5.2 INDEPENDENCE AND CLIQUES

745min 4–26–2003

Gregory Gutin, Royal Holloway, University of London

INTRODUCTION

Finding maximum cliques and maximum independent sets are among the most applicable problems in graph theory. We give an overview of both algorithmic and theoretical results on these problems.

5.2.1 Basic Definitions and Applications

In this section, all graphs are *simple*, i.e., they do not have self-loops or multi-edges.

Some Combinatorial Optimization Problems

DEFINITIONS

D1: For a graph G, a set S of vertices is an ***independent set*** if no two vertices in S are adjacent.

D2: The number of vertices in a maximum-size independent set of G is called the ***independence number*** of G and is denoted $ind(G)$.

D3: A ***clique*** in a graph G is a maximal set of mutually adjacent vertices of G. The ***clique number***, denoted $\omega(G)$, is the number of a vertices in a largest clique of G.

D4: A ***vertex cover*** in a graph G is a set S of vertices such that at least one endpoint of every edge of G is in S.

D5: A ***matching*** in a graph G is a set of mutually non-adjacent edges of G.

REMARKS

R1: The ***maximum-clique problem*** (determining $\omega(G)$ for a given graph G) and the ***maximum-independent-set problem*** (determining $ind(G)$) are the main problems considered in this section. Observe that $\omega(G) = ind(\overline{G})$ for any graph G, where $\overline{G}$ denotes the edge-complement graph.

R2: Also considered in this section is the ***minimum-vertex-cover problem*** of finding a vertex cover of minimum cardinality.

R3: Sometimes, we will consider graphs with non-negative weights on their vertices.

EXAMPLE

E1: It is easy to verify for the Peterson graph G shown in Figure 5.2.1 that $\omega(G) = 2$, $ind(G) = 4$, and a minimum vertex cover has size 5.

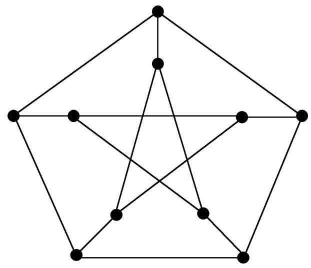

Figure 5.2.1 **The Petersen graph.**

DEFINITIONS

D6: A graph G is ***vertex-weighted*** if every vertex x is assigned a non-negative weight $w(x)$; the graph is denoted (G, w).

D7: The ***weight of a vertex set*** S in a vertex-weighted graph is the sum of the weights of the vertices in S.

NOTATION: The weight of a maximum-weight independent set in G is denoted $ind(G, w)$. The weight of a maximum-weight clique in G is denoted $\omega(G, w)$.

REMARK

R4: The ***maximum-weight-independent-set problem*** (determining $ind(G, w)$) generalizes the unweighted one: assign weight 1 to every vertex of a graph. A similar remark holds for the ***maximum-weight-clique problem***.

R5: The first three of the following facts are immediate consequences of the definitions.

FACTS

F1: For every vertex-weighted graph (G, w), $ind(G, w) = \omega(\overline{G}, w)$. Thus, an obvious duality between the two problems.

F2: A set S of vertices in a graph $G = (V, E)$ is a vertex cover if and only if $V - S$ is an independent set.

F3: The size of a maximum matching in a graph G is less or equal to the size of a minimum cover of G.

F4: [Ko31,Eg31] If G is a bipartite graph, then the maximum size of a matching in G equals the minimum cardinality of a vertex cover of G.

Applications Involving Hamming Distance

There are numerous and varied applications of the combinatorial optimization problems introduced above. Perhaps among the most studied are those from coding theory.

DEFINITIONS

D8: The ***Hamming distance*** between a pair $u = (u_1, \ldots, u_n)$, $v = (v_1, \ldots, v_n)$ of binary vectors is the number of indices i for which $u_i \neq v_i$.

D9: The vertex set of the ***Hamming graph*** $H(n, d)$ consists of all binary vectors with n coordinates. A pair u, v of vertices in $H(n, d)$ are adjacent if the Hamming distance between them is at least d.

REMARKS

R6: The Hamming graph is of interest for error-correcting codes: A binary code consisting of a set of binary vectors, any pair of which have Hamming distance at least d can correct $\lfloor (d-1)/2 \rfloor$ errors. [MaSl79]

R7: A natural question arises: How many vectors with n coordinates can be in a code in which any two vectors are at least Hamming distance d apart? It is obvious from the definitions that this number equals the order of a maximum clique in $H(n, d)$.

R8: For further discussions and results of this application, see, for example, [BoBuPaPe99] and [Os01]. Other applications include fault diagnosis [BePe90,HaPaVa93], machine learning [HoSk89,HaJa90], and detecting embedded network structures in linear programs [GuGuMiMa00,GuGuMiZv].

5.2.2 Integer Programming Formulations

The simplest formulation of the maximum-weight-clique problem is based on the edges of the input graph.

Edge-based formulation: Let $G = (V, E)$ be a vertex-weighted graph with $V = \{v_1 \ldots, v_n\}$ and weights $w_i = w(v_i)$. The maximum weight of a clique can be found by solving the following integer program:

$$\begin{aligned} \max w &= \sum_{i=1}^{n} w_i x_i \\ \text{subject to } \; & x_i + x_j \leq 1 \quad \forall \{v_i, v_j\} \notin E \\ & x_i = 0 \text{ or } 1, \quad i = 1, \ldots, n \end{aligned}$$

FACTS

F5: In the edge-based formulation, every feasible solution x corresponds to the vertex set S of a complete subgraph of G as follows: $x_i = 1$ if and only if $v_i \in S$.

F6: [NeTr74,NeTr75] Let x be an optimum $(0, \frac{1}{2}, 1)$-valued solution to the linear relaxation of the edge formulation, and let $J = \{j : \; x_j = 1\}$. Then there exists an optimal solution x^* to the edge-based formulation such that $x_j^* = 1$ for every $j \in J$.

REMARKS

R9: Unfortunately, the result in Fact 6 above appears to be of relatively minor computational value since optimal solutions of the linear relaxation of the edge-based formulation normally have only a small number of integer components and the gap between optimal solutions of the edge-based formulation and its linear relaxation is usually too large. [BoBuPaPe99]

R10: The obvious alteration transforms the edge-based formulation to the corresponding formulation of the maximum-weight-independent-set problem, which is clearly equivalent to the following nonlinear optimization problem first studied in [Sh90].

Shor formulation:

$$\begin{aligned} \min w &= \sum_{i=1}^{n} w_i x_i \\ \text{subject to} \quad & x_i x_j = 0 \quad \forall \{v_i, v_j\} \in E \\ & x_i^2 - x_i = 0 \quad i = 1, \ldots, n \end{aligned}$$

COMPUTATIONAL NOTE: Shor [Sh90] reported very good computational results using his formulation.

Two More Formulations of the Maximum-Clique Problem

While the formulations above are relatively straightforward, the following ones initiated by Motzkin and Straus [MoSt65] are less obvious.

DEFINITIONS

D10: The ***standard simplex*** Δ in R^n is defined as follows:

$$\Delta = \{x \in R^n : \; x_i \geq 0, \; i = 1, \ldots, n, \; \sum_{i=1}^{n} x_i = 1\}$$

NOTATION: For a graph G with adjacency matrix A_G and a vector $x \in R^n$, let $g(x) = x^T A_G x$.

D11: Let $G = (V, E)$ be a graph with vertices $v_1, \ldots, v_n$ and let $S \subseteq V$ be arbitrary. The ***characteristic vector*** $x^S \in R^n$ is defined as follows: $x^S = (x_1^S, x_2^S, \ldots, x_n^S)$, where $x_i^S = \frac{1}{|S|}$ if $v_i \in S$ and $x_i^S = 0$, otherwise.

Motzkin-Straus Theorem [MoSt65]: Let G be a graph and let $x^* = argmax\{g(x) : \; x \in \Delta\}$ (i.e., the $x \in \Delta$ for which $g(x)$ is maximum). Then

$$\omega(G) = \frac{1}{1 - g(x^*)} \geq \frac{1}{1 - g(x)} \; \forall x \in \Delta.$$

Moreover, a subset S of vertices of G is a maximum clique if and only if

$$x^S = argmax\{g(x) : \; x \in \Delta\}$$

REMARK

R11: One drawback of the Motzkin-Straus formulation $\{g(x) : x \in \Delta\}$ is the fact that some solutions of this optimization problem are not characteristic vectors [PaPh90, PeJa95]. Thus, the following variation of the Motzkin-Straus formulation due to Bomze [Bo97] is of interest.

NOTATION: For a graph G with adjacency matrix A_G, let $f(x) = x^T A_G x + (x^T x)/2$.

Bomze Theorem [Bo97]: Let S be a subset of vertices of a graph G. Then

(a) S is a maximum clique in G if and only if $x^S = argmax\{f(x) : x \in \Delta\}$.

(b) S is a clique in G if and only if x^S is a local maximizer of $\{f(x) : x \in \Delta\}$.

(c) All local maximizers x of $\{f(x) : x \in \Delta\}$ are characteristic vectors.

5.2.3 Complexity and Approximation

The maximum-clique problem is one of the first shown to be NP-hard [Ka72]. Since then many researchers have tried to gain a more precise understanding of the difficulty of the problem (see, e.g., [AuCrGaKaMaPr99, BoBuPaPe99]). One of the strongest indicators of its considerable difficulty is given in Fact 7.

NOTATION: (a) If π is an algorithm for the maximum-independent-set problem, then $\pi(G)$ denotes the independent set produced by π when the input graph is G.

(b) If ψ is an algorithm for the maximum-clique problem, then $\psi(G)$ denotes the clique produced by ψ when the input graph is G.

FACTS

F7: [Ha99] Let ψ be any polynomial-time algorithm for the maximum-clique problem. Unless P = NP, for any $\epsilon \in (0, 1/2]$, there exists an n-vertex graph G such that $\psi(G)$ has fewer than $\omega(G)/n^{1/2-\epsilon}$ vertices.

The following positive result has a nice (and short) proof based on some ideas of Paul Erdős (see [Ha98]).

F8: [BoHa92] There is a polynomial-time algorithm π such that for any n-vertex graph G, $ind(G)/|\pi(G)| = O(n/\log^2 n)$.

This result was recently improved by U. Fiege [Fi].

F9: There is a polynomial-time algorithm π such that for any n-vertex graph G, $ind(G)/|\pi(G)| = O(n(\log \log n)^2/\log^3 n)$.

REMARK

R12: Since the approximation in Fact 9 is still very weak, it is natural to consider the approximation for graphs for which certain parameters are restricted. One such parameter is the maximum degree $\Delta(G)$ of a graph G. For an overview of approximation algorithms whose performance is measured in terms of $\Delta(G)$, see [LaTi01]. Facts 10 and 11 are two of the currently best results.

FACTS

F10: [AlFeWiZu95] Unless P = NP, there exists a constant $\epsilon > 0$ such that there is no polynomial-time algorithm π for which $ind(G)/|\pi(G)| = O(\Delta(G)^\epsilon)$ for every graph

G.

F11: Vishvanathan (see [Ha98]) There is a polynomial-time algorithm π such that for every graph G, $ind(G)/|\pi(G)| = O(\Delta(G) \log\log \Delta(G)/ \log \Delta(G))$.

REMARKS

R13: There is a simple polynomial-time algorithm for the minimum-vertex-cover problem that provides a 2-approximation (i.e., no worse than twice the optimum): find a maximum matching M in a given graph G and output the vertices of M as a vertex cover of G. For slightly better approximation results, see [AuCrGaKaMaPr99].

R14: The last remark and Fact 7 are, in a way, at odds with each other. The maximum-clique and the minimum-vertex-cover problems are dual, in a sense (via the maximum-independent-set problem as noted earlier). Nevertheless, while the former cannot be approximated to any good degree, the latter can be. This 'strange' situation is somewhat resolved by Fact 12. Observe that a feasible solution of the maximum-clique problem is a set of vertices that induces a complete subgraph.

FACT

F12: [GuVaYe] Let ψ be any polynomial-time algorithm for the maximum-clique problem and let $p(n)$ be any polynomial function of n. Unless P = NP, there exists an n-vertex graph G such that $\psi(G)$ has fewer vertices than at least $\frac{p(n)-1}{p(n)}$ of the complete subgraphs of G. The analogous fact holds for the minimum-vertex-cover problem.

5.2.4 Bounds on Independence and Clique Numbers

Every maximum-clique or independent-set heuristic provides an 'algorithmic' lower bound to the corresponding problem. In this subsection, we consider 'analytical' ones that require only certain parameters of the input graph.

Lower Bounds

Caro and Wei obtained the lower bound given in Fact 13, and Alon and Spencer gave an elegant probabilistic proof of this bound [AlSp92]. Recently, Sakai, Togasaki, and Yamazaki generalized Fact 13 to vertex-weighted graphs (Fact 14).

NOTATION: For a vertex v in a graph G, $N(v)$ is the set of vertices adjacent to v and $N[v] = N(v) \cup \{v\}$.

FACTS

F13: [Ca79,We81] Let $G = (V, E)$ be a graph. Then $ind(G) \geq \sum_{v \in V} 1/(deg(v) + 1)$.

F14: [SaToYa03] Every vertex-weighted graph $G = (V, E)$ contains an independence set S of weight at least

$$\max\{\sum_{x \in V} w(x)/(deg(x) + 1), \sum_{x \in V} w(x)^2/[\sum_{y \in N[x]} w(y)]\}$$

Moreover, a set S with the above property can be found in polynomial time.

Selkow [Se94] improved the Caro-Wei bound using additional information.

F15: [Se94] Let $G = (V, E)$ be a graph. Then

$$ind(G) \geq \sum_{v \in V} \frac{1}{deg(v)+1}(1 + \max\{0, \frac{deg(v)}{deg(v)+1} - \sum_{u \in N(v)} \frac{1}{deg(u)+1}\})$$

REMARK

R15: The next few results involve the eigenvalues of the adjacency matrix A_G of G.

DEFINITION

D12: The ***Perron root*** $\lambda_P(G)$ is the largest eigenvalue of A_G.

FACTS

F16: The adjacency matrix of a connected graph G is irreducible, symmetric, and has all non-negative entries; hence, all its eigenvalues are real (see, e.g., [HoJo85]).

F17: [Wi86] For a connected graph G on n vertices, $\omega(G) \geq \frac{n}{n-\lambda_P(G)}$. (This bound was recently improved by Budinich [Bu]).

Upper bounds

FACTS

F18: [Wi67] For a connected graph G, $\omega(G) \leq \lambda_P(G) + 1$. Equality holds if and only if G is complete.

NOTATION: For a connected graph G, let $\Lambda_{-1}(G)$ denote the number of eigenvalues of A_G that do not exceed -1.

F19: [AmHa72] For a connected graph G, $\omega(G) \leq \Lambda_{-1}(G) + 1$. Equality holds if and only if G is complete multipartite.

F20: [Bu] For a connected graph G on n vertices, $\omega(G) \leq n - \frac{1}{2} rank A_{\overline{G}}$.

F21: Each of the three upper bounds given above can be computed in time $O(n^3)$, and one can find examples that show the bounds are sharp [Bu].

REMARK

R16: Budinich [Bu] tested the upper bounds on a set of 700 random graphs of order 100 and 200. For these graphs, the smaller of the first two upper bounds was almost always better than the third upper bound.

5.2.5 Exact Algorithms

Clique Enumeration

Harary and Ross [HaRo57] initiated an algorithmic and theoretical study of the enumeration of all cliques in a graph. This topic has a variety of applications (see, e.g., [Bo64,HaRo57,PaUn59]). The first significant theoretical result (Fact 22) is due to Erdös, Moon, and Moser [MoMo65]. The currently best theoretical results regarding the performance of clique-enumeration algorithms are given in Facts 23 and 24.

DEFINITION

D13: Given a graph G, the ***arboricity*** is the minimum number of edge-disjoint acyclic subgraphs whose union is G.

FACTS

F22: [MoMo65] The maximum number of cliques in an n-vertex graph equals

$$\begin{cases} 3^{n/3} & \text{if } n \equiv 0\ (mod\ 3) \\ 4 \cdot 3^{(n-4)/3} & \text{if } n \equiv 1\ (mod\ 3) \\ 2 \cdot 3^{(n-2)/3} & \text{if } n \equiv 2\ (mod\ 3) \end{cases}$$

(Some extensions of this result are discussed in [SaVa].)

F23: [ChNi85] There is an algorithm for listing all cliques of a graph $G = (V, E)$ in time $O(a(G)|E|\mu(G))$, where $a(G)$ is the arboricity of G and $\mu(G)$ is the number of cliques in G.

F24: [ToTaTa88] There is an algorithm for listing all cliques of an n-vertex graph in time $O(3^{n/3})$.

REMARKS

R17: The algorithm in Fact 23 improves slightly on the running time $O(|V||E|\mu(G))$ of the algorithm given in [TsIdAvSh77].

R18: The algorithm in Fact 24 is a modification of the backtracking algorithm of Bron and Kerbosch [BrKe73]. In light of the Erdős-Moon-Moser result in Fact 22, the result in Fact 24 is, in a sense, best possible.

COMPUTATIONAL NOTE: In computational experiments with graphs of order ranging from 30 to 220, Loukakis [Lo83] showed that his depth-first enumerative algorithm is much faster than the algorithms from [BrKe73], [TsIdAvSh77], and [LoTs81]. Loukakis's algorithm seems to be among the most efficient practical algorithms.

Maximum-Clique and Maximum-Weight-Clique Algorithms

Clearly, the algorithms mentioned previously can be used (directly or after simple modifications) to find a maximum clique in a graph. However, the maximum-clique problem has attracted more attention than the clique-enumeration problem and the use of certain procedures has made algorithms for the maximum-clique problem quite fast.

FACT

F25: [Ro86] There is an algorithm for solving the maximum-clique problem with time complexity $O(n^{0.276})$, where n is the number of vertices.

REMARKS

R19: Fact 25 was established by Robson [Ro86] by modifying a recursive algorithm of Tarjan and Trojanowski [TaTr77] and by using a detailed case analysis. The result seems to be the best time-complexity upper bound currently known for maximum-clique algorithms.

COMPUTATIONAL NOTE: Branch-and-cut algorithms were used with great success for several combinatorial optimization problems, see, e.g., Section 5 of Chapter 4. However, for the maximum-clique problem, branch-and-cut algorithms currently remain, in general, inferior to the state-of-the-art branch-and-bound algorithms [RoSm01].

COMPUTATIONAL NOTE: There are several quite efficient branch-and-bound algorithms that use fast coloring heuristics to produce upper and lower bounds. See, e.g., the new algorithm in [Os02], which uses a coloring heuristic by Biggs [Bi90]. Additional information and references may be found in [Os02].

R20: Branch-and-bound algorithms for the maximum-weight-clique problem are discussed in [BoBuPaPe99,Os01]. In [Os01], a new algorithm is compared with a few others.

R21: Some algorithms have been compared on random graphs and on special families of graphs inspired by applications. For the maximum-clique problem, frequently-occurring special families of graphs are collected in [JoTr96] and explained in [HaPaVa93]. Östergård [Os01] introduces a special family of instances for the maximum-weight-clique problem.

R22: There are many algorithms for the maximum-clique and maximum-weight-clique problems. Some partial comparisons seem to suggest that the relative performance of various algorithms vary for different graph densities and instances (see, e.g., [Os01,Os2]).

COMPUTATIONAL NOTE: There are a few maximum-clique and maximum-weight-clique computer codes freely available for research purposes, see, e.g., [Di,NiOs03].

5.2.6 Heuristics

When the instance of the maximum-weight clique problem under consideration is of large size or the data is not precise (which is the case in many applications) or a solution has to be obtained very quickly, one should resort to heuristics rather than exact algorithms. Moreover, heuristics form important parts of many exact algorithms.

Construction Heuristics and Local Search

D14: A ***construction heuristic*** produces a feasible solution without any attempt to improve it.

REMARKS

R23: Construction heuristics are normally very fast and provide quick solutions, and lower or upper bounds. However, their solutions cannot be expected to be of high quality.

R24: The simplest construction heuristic for the maximum-clique problem is to add one vertex at a time to an emerging clique. It is logical to choose in each iteration an eligible vertex of maximum degree [KoRu87]. Alternatively, one may delete vertices from the given graph one by one until a clique is obtained [KoRu87].

R25: Most approximation algorithms for various optimization problems are, in fact, construction heuristics. Examples of construction heuristics used in maximum-clique approximation algorithms may be found in [BoHa92] and [Fi].

DEFINITIONS

D15: A ***local search (LS)*** heuristic starts from a feasible solution and in each iteration until termination, chooses the next solution from a *neighborhood* of solutions that are in some prescribed sense, close to the current solution.

D16: An ***improvement LS*** (or simply ***local improvement***) is a local search that always chooses a solution that is better than the current one and terminates when it cannot find one in the neighborhood.

COMPUTATIONAL NOTE: Local improvement does not appear to perform particularly well for the maximum-clique problem [GuGuMiZv] as it may terminate at a relatively small clique. Local search algorithms that do not require monotonic improvement are much more flexible in that they can escape from local optima that are non-maximum cliques; perhaps the most flexible among them is briefly discussed below.

Tabu Search

This *metaheuristic* appears to provide a good trade-off between computational time and solution quality.

DEFINITION

D17: ***Tabu search*** is a local search in which solutions that are worse than the current one can be chosen provided that they are not in any of the so-called *tabu lists*.

REMARKS

R26: Tabu search was introduced independently by Glover [Gl89,Gl90] and Hansen and Jaumard [HaJa90]. 'Pure' tabu search techniques for the maximum-clique problem were implemented in a number of papers, see, e.g., [FrHeWe90,SoGe96].

COMPUTATIONAL NOTE: Tabu-search algorithms use parameters that have to be fine-tuned in order to achieve good results. This slows down the development and use of tabu-search computer codes. Battiti and Protasi [BaPr01] deal with this issue by adjusting the parameters using an internal learning loop.

R27: For brief descriptions and discussions of other metaheuristics applied to the maximum-clique problem, see [BoBuPaPe99].

REFERENCES

[AlFeWiZu95] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman, Derandomized graph products. *Computational Complexity* 5 (1995), 60-75.

[AlSp92] N. Alon and J. H. Spencer, *The Probabilistic Method*, Wiley, 1992.

[AmHa72] A. T. Amin and S. L. Hakimi, Upper bounds on the order of a clique of a graph. *SIAM J. Appl. Math.* 22 (1972) 569-573.

[AuCrGaKaMaPr99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation*, Springer, 1999.

[BaPr01] R. Battiti and M. Protasi, Reactive local search for the maximum clique problem. *Algorithmica* 29 92001), 610-637.

[BePe90] P. Berman and A. Pelc, Distributed fault diagnosis for multiprocessor systems. In *Proc. 2oth Annual Int. Symp. Fault-Tolerant Comput.* (Newcastle, UK), 340-346, 1990.

[Bi90] N. Biggs, Some heuristics for graph coloring. In *Graph Colourings* (R. Nelson and R. J. Wilson, eds.), Longman, 1990.

[Bo97] I. M. Bomze, Evolution towards the maximum clique. *J. Global Optim.* 10 (1997), 143-164.

[BoBuPaPe99] I. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, The Maximum Clique Problem. In *Handbook of Combinatorial Optimization (Supplement Volume A)* (D.-Z. Du and P. M. Pardalos, eds.), Kluwer, 1999.

[Bo64] R. E. Bonner, On some clustering techniques. *IBM J. Res. Develop.* 8 (1964), 22-32.

[BrKe73] C. Bron and J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM* 16 (1973), 575-577.

[Bu] M. Budinich, Bounds on the maximum clique of a graph. *Discrete Appl. Math.*, to appear.

[Ca79] Y. Caro, New results on the independence number. Tech. Report, Tel-Aviv University, 1979.

[ChNi85] N. Chiba and T. Nashizeki, Arboricity and subgraph listing algorithms. *SIAM J. Comput.* 14 (1985) 210-223.

[Di] C programs available at ftp://dimacs.rutgers.edu/pub/challenge/graph/solvers/

[Eg31] E. Egerváry, On combinatorial properties of matrices. *Math. Lapok* 31 (1931), 16-28.

[Fi] U. Fiege, Approximating maximum clique by removing subgraphs. Manuscript, 2002.

[FrHeWe89] C. Friden, A. Hertz, and M. de Werra, STABULUS: A technique finding stable sets in large graphs with tabu search. *Computing* 42 (1989), 35-44.

[GrYe99] J. L. Gross and J. Yellen, *Graph Theory and Its Applications*, CRC Press, 1999.

[Gl89] F. Glover, Tabu search - Part I. *ORSA J. Comput.* 1 (1989), 190-260.

[Gl90] F. Glover, Tabu search - Part II. *ORSA J. Comput.* 2 (1990), 4-32.

[GuGuMiMa00] N. Gulpinar, G. Gutin, G. Mitra, and I. Maros, Detecting embedded network structures in linear programs. *Comput. Opt. Appl.* 15 (2000) 235-247.

[GuGuMiZv] N. Gulpinar, G. Gutin, G. Mitra, and A. Zverovitch, Extracting pure network submatrices in linear programs using signed graphs. *Discrete Appl. Math.*, to appear.

[GuVaYe] G. Gutin, A. Vainshtein, and A. Yeo, Domination analysis of combinatorial optimization problems. Preprint, 2002.

[Ha98] M. Halldórsson, Approximation of independent sets in graphs. In *Proc. of APPROX'98*, 1998, 1-13.

[HaJa90] M.-H. Han and D. Jang, The use of maximum curvature points for the recognition of partially occluded objects. *Pattern Recognition* 23 (1990), 21-33.

[HaJa90] P. Hansen and B. Jaumard, Algorithms for the maximum satisfability problem. *computing* 44 (1990) 279-303.

[HaPaVa93] J. Hasselberg, P. P. Pandalos, and G. Vairaktarakis, Test case generators and computational results for the maximum clique problem. *J. Global Optim.* 3 (1993), 463-482.

[HaRo57] F. Harary and I. C. Ross, A procedure for clique detection using the group matrix. *Sociometry* 20 (1957) 205-215.

[Ha99] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica* 182 (1999), 105-142.

[HoJo85] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge Uni. Press, 1985.

[HoSk89] R. Horaud and T. Skordas, Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Pattern. Anal. Machine Intell.* 11 (1989), 1168-1180.

[JoTr96] D. S. Johnson, and M. Trick (eds.), *Clique, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Vol. 26, AMS, 1996 (see also http://dimacs.rutgers.edu/Volumes/Vol26.html).

[Ka72] R. M. Karp, Reducibility among combinatorial problem. In *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds.), Plenum Press, 1972.

[Ko31] D. Kőnig, Graphen und matrizen. *Math. Lapok* 38 (1931), 116-119.

[KoRu87] R. Kopf and G. Ruhe, A computational study of the weighted independent set problem for general graphs. *Found. Control Eng.* 12 (1987), 167-180.

[LaTi01] H. Y. Lau and H. F. Ting, The greedier the better: an efficient algorithm for approximating maximum independent set. *J. Combin. Optim.* 5 (2001), 411-420.

[Lo83] E. Loukakis, A new backtracking algorithm for generating the family of maximal independent sets of a graph. *Comp. Math. Appl.* 9 (1983) 583-589.

[LoTs81] E. Loukakis and C. Tsouros, A depth first search algorithm to generate the family of maximal independent sets of a graph lexicographically. *Computing* 27 (1981), 249-266.

[MaSl79] J. MacWillimas and N. J. A. Slone, *The Theory of Error Correcting Codes*, North-Holland, 1979.

[MoMo65] J. W. Moon and L. Moser, On cliques in graphs, *Israel J. Math.* 3 1965), 23-28.

[MoSt65] T. S. Motzkin and E. G. Straus, Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.* 17 (1965), 533-540.

[NeTr74] G. L. Nemhauser and L. E. Trotter, Properties of vertex packings and independence system polyhedra. *Math. Prog.* 6 (1974), 48-61.

[NeTr75] G. L. Nemhauser and L. E. Trotter, Vertex packings: Structural properties and algorithms. *Math. Prog.* 8 (1975), 232-248.

[NiOs03] S. Niskanen and P. R. J. Östergård, Cliquer User's Guide, Version 1.0, *Communications Lab., Helsinki Univ. Technology, Tech. Rep.* T48, 2003. The current release of the `Cliquer` code is available from `www.hut.fi/~pat/cliquer.html`

[Os01] P. R. Östergård, A new algorithm for the maximum-weight clique problem. *Nordic J. Comput.* 8 (2001), 424-436.

[Os02] P. R. Östergård, A fast algorithm for the maximum clique problem. *Discrete Appl. Math.* 120 (2002), 197-207.

[PaPh90] P. M. Pardalos and A. T. Phillips, A global optimization approach for solving the maximum clique problem. *Int. J. Comput. Math.* 33 (1990), 209-216.

[PaUn59] M. C. Paull and S. H. Unger, Minimizing the number of sates in incompletely specified sequential switching functions. *IRE Trans. Electr. Comput.* EC-8 (1959), 356-367.

[PeJa95] M. Pelillo and A. Jagota, Feasible and infeasible maxima in a quadratic program for maximum clique. *J. Artif. Neural Networks* 2 (1995) 411-420.

[Ro86] J. M. Robson, Algorithms for maximum independent sets. *J. Algorithms* 7 (1986), 425-440.

[RoSm01] F. Rossi and S. Smriglio, A branch-and-cut algorithm for the maximum cardinality stable set problem. *Oper. Res. Lett.* 28 (2001), 63-74.

[SaVa] B. E. Sagan and V. R. Vatter, Maximal independent sets in graphs with at most r cycles. Preprint, 2002.

[SaToYa03] S. Sakai, M. Togasaki, and K. Yamazaki, A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Appl. Math.* 126 (2003) 313-322.

[Se94] S. M. Selkow, A probabilistic lower bound on the independence number of graphs. *Discrete Math.* 132 (1994), 363-365.

[Sh90] N. Z. Shor, Dual quadratic estimates in polynomial and Boolean programming. In *Computational Methods in Global Optimization* (P. M. Pardalos and J. B. Rosen, eds.), *Ann. Oper. Res.* 25 (1990), 163-168.

[SoGe96] P. Soriano and M. Gendreau, Tabu search algorithms for the maximum clique problem. In [JoTr96], 221-242, 1996.

[TaTr77] R. E. Tarjan and A. E. Trojanowski, Finding a maximum independent set. *SIAM J. Comput.* 13 (1977), 537-546.

[ToTaTa88] E. Tomita, A. Tanaka and H. Takahashi, The worst-case time complexity for finding all the cliques. *Tech. Report UEC-TR-C1* 1988.

[TsIdAvSh77] S. Tsukiyama, M. Ide, H. Aviyoshi and I. Shirakawa, A new algorithm for generating all maximum independent sets. *SIAM J. Comput.* 6 (1977) 505-517.

[We81] V. K. Wei, A lower bound on the stability number of a simple graph. Bell Lab. Tech. Memo., No. 81-11217-9, 1981.

[Wi67] H. S. Wilf, The eigenvalues of a graph and its chromatic number. *J. London Math. Soc.* 42 (1967) 330-332.

[Wi86] H. S. Wilf, The spectral bounds for the clique and independent numbers of graphs. *J. Combin. Theory B* 40 (1986) 113-117.

GLOSSARY

approximate (or approximation) algorithm: an algorithm that typically makes use of heuristics in reducing its computation but produces solutions that are not necessarily optimal.

arboricity of a graph G: the minimum number of edge-disjoint acyclic subgraphs whose union is G.

clique: a maximal mutually adjacent set of vertices.

construction heuristic: a heuristic that produces a feasible solution without any attempt to improve it.

exact algorithm: an algorithm that solves a certain optimization problem to optimality.

Hamming distance between a pair $u = (u_1, \ldots, u_n)$, $v = (v_1, \ldots, v_n)$ **of binary vectors**: the number of indices i for which $u_i \neq v_i$.

Hamming graph $H(n, d)$: a graph whose vertices are all binary vectors with n coordinates. A pair u, v of vertices in $H(n, d)$ are adjacent if the Hamming distance between them is at least d.

improvement local search: a local search in which we choose only a solution that is better than the current one and stop if we cannot find one.

independent set: a mutually non-adjacent set of vertices.

independence number: the number of vertices in a maximum-size independent set of a graph.

local search: a heuristic that starts from a feasible solution and in each iteration, until termination, chooses the next solution from a *neighborhood* of solutions that are in some prescribed sense, close to the current solution.

maximum clique: the number of vertices in a maximum-size clique of a graph.

Perron root $\lambda(G)$: the largest eigenvalue of A_G.

standard simplex: $\Delta = \{x \in R^n : \ x_i \geq 0, \ i = 1, \ldots, n, \ \sum_{j=1}^n x_j = 1\}$.

tabu search: a local search in which solutions that are worse than the current one can be chosen provided that they are not in any of the so-called tabu lists.

vertex cover: a set S of vertices such that at least one endpoint of every edge of a graph is in S.

vertex-weighted graph: a graph G is in which every vertex x is assigned a non-negative weight $w(x)$.