our algorithm with the one given by Alzoubi et al. [14].

We will consider two kinds of topologies: random and uniform. We assume there are N hosts distributed randomly or uniformly in a 100×100 square units. Transmission range R is chosen to be 15, 25 or 50 units. For each value of R, we run our algorithms 100 times for different values of N. The averaged results are reported in Figures 6–2, 6–3 and 6–4 for random graphs, and in Figures 6–5, 6–6 and 6–7 for uniform graphs. From these figures it is clear that in all of our simulation scenarios, our algorithm performs better than the algorithm of Alzoubi et al. [14].

6.4 Conclusion

In this chapter we provide a distributed algorithm which computes a connected dominating set of a small size. Our algorithm has performance ratio at most 8. This algorithm takes time $O(n)$ and message $O(n \cdot \Delta)$. Our future work is to investigate the performance of virtual backbone routing and to study the problem of maintaining the connected dominating set in a mobility environment.

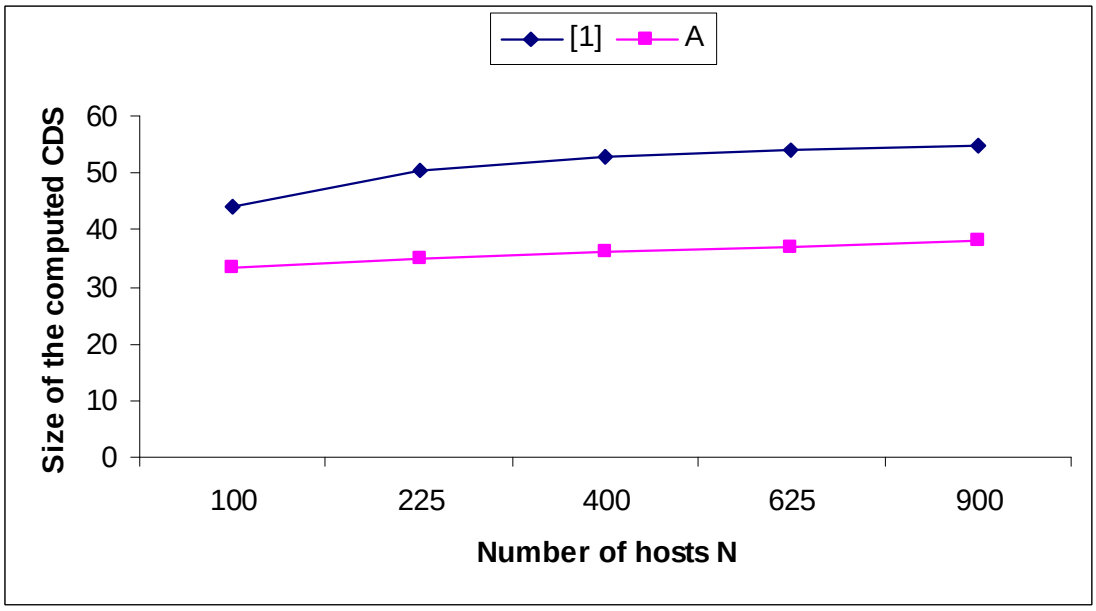

Figure 6–2: Averaged results for $R = 15$ in random graphs.

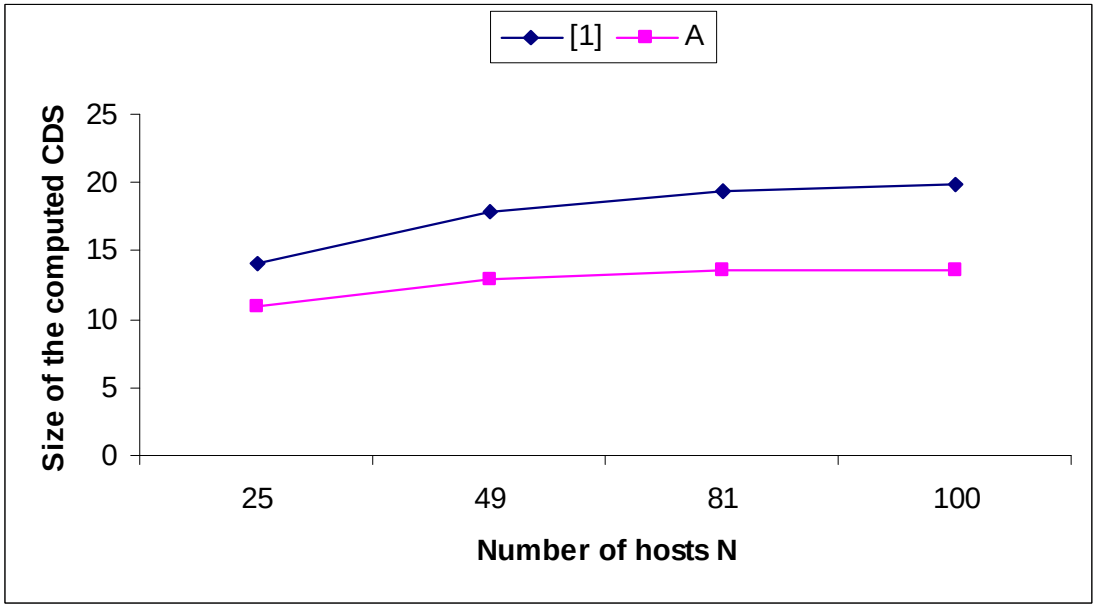

Figure 6–3: Averaged results for $R = 25$ in random graphs.

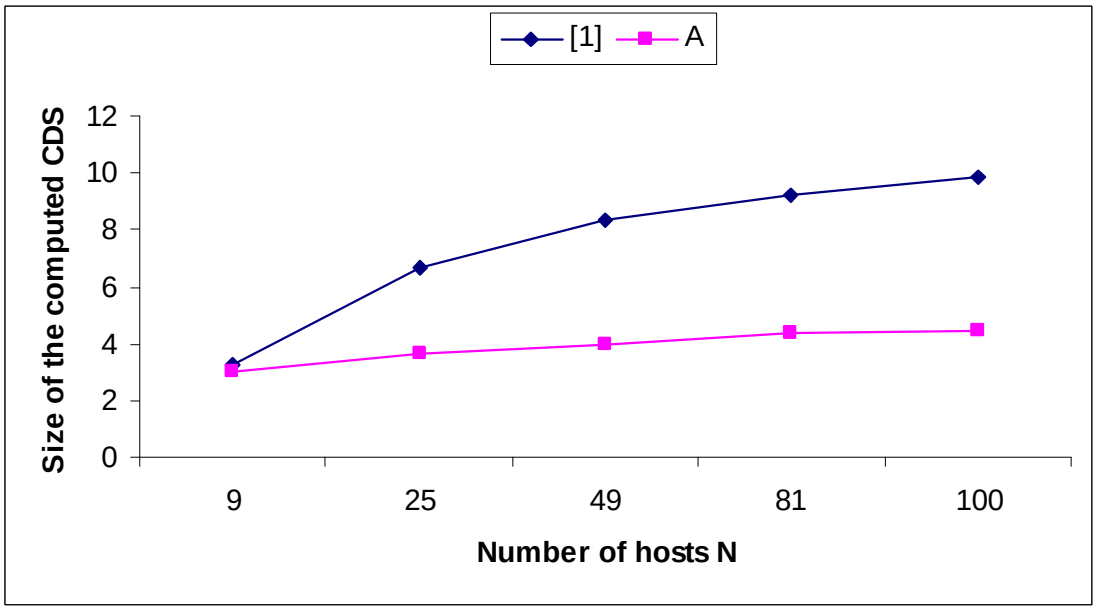

Figure 6–4: Averaged results for $R = 50$ in random graphs.

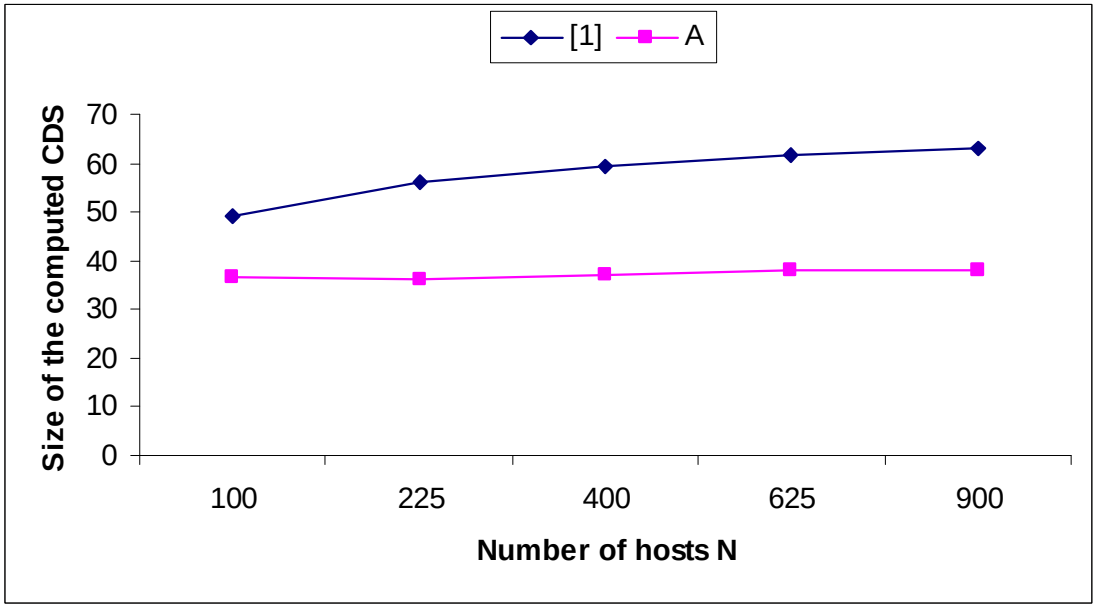

Figure 6–5: Averaged results for $R = 15$ in uniform graphs.

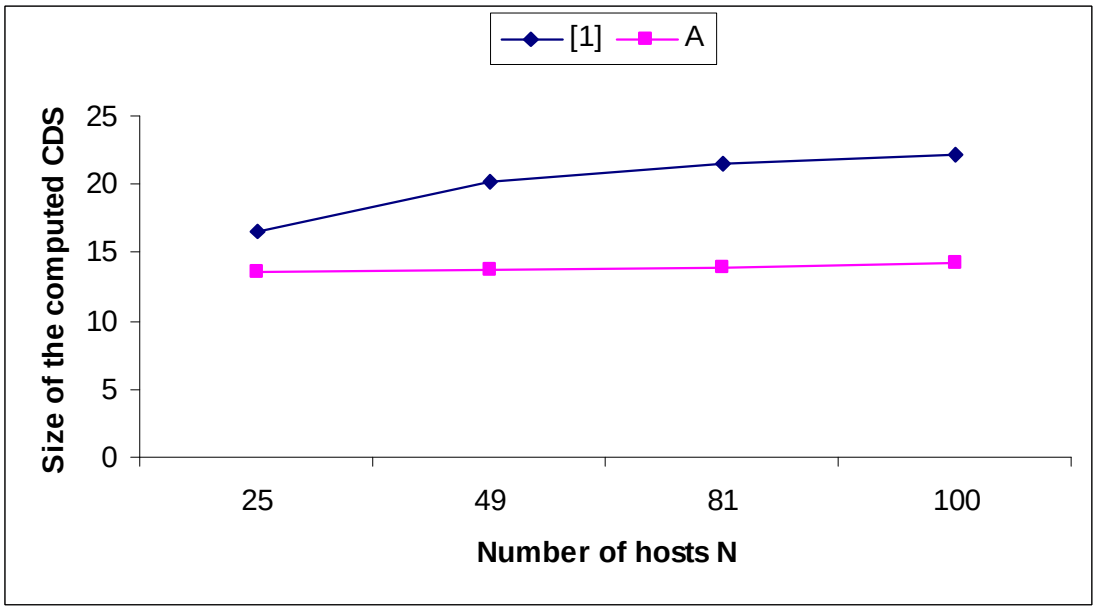

Figure 6–6: Averaged results for $R = 25$ in uniform graphs.

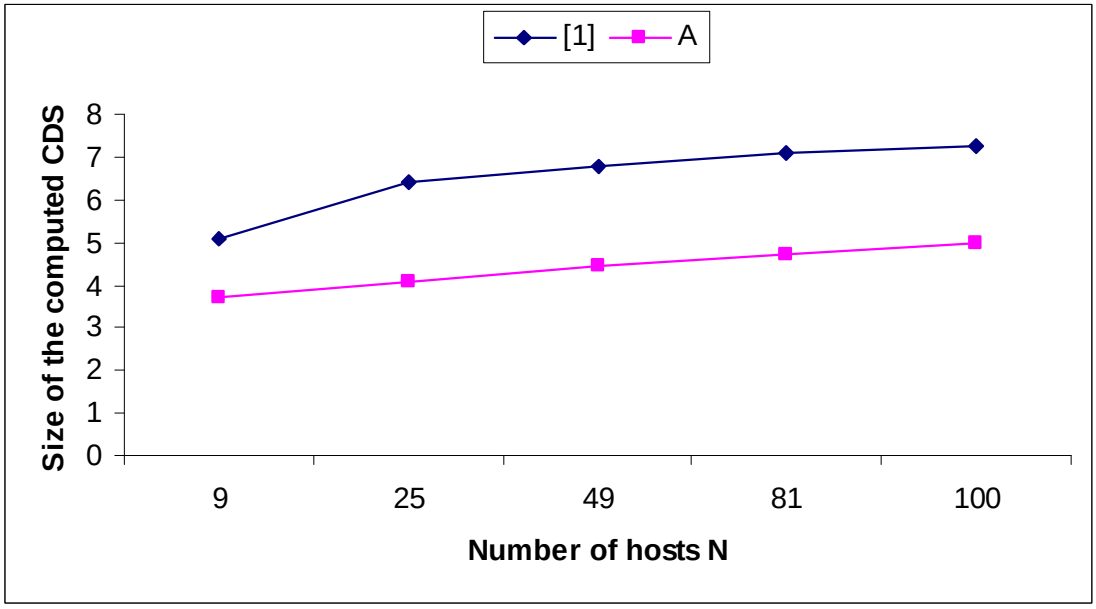

Figure 6–7: Averaged results for $R = 50$ in uniform graphs.

CHAPTER 7
CONCLUSIONS AND FUTURE RESEARCH

We presented new efficient approaches to solving the maximum independent set and related combinatorial optimization problems. In Chapter 2 we proved several continuous optimization formulations. These formulations and some classical results concerning the problem of optimizing a quadratic over a sphere were used to develop several heuristics for the maximum independent set problem in Chapter 3. In the following section we will show how these approaches can be extended to another important combinatorial optimization problem: the MAX-CUT problem. Then, in Section 7.2 we present a new idea of using the so-called *critical* sets for the maximum independent set problem solving. Finally, in Section 7.3 we discuss applications considered in this dissertation and the directions of future work concerning the applications.

7.1 Extensions to the MAX-CUT Problem

Given an undirected graph with edge weights, the MAX-CUT problem is to find a partition of the set of vertices into two subsets, such that the sum of the weights of the edges having endpoints in different subsets is maximized. This well-known NP-hard problem has applications in VLSI design, statistical physics, and several other fields [51, 73, 87, 111]. Similarly to the maximum independent set problem, the MAX-CUT problem can be formulated in terms of optimization of a quadratic over the unit hypercube.

Theorem 7.1. *Given a graph $G = (V, E)$ with vertex set $V = \{1, 2, \ldots, n\}$ and a set of weighted edges E, the optimal objective function value of MAX-CUT problem*

is given by

$$\max_{x\in[0,1]^n} x^T W(e-x), \tag{7.1}$$

where $W=[w_{ij}]_{i,j=1}^n$ *is the matrix of edge weights, and* $e=[1,1,\ldots,1]^T$ *is the unit vector.*

Proof. Denote the objective of (7.1) by

$$f(x)=x^T W(e-x).$$

First, note that matrix W has zero diagonal, therefore $f(x)$ is linear with respect to each variable, and there always exist an optimal solution $\hat{x}\in\{0,1\}^n$ of (7.1), *i.e.*,

$$\max_{x\in[0,1]^n} x^T W(e-x)=\max_{x\in\{0,1\}^n} x^T W(e-x).$$

Next, for any binary vector $\bar{x}\in\{0,1\}^n$, consider the partition of the set of vertices into two nonoverlapping sets $V_1=\{i|\,\bar{x}_i=0\}$ and $V_2=\{i|\,\bar{x}_i=1\}$. Then

$$f(\bar{x})=\sum_{(i,j)\in V_1\times V_2} w_{ij},$$

which is the value of the cut defined by the partition $V=V_1\bigcup V_2$.

Alternatively, we consider any partition of vertex set into two nonoverlapping sets,

$$V=V_1\bigcup V_2,\ \ V_1\bigcap V_2=\emptyset,$$

and construct the vector $\bar{x}$, such that

$$x_i=\begin{cases} 0,\ \text{if } i\in V_1,\\ 1,\ \text{if } i\in V_2.\end{cases}$$

Then, again

$$f(\bar{x})=\sum_{(i,j)\in V_1\times V_2} w_{ij}.$$

Therefore, there is a one-to-one correspondence between binary vectors of length n and cuts in G. So,

$$\max_{x\in[0,1]^n} x^T W(e-x) = \max_{x\in\{0,1\}^n} x^T W(e-x) = \max_{V=V_1\bigcup V_2,\, V_1\bigcap V_2=\emptyset} \sum_{(i,j)\in V_1\times V_2} w_{ij}$$

□

Analogous formulations for the maximum independent set problem were used to derive efficient heuristics in Chapter 3. Thus, it would be interesting to try similar approaches for the MAX-CUT problem. In future research, when using similar techniques for combinatorial optimization problems, one should try to address the following issues:

- How to optimally choose the parameters of the quadratic objective and of the sphere used as the feasible set?
- What is the optimal way to extract the sought combinatorial object from the information provided by stationary (or optimal) points of the considered quadratic problem?

To answer these questions both theoretical and empirical studies are required.

7.2 Critical Sets and the Maximum Independent Set Problem

In this section we discuss a possibility of using properties of critical independent sets in studying the maximum independent set problem.

Let $G = (V, E)$ be a simple undirected graph with the vertex set $V = \{1, 2, \ldots, n\}$ and the set of edges E. For a set $I \subseteq V$, let $N(I)$ denote the set of all vertices of G, which are adjacent to at least one vertex of I. An independent set I_c (possibly empty) is called *critical* if

$$|I_c| - |N(I_c)| = \max\{|I| - |N(I)| : I \text{ is an independent set of } G\}.$$

Although the maximum independent set problem is NP-hard, the problem of finding a critical independent set is polynomially solvable [9, 204]. Therefore, an interesting question to investigate is, how useful the knowledge about the critical independent

sets of a graph can be for solving the maximum independent set problem in this graph. We were able to prove some related results.

7.2.1 Results

Lemma 7.1. *If I_c is a critical independent set and a maximal independent set at the same time, then I_c is a maximum independent set.*

Proof. Since I_c is a maximal independent set, we have $N(I_c) = V \setminus I_c$. Assume that there exists an independent set I with cardinality $|I| > |I_c|$. Then

$$|I| - |N(I)| \geq |I| - |V \setminus I| > |I_c| - |V \setminus I_c|,$$

which contradicts to the fact that I_c is a critical independent set. □

Corollary 7.1. *If I_c is a critical independent set and a maximal independent set at the same time, then $\alpha(G) \geq n/2$.*

Proof. Follows from nonnegativity of $|I_c| - |N(I_c)|$. □

Theorem 7.2. *If I_c is a critical independent set, then there exists a maximum independent set I, such that $I_c \subseteq I$.*

Proof. Let J be a maximum independent set in G, and I_c be a critical independent set. Put

$$I = (J \cup I_c) \setminus N(I_c).$$

Then I is an independent set, and $I_c \subseteq I$. To prove that I is a maximum independent set, it is enough to show that $|I| \geq |J|$. Assume that $|I| < |J|$, then

$$|N(I_c) \cap J| > |I_c \setminus J|.$$

Using the last inequality and the inequality

$$|N(I_c)| \geq |N(I_c) \cap J| + |N(I_c \cap J)|,$$

we have

$$|I_c| - |N(I_c)| = |I_c \setminus J| + |I_c \cap J| - |N(I_c)| <$$

$$< |N(I_c) \cap J| + |I_c \cap J| - |N(I_c) \cap J| - |N(I_c \cap J)| =$$

$$= |I_c \cap J| - |N(I_c \cap J)|.$$

We obtain a contradiction with the fact that I_c is a critical independent set. The statement is proved. □

Theorem 7.2 states that a nonempty critical set is always a part of a maximum independent set, therefore it can be used in order to reduce the number of vertices to be analyzed when solving the maximum independent set problem. The obtained results suggest that the idea presented in this section deserves a more thorough study. For example, it would be interesting to investigate in what types of graphs the maximum independent set can be found by means of finding a critical set.

7.3 Applications

A large part of this dissertation is devoted to various applications of the considered combinatorial optimization problems. This can be seen as a reflection of the current tendency towards interdisciplinary research in the field of optimization and science overall.

In Chapter 4 of this work, novel techniques for analyzing the structure of financial markets based on their network representation were proposed and verified using massive data sets generated by the U.S. stock markets. The network representation of a market is based on cross-correlations of price fluctuations of the financial instruments, and provides a valuable tool for classifying the instruments. We investigated stability and dynamics of structural properties (such as power-law degree distribution) of the market graph constructed over changing time periods. The methodology for analyzing markets structure presented in Chapter 4 can be used to analyze another type of the market graph based on the data of the liquidity of

different instruments, instead of considering the returns. It would be interesting to study the properties of this graph and compare it with the market graph considered in this work. We also plan to investigate other extensions of this novel methodology.

In Chapter 5, new exact values and estimates of size of the largest error correcting codes were computed using optimization in specially constructed graphs. Error correcting codes lie in the heart of digital technology, making cell phones, compact disk players and modems possible. They are also of a special significance due to increasing importance of reliability issues in internet transmissions. There are still many large-scale problems in this area that need to be solved. Exact solutions and estimates for the size of largest error-correcting codes of larger length can be computed by designing and applying even more efficient, possibly parallel, algorithms for solving maximum independent set problem to optimality.

In Chapter 6, efficient approximate algorithms for construction of a virtual backbone in ad hoc wireless networks by means of solving the minimum connected dominating set problem in unit-disk graphs were developed and tested. Despite excellent performance of the proposed approach, one of its disadvantages is that a feasible solution is constructed only when the algorithm terminates. This drawback can be overcome by developing new algorithms which would maintain a feasible solution at any stage of their execution. Another important problem to investigate is maintaining the connected dominating set in a mobility environment.

In future work, we would also like to extend our work to applications in biomedical engineering. Some of such possibilities were briefly discussed in Section 1.6.

REFERENCES

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons Incorporated, Chichester, UK, 1989.

[2] J. Abello, A. Buchsbaum, and J. Westbrook. A functional approach to external graph algorithms. In *Proceedings of the European Symposium on Algorithms*, volume 1461 of *Lecture Notes in Computer Science*, pages 332–343. Springer-Verlag, Berlin, 1998.

[3] J. Abello, S. Butenko, P. Pardalos, and M. Resende. Finding independent sets in a graph using continuous multivariable polynomial formulations. *Journal of Global Optimization*, 21:111–137, 2001.

[4] J. Abello, P. M. Pardalos, and M. G. C. Resende. On maximum clique problems in very large graphs. In [6], pages 119–130.

[5] J. Abello, P.M. Pardalos, and M.G.C. Resende, editors. *Handbook of Massive Data Sets*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[6] J. Abello and J. S. Vitter, editors. *External Memory Algorithms*, volume 50 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, RI, 1999.

[7] L. Adamic. The small world Web. In *Proceedings of ECDL'99*, volume 1696 of *Lecture Notes in Computer Science*, pages 443–452. Springer-Verlag, Berlin, 1999.

[8] L. Adamic and B. Huberman. Power-law distribution of the World Wide Web. *Science*, 287:2115a, 2000.

[9] A. A. Ageev. On finding critical independent and vertex sets. *SIAM J. Discrete Mathematics*, 7:293–295, 1994.

[10] W. Aiello, F. Chung, and L. Lu. Random evolution in massive graphs. In [5], pages 97–122.

[11] W. Aiello, F. Chung, and L. Lu. A random graph model for power law graphs. *Experimental Math.*, 10:53–66, 2001.

[12] R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the World-Wide Web. *Nature*, 401:130–131, 1999.

[13] N. Alon and M. Krivelevich. The concentration of the chromatic number of random graphs. *Combinatorica*, 17:303–313, 1997.

[14] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Distributed heuristics for connected dominating sets in wireless ad hoc networks. *Journal of Communications and Networks*, 4:22–29, 2002.

[15] L. Amaral, A. Scala, M. Barthélémy, and H. Stanley. Classes of small-world networks. *Proc. of National Academy of Sciences USA*, 97:11149–11152, 2000.

[16] A. T. Amin and S. L. Hakimi. Upper bounds on the order of a clique of a graph. *SIAM J. Appl. Math.*, 22:569–573, 1972.

[17] A. D. Amis and R. Prakash. Load-balancing clusters in wireless ad hoc networks. In *Proceedings of Conference on Application-Specific Systems and Software Engineering Technology (ASSET 2000)*, pages 25–32, Richardson, Texas, 2000.

[18] L. Arge. The buffer tree: A new technique for optimal I/O algorithms. In *Proceedings of the Workshop on Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 334–345. Springer-Verlag, Berlin, 1995.

[19] L. Arge, G. S. Brodal, and L. Toma. On external memory MST, SSSP and multi-way planar graph separation. In *Proceedings of the Seventh Scandinavian Workshop on Algorithmic Theory*, volume 1851 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2000.

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

[21] S. Arora and S. Safra. Approximating clique is NP-complete. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 2–13, Piscataway, NJ, 1992.

[22] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw-Hill Publishing Company, Maidenhead, England, 1998.

[23] G. Avondo-Bodeno. *Economic Applications of the Theory of Graphs*. Gordon and Breach Science Publishers, New York, 1962.

[24] B. Awerbuch. Optimal distributed algorithm for minimum weight spanning tree, counting, leader election and related problems. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 230–240, New York, 1987.

[25] E. Balas and C.S. Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal of Computing*, 15:1054–1068, 1986.

[26] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[27] A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the World-Wide Web. *Physica A*, 281:69–77, 2000.

[28] R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. *Algorithmica*, 29:610–637, 2001.

[29] A. R. Bednarek and O. E. Taulbee. On maximal chains. *Roum. Math. Pres et Appl.*, 11:23–25, 1966.

[30] C. Berge. *The Theory of Graphs and its Applications*. Methuen, London, 1962.

[31] C. Berge. *Graphs and Hypergraphs*. North-Holland Mathematical Library, Amsterdam, 1976.

[32] V. Boginski, S. Butenko, and P. M. Pardalos. On structural properties of the market graph. In A. Nagurney, editor, *Innovation in Financial and Economic Networks*. Edward Elgar Publishers, London. In press.

[33] V. Boginski, S. Butenko, and P. M. Pardalos. Modeling and optimization in massive graphs. In P. M. Pardalos and H. Wolkowicz, editors, *Novel Approaches to Hard Discrete Optimization*, pages 17–39. American Mathematical Society, Providence, RI, 2003.

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

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

[36] B. Bollobás. The chromatic number of random graphs. *Combinatorica*, 8:49–56, 1988.

[37] B. Bollobás and P. Erdös. Cliques in random graphs. *Math. Proc. Camb. Phil. Soc.*, 80:419–427, 1976.

[38] 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. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[39] I. M. Bomze, M. Budinich, M. Pelillo, and C. Rossi. A new "annealed" heuristic for the maximum clique problem. In P. M. Pardalos, editor, *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*, pages 78–96. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[40] I. M. Bomze, M. Pelillo, and R. Giacomini. Evolutionary approach to the maximum clique problem: empirical evidence on a larger scale. In I. M. Bomze, T. Csendes, R. Horst, and P. M. Pardalos, editors, *Developments of Global Optimization*, pages 95–108. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[41] R. E. Bonner. On some clustering techniques. *IBM J. Res. Develop.*, 8:22–32, 1964.

[42] S. Brin and L. Page. The anatomy of a large scale hypertextual Web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, 1998.

[43] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the Web. *Computer Networks*, 33:309–320, 2000.

[44] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tompkins, and J. Wiener. The Bow-Tie Web. In *Proceedings of the Ninth International World Wide Web Conference*, Brisbane, Australia, 2000.

[45] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques on an undirected graph. *Communications of ACM*, 16:575–577, 1973.

[46] L. Brotcorne, G. Laporte, and F. Semet. Fast heuristic for large scale covering-location problems. *Computers & Operations Research*, 29:651–665, 2002.

[47] A. Brouwer, J. Shearer, N. Sloane, and W. Smith. A new table of constant weight codes. *IEEE Trans. Inform. Theory*, 36:1334–1380, 1990.

[48] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *Proceedings of IEEE INFOCOM*, Piscataway, NJ, 2002.

[49] A. L. Buchsbaum, M. Goldwasser, S. Venkatasubramanian, and J. R. Westbrook. On external memory graph traversal. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 859–860, San Francisco, CA, 2000.

[50] T. N. Bui and P. H. Eppley. A hybrid genetic algorithm for the maximum clique problem. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 478–484, Pittsburgh, PA, 1995.

[51] S. Burer, R. D. C. Monteiro, and Y. Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM J. on Optimization*, 12:503–521, 2001.

[52] S. Burer, R. D. C. Monteiro, and Y. Zhang. Maximum stable set formulations and heuristics based on continuous optimization. *Mathematical Programming*, 94:137–166, 2002.

[53] S. Busygin. QUALEX 2.0 Release Notes. http://www.busygin.dp.ua/npc.html, 2001. Accessed March 2002.

[54] S. Busygin, S. Butenko, and P. M. Pardalos. A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. *Journal of Combinatorial Optimization*, 6:287–297, 2002.

[55] S. Butenko, P. M. Pardalos, I. V. Sergienko, V. Shylo, and P. Stetsyuk. Estimating the size of correcting codes using extremal graph problems. In C. Pearce, editor, *Optimization: Structure and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands. In press.

[56] S. Butenko, P. M. Pardalos, I. V. Sergienko, V. Shylo, and P. Stetsyuk. Finding maximum independent sets in graphs arising from coding theory. In *Proceedings of the Seventeenth ACM Symposium on Applied Computing*, pages 542–546, Madrid, Spain, 2002.

[57] Y. Caro and Zs. Tuza. Improved lover bounds on k-independence. *J. Graph Theory*, 15:99–107, 1991.

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

[59] S. Chakrabarti, B. Dom, D. Gibson, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation. In *Proceedings of ACM SIGIR Workshop on Hypertext Information Retrieval on the Web*, Melbourne, Australia, 1998.

[60] W. Cheswick and H. Burch. Internet Mapping Project. http://www.cs.bell-labs.com/who/ches/map/. Accessed March 2003.

[61] Y.-J. Chiang, M. T. Goodrich, E. F. Grove, R. Tamassia, D. E. Vengroff, and J. S. Vitter. External-memory graph algorithms. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 139–149, San Francisco, CA, 1995.

[62] F. Chung and L. Lu. The diameter of random sparse graphs. *Advances in Applied Math.*, 26:257–279, 2001.

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

[64] S. D. Constantin and T. R. N. Rao. On the theory of binary asymmetric error correcting codes. *Information and Control*, 40:20–36, 1979.

[65] C. Cooper and A. Frieze. A general model of undirected Web graphs. In *Algorithms - ESA 2001, Ninth Annual European Symposium*, pages 500–511, Aarhus, Denmark, 2001.

[66] C. Cooper and A. Frieze. The size of the largest strongly connected component of a random graph with a given degree sequence. http://www.math.cmu.edu/~af1p/papers.html, 2002. Accessed February 2002.

[67] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *Proceedings of ICC 1997*, pages 376–380, Montreal, Canada, 1997.

[68] Dash Optimization. Xpress. http://www.dashoptimization.com. Accessed July 2003.

[69] M.S. Daskin. *Network and Discrete Location*. John Wiley & Sons Incorporated, New York, 1995.

[70] P. L. de Angelis, I. M. Bomze, and G. Toraldo. Ellipsoidal approach to box-constrained quadratic problems. *J. Global Optimization*. In press.

[71] P. Delsarte and P. Piret. Bounds and constructions for binary asymmetric error correcting codes. *IEEE Trans. Inform. Theory*, IT-27:125–128, 1981.

[72] N. Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[73] M. Deza and M. Laurent. *Geometry of Cuts and Metrics*. Springer-Verlag, New York, 1997.

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

[75] I. I. Dikin. Iterative solution of linear and quadratic programming problems. *Doklady of Soviet Academy of Sciences*, 174:747–748 (in Russian), 1967.

[76] I. I. Dikin and Zorkal'tsev. *Iterative Solution of Problems of Mathematical Programming (Interior Point Algorithms)*. Nauka, Novosibirsk, 1980.

[77] J.J. Dongarra, C.B. Moler, J.R.Bunch, and G.W.Stewart. Linpack users' guide. http://www.netlib.org/linpack/index.html, 1979. Accessed January 2002.

[78] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[79] P. Erdös and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.

[80] T. Etzion. New lower bounds for asymmetric and undirectional codes. *IEEE Trans. Inform. Theory*, 37:1696–1704, 1991.

[81] T. Etzion and P. R. J. Östergård. Greedy and heuristic algorithms for codes and colorings. *IEEE Trans. Inform. Theory*, 44:382–388, 1998.

[82] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Proceedings of the ACM-SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 251–262, Cambridge, MA, 1999.

[83] U. Feige, S. Goldwasser, L. Lovász, S Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43:268–292, 1996.

[84] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57:187–199, 1998.

[85] T. A. Feo and M. G. C. Resende. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878, 1994.

[86] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[87] P. Festa, P. M. Pardalos, M. G. C. Resende, and C. C. Ribeiro. Randomized heuristics for the MAX-CUT problem. *Optimization Methods and Software*, 7:1033–1058, 2002.

[88] C. Floudas and P. M. Pardalos, editors. *Encyclopedia of Optimization* (in 6 volumes). Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[89] G. E. Forsythe and G. H. Golub. On the stationary values of a second degree polynomial on the unit sphere. *SIAM J. on Applied Mathematics*, 13:1050–1068, 1965.

[90] C. Friden, A. Hertz, and D. de Werra. Stabulus: A technique for finding stable sets in large graphs with tabu search. *Computing*, 42:35–44, 1989.

[91] A. Frieze. On the independence number of random graphs. *Disctere Mathematics*, 81:171–175, 1990.

[92] T. Gallai. Über extreme Punkt- und Kantenmengen. *Ann. Univ. Sci. Budapest. Eötvös Sect. Math.*, 2:133–138, 1959.

[93] E. Gardiner, P. Artymiuk, and P. Willett. Clique-detection algorithms for matching three-dimensional molecular structures. *Journal of Molecular Graphics and Modelling*, 15:245–253, 1997.

[94] E. Gardiner, P. Willett, and P. Artymiuk. Graph-theoretic techniques for macromolecular docking. *J. Chem. Inf. Comput.*, 40:273–279, 2000.

[95] M.R. Garey and D.S. Johnson. The complexity of near-optimal coloring. *Journal of the ACM*, 23:43–49, 1976.

[96] 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.

[97] L. E. Gibbons, D. W. Hearn, and P. M. Pardalos. A continuous based heuristic for the maximum clique problem. In [136], pages 103–124.

[98] L. E. Gibbons, D. W. Hearn, P. M. Pardalos, and M. V. Ramana. Continuous characterizations of the maximum clique problem. *Math. Oper. Res.*, 22:754–768, 1997.

[99] F. Glover. Tabu search - part I. *ORSA J. Comput.*, 1:190–260, 1989.

[100] F. Glover. Tabu search - part II. *ORSA J. Comput.*, 2:4–32, 1990.

[101] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[102] F. Glover, M. Laguna, and R. Marti. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

[103] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42:1115–1145, 1995.

[104] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[105] O. Goldreich. Property testing in massive graphs. In [5], pages 123–147.

[106] G. R. Grimmett and C. J. H. McDiarmid. On coloring random graphs. *Mathematical Proceedings of Cambridge Phil. Society*, 77:313–324, 1975.

[107] T. Grossman. Applying the INN model to the max clique problem. In [136], pages 125–146.

[108] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 2nd edition, 1993.

[109] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, 1998.

[110] W. Hager. Minimizing a quadratic over a sphere. *SIAM J. on Optimization*, 12:188–208, 2001.

[111] W. Hager and Y. Krylyuk. Graph partitioning and continuous quadratic programming. *SIAM Journal on Discrete Mathematics*, 12:500–523, 1999.

[112] J. Harant. Some news about the independence number of a graph. *Discussiones Mathematicae Graph Theory*, 20:71–79, 2000.

[113] J. Harant, A. Pruchnewski, and M. Voigt. On dominating sets and independent sets of graphs. *Combinatorics, Probability and Computing*, 8:547–553, 1999.

[114] F. Harary and I. C. Ross. A procedure for clique detection using the group matrix. *Sociometry*, 20:205–215, 1957.

[115] E. Harley, A. Bonner, and N. Goodman. Uniform integration of genome mapping data using intersection graphs. *Bioinformatics*, 17:487–494, 2001.

[116] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[117] B. Hayes. Graph theory in practice. *American Scientist*, 88:9–13 (Part I), 104–109 (Part II), 2000.

[118] M. Hifi. A genetic algorithm - based heuristic for solving the weighted maximum independent set and some equivalent problems. *J. Oper. Res. Soc.*, 48:612–622, 1997.

[119] J. H. Holland. *Adaption in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[120] S. Homer and M. Peinado. Experiments with polynomial-time clique approximation algorithms on very large graphs. In [136], pages 147–167.

[121] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[122] J. J. Hopfield and D. W. Tank. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[123] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2 edition, 2000.

[124] B. Huberman and L. Adamic. Growth dynamics of the World-Wide Web. *Nature*, 401:131, 1999.

[125] ILOG CPLEX. http://www.ilog.com/products/cplex/. Accessed July 2003.

[126] Internet Domain Survey: Number of Internet Hosts. http://www.isc.org/ds/host-count-history.html. Accessed February 2002.

[127] A. Jagota. Approximating maximum clique with a Hopfield network. *IEEE Trans. Neural Networks*, 6:724–735, 1995.

[128] A. Jagota, L. Sanchis, and R. Ganesan. Approximately solving maximum clique using neural networks and related heuristics. In [136], pages 169–204.

[129] S. Janson, T. Łuczak, and A. Ruciński. *Random Graphs*. John Wiley & Sons Incorporated, New York, 2000.

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

[131] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.

[132] M. Joa-Ng and I.-T. Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17:1415–1425, 1999.

[133] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad hoc networks. In *Proceedings of IEEE MOBICOM*, pages 195–206, Seattle, WA, 1999.

[134] D. B. Johnson, D. A. Maltz, Y.-C. Hu, and J. G. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks. http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt, 2001. Accessed December 2001.

[135] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9:256–278, 1974.

[136] D. S. Johnson and M. A. Trick, editors. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series*. American Mathematical Society, Providence, RI, 1996.

[137] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, pages 373–395, 1984.

[138] L. G. Khachian. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:1093–1096, 1979.

[139] S. Kirkpatrick, C. D. Gellat Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[140] V. Klee and D. Larman. Diameters of random graphs. *Canadian Journal of Mathematics*, 33:618–640, 1981.

[141] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, 1998.

[142] D. E. Knuth. The sandwich theorem. *Electronic Journal of Combinatorics*, 1:A1, 1994. Accessed June 2003.

[143] V. F. Kolchin. *Random Graphs*. Cambridge University Press, Cambridge, UK, 1999.

[144] R. Kopf and G. Ruhe. A computational study of the weighted independent set problem for general graphs. *Found. Control Engin.*, 12:167–180, 1987.

[145] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. The Web as a graph. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–10, Dallas, TX, 2000.

[146] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for cyber communities. In *Proceedings of the Eighth International World Wide Web Conference*, Brisbane, Australia, 1999.

[147] V. Kumar and E. Schwabe. Improved algorithms and data structures for solving graph problems in external memory. In *Proceedings of the Eighth IEEE Symposium on Parallel and Distributed Processing*, pages 169–176, New Orleans, LA, 1996.

[148] S. Lawrence and C. L. Giles. Accessibility of information on the Web. *Nature*, 400:107–109, 1999.

[149] L. Lovász. On the shannon capacity of a graph. *IEEE Trans. Inform. Theory*, 25:1–7, 1979.

[150] T. Łuczak. A note on the sharp concentration of the chromatic number of random graphs. *Combinatorica*, 11:295–297, 1991.

[151] T. Łuczak. Random trees and random graphs. *Random Structures and Algorithms*, 13:485–500, 1998.

[152] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994.

[153] S. Lyle and M. Szularz. Local minima of the trust region problem. *Journal of Optimization Theory and Applications*, 80:117–134, 1994.

[154] D. Mackenzie. Graph theory uncovers the roots of perfection. *Science*, 297:38, 2002.

[155] C. Mannino and E. Stefanutti. An augmentation algorithm for the maximum weighted stable set problem. *Computational Optimization and Applications*, 14:367–381, 1999.

[156] R. N. Mantegna and H. E. Stanley. *An Introduction to Econophysics: Correlations and Complexity in Finance*. Cambridge University Press, Cambridge, UK, 2000.

[157] E. Marchiori. Genetic, iterated and multistart local search for the maximum clique problem. In *Applications of Evolutionary Computing*, volume 2279 of *Lecture Notes in Computer Science*, pages 112–121. Springer-Verlag, Berlin, 2002.

[158] P. M. Marcus. Derivation of maximal compatibles using Boolean algebra. *IBM J. Res. Develop.*, 8:537–538, 1964.

[159] D. Matula. On the complete subgraph of a random graph. In R. Bose and T. Dowling, editors, *Proceedings of Second Chapel Hill Conference on Combinatorial Mathematics and Its Applications*, pages 356–369. University of North Carolina, Chapel Hill, 1970.

[160] A. Mendelzon, G. Mihaila, and T. Milo. Querying the World Wide Web. *Journal of Digital Libraries*, 1:68–88, 1997.

[161] A. Mendelzon and P. Wood. Finding regular simple paths in graph databases. *SIAM J. Computing*, 24:1235–1258, 1995.

[162] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6:161–180, 1995.

[163] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23–28, 1965.

[164] J. J. Moré and D. S. Sorensen. Computing a trust region step. *SIAM J. Sci. Statist. Comput.*, 4:553–572, 1983.

[165] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, 17:533–540, 1965.

[166] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of MOBICOM*, pages 151–162, Seattle, WA, 1999.

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

[168] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., Mineola, NY, 1988.

[169] P. M. Pardalos and M. G. C. Resende, editors. *Handbook of Applied Optimization*. Oxford University Press, New York, 2002.

[170] P. M. Pardalos and G. P. Rodgers. A branch and bound algorithm for the maximum clique problem. *Computers Ops Res.*, 19:363–375, 1992.

[171] P. M. Pardalos and J. Xue. The maximum clique problem. *J. Global Optim.*, 4:301–328, 1992.

[172] R. Pastor-Satorras, A. Vazquez, and A. Vespignani. Dynamical and correlation properties of the Internet. *Physical Review Letters*, 87:258701, 2001.

[173] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86:3200–3203, 2001.

[174] M. C. Paull and S. H. Unger. Minimizing the number of states in incompletely specified sequential switching functions. *IRE Transactions Electr. Comput.*, EC-8:356–367, 1959.

[175] G. Pei, M. Gerla, and T.-W. Chen. Fisheye state routing: a routing scheme for ad hoc wireless networks. In *IEEE International Conference on Communications, ICC 2000*, volume 1, pages 70–74, New Orleans, LA, 2000.

[176] C. E. Perkins, E. M. Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08.txt. Accessed December 2001.

[177] T. R. N. Rao and A. S. Chawla. Asymmetric error codes for some LSI semiconductor memories. *Proceedings of the Seventh Annual Southeastern Symposium of System Theory*, pages 170–171, 1975.

[178] R. Rizzi. A short proof of matching theorem. *J. Graph Theory*, 33:138–139, 2000.

[179] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7:425–440, 1986.

[180] R. Samudrala and J. Moult. A graph-theoretic algorithm for comparative modeling of protein structure. *J. Mol. Biol.*, 279:287–302, 1988.

[181] I. V. Sergienko, V. P. Shylo, and P. I. Stetsyuk. Approximate algorithm for solving the maximum independent set problem. *Computer Mathematics, V.M. Glushkov Institute of Cybernetics NAS of Ukraine, Kiev.*, pages 4–20 (in Russian), 2000.

[182] E. Shamir and J. Spencer. Sharp concentration of the chromatic number on random graphs $G_{n,p}$. *Combinatorica*, 7:124–129, 1987.

[183] N. Z. Shor. Dual quadratic estimates in polynomial and Boolean programming. *Ann. Oper. Res.*, 25:163–168, 1990.

[184] V. Shylo. New lower bounds of the size of error-correcting codes for the Z-channel. *Cybernetics and Systems Analysis*, 38:13–16, 2002.

[185] V. Shylo and D. Boyarchuk. An algorithm for construction of covering by independent sets. *Computer Mathematics (Publication of Glushkov Institute of Cybernetics, Kiev, Ukraine)*, pages 151–157, 2001.

[186] P. Sinha, R. Sivakumar, and V. Bharghavan. Enhancing ad hoc routing with dynamic virtual infrastructures. volume 3, pages 1763–1772, Piscataway, NJ, 2001.

[187] N. Sloane. Challenge Problems: Independent Sets in Graphs. http://www.research.att.com/~njas/doc/graphs.html, 2000. Accessed July 2003.

[188] N. J. A. Sloane. On single-deletion-correcting codes. In K. T. Arasu and A. Seress, editors, *Codes and Designs*, volume 10 of *Ohio State University Mathematical Research Institute Publications*, pages 273–291. Walter de Gruyter, Berlin, 2002.

[189] P. Soriano and M. Gendreau. Tabu search algorithms for the maximum clique problem. In [136], pages 221–242.

[190] V. T. Sós and E. G. Straus. Extremal of functions on graphs with applications to graphs and hypergraphs. *J. Combin. Theory B*, 32:246–257, 1982.

[191] R. E. Tarjan and A. E. Trojanowski. Finding a maximum independent set. *SIAM Journal of Computing*, 6:537–546, 1977.

[192] E. Tomita, A. Tanaka, and H. Takahashi. The worst-time complexity for finding all the cliques. Technical Report UEC-TR-C5, University of Electro-Communications, Tokyo, Japan, 1988.

[193] L.E. Trotter Jr. Solution characteristics and algorithms for the vertex packing problem. Technical Report 168, Dept. of Operations Research, Cornell University, Ithaca, NY, 1973.

[194] J. D. Ullman and M. Yannakakis. The input/output complexity of transitive closure. *Annals of Mathematics and Artificial Intelligence*, 3:331–360, 1991.

[195] C. L. M. van Pul and T. Etzion. New lower bounds for constant weight codes. *IEEE Trans. Inform. Theory*, 35:1324–1329, 1989.

[196] R. R. Varshamov. A class of codes for asymmetric channels and a problem from the additive theory of numbers. *IEEE Trans. Inform. Theory*, IT-19:92–95, 1973.

[197] J. S. Vitter. External memory algorithms and data structures: dealing with MASSIVE DATA. *ACM Computing Surveys*, 33:209–271, 2001.

[198] D. Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, Princeton, NJ, 1999.

[199] D. Watts and S. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.

[200] V. K. Wei. A lower bound on the stability number of a simple graph. Technical Report TM 81-11217-9, Bell Laboratories, Murray Hill, NJ, 1981.

[201] H. S. Wilf. The eigenvalues of a graph and its chromatic number. *J. London Math. Soc.*, 42:330–332, 1967.

[202] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the Third International Workshop on Discrete Algothrithms and Methods for MOBILE Computing and Communications*, pages 7–14, Seattle, WA, 1999.

[203] Y. Ye. A new complexity result on minimization of a quadratic function with a sphere constraint. In C. Floudas and P. Pardalos, editors, *Recent Advances in Global Optimization*, pages 19–31. Princeton University Press, 1992.

[204] C.-Q. Zhang. Finding critical independent sets and critical vertex subsets are polynomial problems. *SIAM J. Discrete Mathematics*, 3:431–438, 1990.

[205] X.-S. Zhang. *Neural Networks in Optimization.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

BIOGRAPHICAL SKETCH

Sergiy Butenko was born on June 19, 1977, in Crimea, Ukraine. In 1994, he completed his high school education in the Ukrainian Physical-Mathematical Lyceum of Kiev University in Kiev, Ukraine. He received his bachelor's and master's degrees in mathematics from Kiev University in Kiev, Ukraine, in 1998 and 1999, respectively. In August 1999, he began his doctoral studies in the Industrial and Systems Engineering Department at the University of Florida. He earned his Ph.D. in August 2003.