Computing 46, 321–341 (1991)

Computing

© by Springer-Verlag 1991

Finding Maximum Cliques in Arbitrary and in Special Graphs

L. Babel, München

Received September 28, 1990

Abstract — Zusammenfassung

Finding Maximum Cliques in Arbitrary and in Special Graphs. The classical problem of finding a clique of largest cardinality in an arbitrary graph is NP-complete. For that reason earlier work diverges into two directions. The first concerns algorithms solving the problem for arbitrary graphs in reasonable (but exponential) time, the other restricts to special classes of graphs where polynomial methods can be found. Here, the two directions are combined in a way. A branch and bound algorithm is developed treating the general case. Computational experiments on random graphs show that this algorithm compares favorable to the fastest known method. Furthermore, it consumes only polynomial time for quite a few graph classes. For some of them no polynomial solution method is given so far.

AMS Subject Classification: O5C35, 68R10

Key words: Maximum clique problem, branch and bound algorithm, polynomial solvable problems.

Bestimmung von Maximum Cliquen in beliebigen und in speziellen Graphen. Das klassische Problem der Ermittlung einer Clique größter Mächtigkeit in einem beliebigen Graph ist NP-vollständig. Deshalb teilen sich bisherige Untersuchungen in zwei Richtungen. Die erste beschäftigt sich mit Algorithmen, die das Problem für beliebige Graphen in vernünftiger (aber exponentieller) Zeit lösen, die andere beschränkt sich auf spezielle Graphenklassen, für die polynomiale Methoden möglich sind. Hier werden diese beiden Richtungen kombiniert. Es wird ein Branch and Bound-Algorithmus für den allgemeinen Fall entwickelt. Praktische Rechenexperimente an Zufallsgraphen zeigen, daß dieser Algorithmus dem schnellsten bisher bekannten Verfahren überlegen ist. Darüberhinaus benötigt er nur polynomiale Rechenzeit für eine Vielzahl von Graphenklassen, darunter einige, für die noch keine polynomiale Lösungsmethode bekannt ist.

1. Introduction

Given an undirected graph a *clique* is defined to be a subgraph with pairwise adjacent vertices. A *maximum clique* is a clique of largest cardinality. A set of vertices is called *independent set* if there is no edge between any two of these vertices and *vertex cover* if every edge of the graph is incident to one of the vertices. Obviously the problem of finding a maximum clique is equivalent to finding an independent set of largest or a vertex cover of smallest cardinality in the complement of the graph.

There are a lot of important applications of these problems in practice reaching from computer vision, information retrieval, cluster analysis, classification theory to signal transmission (see for example [5], [13]–[15]).

The maximum clique problem, one of the classical problems in combinatorial optimization, is well known to be NP-complete [7]. So the existence of a polynomial time solution method for arbitrary graphs seems unprobable. For that reason earlier research has divided into two directions. The first concerns algorithms solving the problem for arbitrary graphs in reasonable but exponential time. For the most interesting of them see [13]–[22]. The other direction restricts to special graph classes as chordal, comparability, circular arc, circle graphs and many others where polynomial methods can be found [23]–[36]. This situation is not so favorable in a certain sense. Suppose a greater number of graphs have to be examined for their maximum cliques. We want to solve the problems efficiently, that means in polynomial time whenever possible. For this purpose it is necessary to implement many special algorithms for the different occurring graph classes, furthermore an algorithm which produces the solution for graphs which do not belong to any of this classes (often it is not known in advance whether a graph belongs to a certain class such that expensive tests have to be performed to check this). Naturally, it would be of great advantage to have a kind of a universal method which covers both directions. This approach is realized in this paper. We develop an algorithm which solves the problem very fast for arbitrary graphs and which additionally has polynomial running time for quite a few graph classes.

The paper is organized as follows. First some basic graphtheoretical definitions and notations are stated. In section 3 we present the solution method, a branch and bound algorithm. The main ingredients of this algorithm are an economic way to compute bounds by colorings of graphs and branching rules that use the information of the colorings. After that interesting complexity results of the algorithm are shown when the input is restricted to graphs with certain properties. Beneath this theoretical performance results the algorithm also works very well in practice. In section 5 computational comparisons with the most efficient algorithm known so far are documented. The new method is considerably faster, especially if very difficult problems are examined.

2. Graphtheoretical Notation

A *graph* $G = (V, E)$ is an ordered pair consisting of a finite set V of *vertices* and a set of unordered pairs (u, v) of distinct vertices, called *edges*. n and m denotes the number of vertices and edges respectively. n is also called *cardinality* of the graph. Two vertices u and v are *adjacent* or *neighbours* if $(u, v) \in E$. $N(v) = \{u \in V | (v, u) \in E\}$ is the *neighbourhood*, $\deg(v) = |N(v)|$ the *degree* of v. For $X \subseteq V$ let $\deg_X(v) = |N(v) \cap X|$. If $e = (u, v) \in E$ then vertex v is said to be *incident* to edge e. Two edges are *adjacent* if they share a common vertex.

$\bar{G} = (V, \bar{E})$ with $\bar{E} = \{(u, v) | (u, v) \notin E, u, v \in V, u \neq v\}$ is the *complement* of G. $G(V') = (V', E')$ is an (*induced*) *subgraph* of $G = (V, E)$, $G(V') \subseteq G$ for short, if $V' \subseteq V$ and $E' = \{(u, v) | (u, v) \in E, u, v \in V'\}$. If $H \not\subseteq G$ for a graph H we call G to be *H-free*, $G' = (V', E')$ is a *partial subgraph* of G if $V' \subseteq V$ and $E' \subseteq \{(u, v) | (u, v) \in E, u, v \in V'\}$.

A *clique* or *complete graph* is defined to be a graph $G(Cl) \subseteq G$ with $(u, v) \in E$ for all $u, v \in Cl$. $G(Cl)$ is called *maximal* if there is no clique $G(Cl')$ with $Cl \subset Cl'$ and *maximum* if no clique $G(Cl'')$ exists with $|Cl''| > |Cl|$. The *clique number* $\omega(G)$ denotes the cardinality of a maximum clique in G. The vertex set of a graph is *independent* if the complement of the graph is complete.

K_r denotes a complete graph, S_r a graph with independent vertex set of cardinality r. $K_{r,s}$ is a graph whose vertex set V can be partitioned into sets V_1, V_2, $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \varnothing$ with $|V_1| = r, |V_2| = s$ and $(u, v) \in E \Leftrightarrow u \in V_1, v \in V_2$. A *path* P_r of *length* r is a graph with $V = \{v_1, v_2, \ldots, v_r\}$ and $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{r-1}, v_r)\}$. v_1 and v_r are the *endpoints* of the path. We write $P_r = (v_1, v_2, \ldots, v_r)$. If additionally $r \geq 3$ and $(v_r, v_1) \in E$ then the graph is said to be a *cycle* C_r of *length* r. Given a graph $G_0 = (V_0, E_0)$ with vertices $v_1, \ldots, v_n$ and disjoint graphs $G_i = (V_i, E_i)$, $i = 1, 2, \ldots, n$, the *composition graph* $G = G_0(G_1, G_2, \ldots, G_n) = (V, E)$ is defined by $V = \bigcup_{i=1}^{n} V_i$, $E = \bigcup_{i=1}^{n} E \cup \{(u, v) | u \in V_i, v \in V_j, 1 \leq i < j \leq n, (v_i, v_j) \in E_0\}$. G is *connected* if for every pair of vertices u, v there exists a path with endpoints u and v. The maximal connected subgraphs of G are called *connected components*.

3. Branch and Bound Algorithm

3.1. General Branching Scheme and Exhaustive Search Tree

In this subsection we develop a general solution scheme for the maximum clique problem. It will be specified later in order to get a powerful method. The resulting algorithm is a further development and refinement of the algorithm [2].

Given a graph $G = (V, E)$ let $(v_1, v_2, \ldots, v_n)$ be any ordering of the vertices of G. Define $V_i := N(v_i) \cap \{v_{i+1}, \ldots, v_n\}$. Then obviously the following holds:

Lemma 1: *If Cl is the vertex set of a clique in G then $Cl \subseteq \{v_i\} \cup V_i$ for at least one $i \in \{1, 2, \ldots, n\}$.*

With this fact the maximum clique problem in a graph of cardinality n can be transformed into n problems in graphs $G(V_i)$ of cardinality at most $n - i$, $i = 1, 2, \ldots, n$. Let $P(U_t, V_t)$ with $U_t, V_t \subseteq V$, $U_t \cap V_t = \varnothing$, $V_t \subseteq \bigcap_{u \in U_t} N(u)$ and $G(U_t)$ complete denote the problem $MC(G(V_t))$: *Find a maximum clique in $G(V_t)$*. With $U_0 = \varnothing$ and $V_0 = V$ the problem $P(U_0, V_0)$ corresponds to the original problem $MC(G)$. The vertices of U_t and V_t are called *fixed* and *free* resp. The structure of this vertex sets results from the

General branching scheme:
Given $P(U_t, V_t)$ let $(v_1^t, v_2^t, \ldots, v_{n_t}^t)$
be an ordering of the vertices of V_t.
For $i = 1, 2, \ldots, n_t$ do
$U_{t_i} := U_t \cup \{v_i^t\}$
$V_{t_i} := N(v_i^t) \cap \{v_{i+1}^t, \ldots, v_{n_t}^t\}$

With this rule an *exhaustive search tree* can be constructed. The problem $P(U_0, V_0)$

corresponds to the root. The subproblems created by the branching rule are the nodes of the tree. $P(U_t, V_t)$ has $n_t = |V_t|$ successors $P(U_{t_1}, V_{t_1}), \ldots, P(U_{t_{n_t}}, V_{t_{n_t}})$. Notice that the shape of the tree depends on the choice of the vertex orderings. If $V_t = \varnothing$ then no further branching is possible and $P(U_t, V_t)$ is a leaf of the tree. It's distance to the root is equal to $|U_t|$. Every leaf with greatest distance corresponds to a maximum clique. The size of the tree can be estimated from above as follows.

Lemma 2: *The number of nodes in the exhaustive search tree with distance at most* k *from the root is restricted by* $b(n,k) = \sum_{i=0}^{k} \binom{n}{i}$.

Proof: The tree has maximal number of nodes if the graph to be examined is complete. Consider the sets of fixed vertices U_t in $P(U_t, V_t)$. If the distance from $P(U_t, V_t)$ to the root is equal to i then $|U_t| = i$. The sets U_t of all subproblems of distance i are distinct. Moreover for every set $U \subseteq V$ of cardinality i there exists a $P(U_t, V_t)$ with $U_t = U$. This shows that the number of nodes of distance i is $\binom{n}{i}$. □

In order not to build the whole exhaustive search tree (the effort for that would be $O\left(\sum_{i=0}^{n} \binom{n}{i}\right) = O(2^n)$) bounds for the clique number are computed which allow to exclude many of the subproblems from further investigation.

3.2. Computing Bounds

A *coloring* of a graph $G = (V, E)$ is a mapping $c: V \to M \subseteq \mathbb{N}$ with the property $(u, v) \in E \Rightarrow c(u) \neq c(v)$ for all $u, v \in V$. If c is surjective and $M = \{1, 2, \ldots, k\}$ then c is called a *k-coloring*. The *chromatic number* $\chi(G)$ is defined to be the smallest k such that a k-coloring exists for G. Every $\chi(G)$-coloring is an *optimal coloring*. In a colored graph all vertices of a maximum clique must have different colors. Therefore, the well known inequality

$$\omega(G) \leq \chi(G)$$

holds. The coloring problem, however, is NP-complete as well as the clique problem [7]. For that reason we turn to approximate colorings, i.e. easily obtainable colorings with eventually more than $\chi(G)$ colors. A good working coloring heuristic is due to Brelaz [4]. In this procedure the vertices of the graph are colored sequentially with the smallest possible color. The order is established by choosing in each step a vertex with maximal saturation degree (the *saturation degree* is defined to be the number of different colors in the neighbourhood of the vertex). This method can be extended without increasing complexity to find the connected components of the graph and to compute simultaneously lower and upper bounds for the clique number of every component. A detailed specification appears below.

Let *cdeg*(v) denote the number of colored neighbours, *satdeg*(v) the saturation degree and *neighbcol*(v) a set containing the colors which occur in the neighbourhood of

vertex v (consequently satdeg(v) = |neighbcol(v)|). Further let W be the set of all uncolored vertices and k a counter for the number of connected components.

Procedure Bounds1(G)

1. For $v \in V$ do
   $c(v) := \text{cdeg}(v) := \text{satdeg}(v) := 0$, $\text{neighbcol}(v) := \varnothing$
   $W := V$, $k := 1$
   $V^1 := Cl^1 := \varnothing$, clmax := false
2. $X := \{v \in W | \text{satdeg}(v) \geq \text{satdeg}(w) \text{ for all } w \in W\}$
(*) Choose $v^* \in X$ with $\text{cdeg}(v^*) \geq \text{cdeg}(w)$ for all $w \in X$
   If satdeg(v^*) = 0 and $V \neq W$
   then $\tilde{\chi}(G(V^k)) := \max\{i \in \mathbb{N} | \exists v \in V^k \text{ with } c(v) = i\}$
   $\tilde{\omega}(G(V^k)) := |Cl^k|$
   $k := k + 1$,
   $V^k := \varnothing$, $Cl^k := \{v^*\}$, clmax := false
   else if clmax = false
   then if satdeg(v^*) = $|Cl^k|$
   then $Cl^k := Cl^k \cup \{v^*\}$
   else clmax := true
3. $c(v^*) := \min\{i \in \mathbb{N} | i \notin \text{neighbcol}(v)\}$
   $W := W - \{v^*\}$, $V^k := V^k \cup \{v^*\}$
   For $v \in N(v^*) \cap W$ do
   $\text{cdeg}(v) := \text{cdeg}(v) + 1$
   if $c(v^*) \notin \text{neighbcol}(v)$
   then $\text{neighbcol}(v) := \text{neighbcol}(v) \cup \{c(v^*)\}$
   $\text{satdeg}(v) := \text{satdeg}(v) + 1$
4. If $W \neq \varnothing$ then goto 2.
5. $\tilde{\chi}(G(V^k)) := \max\{i \in \mathbb{N} | \exists v \in V^k \text{ with } c(v) = i\}$
   $\tilde{\omega}(G(V^k)) := |Cl^k|$
6. $\tilde{\chi}(G) := \max\{\tilde{\chi}(G(V^i)) | i \in \{1, 2, \ldots, k\}\}$
   $\tilde{\omega}(G) := \tilde{\omega}(G(V^{i*})) := \max\{\tilde{\omega}(G(V^i)) | i \in \{1, 2, \ldots, k\}\}$
   $Cl := Cl^{i*}$

Theorem 1: *For any graph G the procedure Bounds1 produces a maximal complete subgraph G(Cl) and a $\tilde{\chi}(G)$-coloring, consequently a lower bound $\tilde{\omega}(G) = |Cl|$ and an upper bound $\tilde{\chi}(G)$ for the clique number $\omega(G)$. If G is not connected then for every component $G(V^i)$, $i = 1, 2, \ldots, k$, lower and upper bounds $\tilde{\omega}(G(V^i))$, $\tilde{\chi}(G(V^i))$ are obtained.*

Proof: If G is connected then k does not change it's value ($k = 1$). The first vertices which are colored induce a clique. The vertex set Cl^1 of this clique can be extended as long as there exists a vertex v^* with saturation degree equal to $|Cl^1|$ (then v^* is adjacent to all vertices in Cl^1). If there is no such vertex the clique is maximal and the boolean variable clmax is set 'true'. Obviously the cardinality $\tilde{\omega}(G) = \tilde{\omega}(G(V^1)) = |Cl^1|$ of the clique and the number $\tilde{\chi}(G) = \tilde{\chi}(G(V^1))$ of used colors are lower and upper bounds for $\omega(G)$.

Let G be disconnected with components $G^1, \ldots, G^k$. Due to the selection of v^* as a vertex of maximal saturation degree G is colored component by component. A new component starts if and only if v^* has saturation degree 0. For every component $G^i = G(V^i)$ bounds $\tilde{\omega}(G(V^i))$ and $\tilde{\chi}(G(V^i))$ are computed in the way described above. The largest of this lower resp. upper bounds yield the lower and upper bound for $\omega(G)$. □

Theorem 2: *Procedure Bounds1 works in time $O(m + n)$.*

Proof: Straightforward. □

In order to get sharper lower bounds it is favourable to use a modification of the above bounding procedure, called *Bounds2*, which arises if line (∗) is replaced by

(∗∗) If clmax = false
then choose $v^* \in X$ with $\deg_X(v^*) \geq \deg_X(w)$ for all $w \in X$
else choose $v^* \in X$ with $\text{cdeg}(v^*) \geq \text{cdeg}(w)$ for all $w \in X$

In the partially colored graph X contains all uncolored vertices with largest saturation degree. Different from *Bounds1* where always a vertex $v^* \in X$ with maximal degree in the colored subgraph is chosen, here during the clique search v^* is a vertex of maximal degree in $G(X)$. Particularly the modified procedure starts with a vertex of maximal degree in G. We will talk about *extended clique search* in contrast to *elementary clique search* when using (∗).

Theorem 3: *Procedure Bounds2 can be implemented to run in time $O(m + n)$.*

Proof: The crucial point is the repeated computation of the vertex degrees in $G(X)$. Build an auxiliary array of possible degrees $1, 2, \ldots, \min\{n - 1, m\}$. Every number k in the array is associated with a doubly linked list of vertices of degree k. Computing the degrees in G and initializing the data structure can be done in $O(m + n)$. Updating the structure and finding v^* (the maximal degree decreases monotonously) in each step (∗∗) requires effort $O(r_i)$ where r_i is the number of edges which are eliminated from $G(X)$ in the i-th iteration. Since $\sum r_i = m$ the assertion follows. □

We have seen that coloring the vertices according to the size of the saturation degree yields very useful additional information. The connected components together with lower and upper bounds for the clique number in each of them can be derived with low effort. Consider now the problem $P(U_t, V_t)$. Let $G_t := G(U_t \cup V_t)$ and $G_t^i := G(U_t \cup V_t^i)$, $i = 1, 2, \ldots, k$. The clique number ω_t and the chromatic number χ_t of G_t are given by $\omega_t = |U_t| + \omega(G(V_t))$ and $\chi_t = |U_t| + \chi(G(V_t))$. Procedure *Bounds1/2* applied to the graph $G(V_t)$ yields

$$\tilde{\omega}(G(V_t)) = \max\{\tilde{\omega}(G(V_t^i)) | i \in \{1, \ldots, k\}\} \quad \text{and}$$

$$\tilde{\chi}(G(V_t)) = \max\{\tilde{\chi}(G(V_t^i)) | i \in \{1, \ldots, k\}\}$$

with the connected components $G(V_t^1), \ldots, G(V_t^k)$ of $G(V_t)$. Define $\tilde{\omega}_t := |U_t| + \tilde{\omega}(G(V_t))$, $\tilde{\chi}_t := |U_t| + \tilde{\chi}(G(V_t))$ and $\tilde{\chi}_t^i := |U_t| + \tilde{\chi}(G(V_t^i))$, $i = 1, 2, \ldots, k$. Further let ω_B be best known lower bound for $\omega(G)$. If $\tilde{\omega}_t > \omega_B$ the global lower bound can be

improved to $\omega_B := \tilde{\omega}_t$. If $\tilde{\chi}_t \leq \omega_B$ then we have $\omega_t \leq \chi_t \leq \tilde{\chi}_t \leq \omega_B$ and the problem $P(U_t, V_t)$ is settled. The corresponding graph G_t cannot contain a clique with more vertices than the currently largest one. Finally if $\tilde{\chi}_t > \omega_B$ and $\tilde{\chi}_t^i \leq \omega_B$ for $i \in I \subseteq \{1, 2, \ldots, k\}$ the graph G_t can contain a larger clique but G_t^i cannot. Thus when looking for a maximum clique in G_t we can restrict to $G(U_t \cup V_t - \bigcup_{i \in I} V_t^i)$.

3.3. *Improved Branching Rules*

Given $P(U_t, V_t)$ the general branching scheme produces n_t subproblems $P(U_{t_i}, V_{t_i})$. In the course of the algorithm developed so far every graph $G(V_{t_i})$ has to be colored by procedure *Bounds1/2*. This large effort can be reduced by using the information received in the coloring of $G(V_t)$. If $G(V_t)$ is colored then each of it's subgraphs $G(V_{t_i})$ is colored too. One only has to restrict the coloring to the vertices of the subgraphs. More formally let $c: V \to \{1, 2, \ldots, k\}$ be a k-coloring of a graph $G = (V, E)$ and $W \subseteq V$. Then $c: W \to \{1, 2, \ldots, k\}$ is called *restriction of c to* $G(W)$. Denote $c(W) := \{c(w) | w \in W\}$.

If c is a coloring of $G(V_t)$ using $\tilde{\chi}(G(V_t))$ colors then $|c(V_{t_i})| < \tilde{\chi}(G(V_t))$. Obviously $\tilde{\chi}_{t_i}^{\text{a priori}} := |U_{t_i}| + |c(V_{t_i})|$ is an upper bound for ω_{t_i}. It will be called *a priori color bound* for G_{t_i}. The general branching scheme works with any given ordering of the vertices. If it is chosen in a clever way the bounds $\tilde{\chi}_{t_i}^{\text{a priori}}$ can be derived without or only with modest additional effort. We present two suitable orderings.

Ordering in reverse sequence of coloring
The vertices of V_t are labeled in the chronological order they are colored by $v_{n_t}^t, \ldots, v_2^t, v_1^t$, i.e. $v_{n_t}^t$ is colored first, v_1^t last. Remind the proceeding in *Bounds1/2*. Step 2, the selection of the vertex colored next is carried out n_t times. In the $(n_t - i + 1)$-th iteration the vertices $v_{n_t}^t, \ldots, v_{i+1}^t$ are colored, all others are uncolored. A vertex $v^* = v_i^t$ is chosen with maximal saturation degree. This value however is just the number of different colors in $G(N(v_i^t) \cap \{v_{i+1}^t, \ldots, v_{n_t}^t\})$, thus $|c(V_{t_i})| = \text{satdeg}(v_i^t)$. The following rule results:

Branching rule I:
Given $P(U_t, V_t)$ let $(v_{n_t}^t, \ldots, v_1^t)$ be
the coloring order of the vertices of V_t.
For $i = 1, 2, \ldots, n_t$ do
$U_{t_i} := U_t \cup \{v_i^t\}$
$V_{t_i} := N(v_i^t) \cap \{v_{i+1}^t, \ldots, v_{n_t}^t\}$
$\tilde{\chi}_{t_i}^{\text{a priori}} := |U_{t_i}| + \text{satdeg}(v_i^t)$

Ordering according to non increasing colors
Build an order $(v_1^t, \ldots, v_{n_t}^t)$ such that the vertex colors do not increase when moving from left to right. Particularly v_1^t has highest, $v_{n_t}^t$ has color 1. Since the coloring method used here assigns every vertex the smallest possible color, v_i^t is adjacent to vertices of color 1, 2, ..., $c(v_i^t) - 1$. All neighbours of this colors (but no one with greater color) appear to the right of v_i^t in the order. This shows $|c(V_{t_i})| = c(v_i^t) - 1$. We get:

Branching rule II:
Given $P(U_t, V_t)$ let c be a coloring of $G(V_t)$,
$(v_1^t, \ldots, v_{n_t}^t)$ a vertex ordering with $c(v_i^t) \geq c(v_j^t)$, $1 \leq i < j \leq n_t$.
For $i = 1, 2, \ldots, n_t$ do
$U_{t_i} := U_t \cup \{v_i^t\}$
$V_{t_i} := N(v_i^t) \cap \{v_{i+1}^t, \ldots, v_{n_t}^t\}$
$\tilde{\chi}_{t_i}^{\text{a priori}} := |U_{t_i}| + c(v_i^t) - 1$

3.4. Algorithm

We are now able to formulate the branch and bound method.

Algorithm BB

1. *Initialization*
   1.1 Generate the root $P(U_0, V_0)$ of the search tree with $U_0 := \varnothing$, $V_0 := V$ and define the set of active nodes $AN := \varnothing$
   1.2 Compute lower bound $\tilde{\omega}(G)$ with clique $G(Cl)$, $\tilde{\omega}(G) = |Cl|$ and set $\omega_B := \tilde{\omega}(G)$, $MCl := Cl$
   1.3 Compute upper bounds $\tilde{\chi}_0 := \tilde{\chi}(G)$ for the graph G and $\tilde{\chi}_0^j$ for it's connected components $G(V_0^j)$, $j \in J$
   1.4 If $\tilde{\chi}_0 \leq \omega_B$ then goto 5.
   1.5 For $j \in J$ do
   If $\tilde{\chi}_0^j \leq \omega_B$ then $V_0 := V_0 - V_0^j$
   1.6 Set $AN := \{P(U_0, V_0)\}$
2. *Subproblem selection*
   2.1 If $AN = \varnothing$ then goto 5.
   2.2 Choose $P(U_t, V_t) \in AN$ according to the node selection rule and set $AN := AN - \{P(U_t, V_t)\}$
   2.3 If $\tilde{\chi}_t \leq \omega_B$ then goto 2.1
3. *Branching*
   3.1 Generate the successors $P(U_{t_1}, V_{t_1}), \ldots, P(U_{t_{n_t}}, V_{t_{n_t}})$ of $P(U_t, V_t)$ in the search tree according to the branching rule with the a priori color bounds $\tilde{\chi}_{t_i}^{\text{a priori}}$, $i = 1, 2, \ldots, n_t$
4. *Bounding*
   4.1 For all $P(U_{t_i}, V_{t_i})$ with $\tilde{\chi}_{t_i}^{\text{a priori}} > \omega_B$ do
      4.1.1 Compute lower bounds $\tilde{\omega}_{t_i}$ for G_{t_i} with clique $G(Cl)$, $\tilde{\omega}_{t_i} = |Cl|$
      If $\omega_B < \tilde{\omega}_{t_i}$ then set $\omega_B := \tilde{\omega}_{t_i}$ and $MCl := Cl$
      4.1.2 Let $G(V_t^j)$, $j \in J$ be the connected components of $G(V_t)$
      Compute upper bounds $\tilde{\chi}_{t_i}$ for the graph G_{t_i}
      and $\tilde{\chi}_{t_i}^j$ for $G_{t_i}^j$, $j \in J$
      Set $\tilde{\chi}_{t_i} := \min\{\tilde{\chi}_{t_i}, \tilde{\chi}_{t_i}^{\text{a priori}}\}$
      4.1.3 If $\tilde{\chi}_{t_i} \leq \omega_B$ then goto 4.1.6
      4.1.4 For $j \in J$ do
      If $\tilde{\chi}_{t_i}^j \leq \omega_B$ then $V_{t_i} := V_{t_i} - V_{t_i}^j$

4.1.5 Set $AN := AN \cup \{P(U_{t_i}, V_{t_i})\}$
4.1.6
4.2 Goto 2.
5. *Optimal solution*
STOP; ω_B is the clique number, $G(MCl)$ a maximum clique

Remarks:
The node selection rule states the strategy that determines which node of the search tree is treated next. For instance one of the most current rules depth first search and best bound search can be used. From experience the former has advantages concerning space requirements, the latter concerning running time. If best bound search is used then step 2.3 can be modified to the stopping criterion

2.3′ If $\tilde{\chi}_t \leq \omega_B$ then goto 5.

The computation of the bounds is performed by procedure *Bounds1* or *Bounds2*. The corresponding versions of algorithm BB will be denoted by BB1 and BB2 resp.

It is straightforward to modify the algorithm such that all maximum cliques are discovered.

4. Polynomially Solvable Problems

In the following the *search tree* $\mathcal{ST}$ belonging to a graph G denotes the set of all problems $P(U_t, V_t)$ which are generated if a version of algorithm BB is applied to G. $\mathcal{ST}$ is the actually examined part of the exhaustive search tree. In particular $\mathcal{ST}$ always contains the original problem $P(U_0, V_0) = P(\emptyset, V)$. $\text{depth}(\mathcal{ST}) := \max\{|U_t| \,|\, P(U_t, V_t) \in \mathcal{ST}\}$ is called *depth* of the search tree $\mathcal{ST}$. It corresponds to the greatest distance of a tree node to the root $P(U_0, V_0)$ and can be used as a measure for the effort of the algorithm. If no branching is necessary for the original problem then $\mathcal{ST}$ consists of this problem only and $\text{depth}(\mathcal{ST}) = 0$. In general obviously $\text{depth}(\mathcal{ST}) \in \{0, 1, \ldots, \omega(G)\}$.

Theorem 4: *Suppose algorithm* BB1 *or* BB2 *applied to a graph G yields a search tree* $\mathcal{ST}$ *with* $\text{depth}(\mathcal{ST}) \leq k$. *Then the time spent to solve* $MC(G)$ *is restricted by* $O(b(n,k) \cdot (m+n))$.

Proof: The main work in both versions of algorithm BB is the computation of lower and upper bounds for each subproblem. The procedures *Bounds1/2* developed for this purpose have time complexity $O(m+n)$. All other steps joined with a subproblem as node selection, test of the stopping criterion and eventually branching don't increase this complexity. If the depth of the search tree is at most k then due to Lemma 2 not more than $b(n,k) = \sum_{i=0}^{k} \binom{n}{i}$ subproblems have to be treated. □

In this section we state graph classes with search trees of limited depth k (k a constant). Hence the algorithm produces a maximum clique in polynomial time.

Some of this classes are well known and explored but for others no polynomial solution method is given in the literature so far.

4.1. *Elementary Clique Search*

Lemma 3: *If $G = (V, E)$ is disjoint union of complete graphs then depth$(\mathcal{SF}) = 0$.*

Proof: Let $G(V^1), \ldots, G(V^k)$ be the connected components of G. Procedure *Bounds1* computes both lower bounds $\tilde{\omega}(G(V^i))$ and upper bounds $\tilde{\chi}(G(V^i))$ for $i = 1, \ldots, k$. $G(V^i)$ is complete, therefore $\tilde{\omega}(G(V^i)) = \tilde{\chi}(G(V^i)) = |V^i|$. From $\omega_B = \max\{\tilde{\omega}(G(V^i))| i = 1, \ldots, k\}$ and $\tilde{\chi}_0 = \tilde{\chi}(G) = \max\{\tilde{\chi}(G(V^i))| i = 1, \ldots, k\}$ it follows that $\tilde{\chi}_0 = \omega_B$. Thus no branching is performed, the search tree consists of the original problem $P(U_0, V_0)$ only. □

With this fact the following new graph classes with polynomial solvable maximum clique problem can be deduced.

Theorem 5: *Let $r \in \mathbb{N}$, $r \geq 3$. If G has no subgraph $K_r - \{e\}$ then depth$(\mathcal{SF}) \leq r - 3$.*

Proof: A graph G is disjoint union of complete graphs iff it contains no subgraph $P_3 = K_3 - \{e\}$. Let $r > 3$ and $G(Cl) \subseteq G$ be a clique with $|Cl| = r - 3$ vertices. $G(\bigcap_{v \in Cl} N(v))$ is P_3-free, otherwise the vertices of Cl together with the vertices of the P_3 would induce a $K_r - \{e\}$. Let $P(U_t, V_t)$ be a node of the search tree with $|U_t| = r - 3$. Every subgraph of a P_3-free graph is P_3-free. With $G(V_t) \subseteq G(\bigcap_{u \in U_t} N(u))$ and Lemma 3 the assertion follows. □

Particularly a polynomial solution method for the class of *diamond*-free graphs, i.e. graphs without subgraphs $K_4 - \{e\}$ is obtained. This has consequences for *line graphs of bipartite graphs*. The *line graph* $L(G)$ of G is a graph whose vertices correspond to the edges of G with two vertices of $L(G)$ being adjacent iff the corresponding edges are adjacent. A graph is *bipartite* if it contains no cycle of odd length C_{2l+1}, $l \geq 1$.

Theorem 6: *For line graphs of bipartite graphs depth$(\mathcal{SF}) \leq 1$ holds*

Proof: Verify that line graphs of bipartite graphs are diamond-free. □

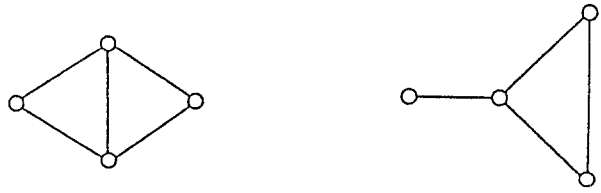

Figure 1

A graph G is called *complete multipartite* if it's vertex set V can be partitioned into non empty sets $V_1, \ldots, V_t$ such that $(v_i, v_j) \in E \Leftrightarrow \exists p, q \in \{1, \ldots, t\}, p \neq q$ with $v_i \in V_p$, $v_j \in V_q$. Obviously $\omega(G) = t$.

Lemma 4.: *If G is complete multipartite then depth$(\mathcal{S}\mathcal{T}) = 0$.*

Proof: Let $v_1, v_2, \ldots, v_s$, $1 \leq s \leq t$, be pairwise adjacent vertices. W.l.o.g. $v_i \in V_i$. Then $\bigcap_{i=1}^{s} N(v_i) = \bigcup_{i=s+1}^{t} V_i$. It follows $\tilde{\omega}(G) = t$ with corresponding clique $G(Cl)$, $Cl = \{v_1, \ldots, v_t\}$. For the coloring c which is produced simultaneously with the clique search $c(v_i) = i$, $i = 1, 2, \ldots, t$, holds. The vertices $v_j, j \notin \{1, \ldots, t\}$, which are not in the clique and not yet colored, have saturation degree $t - 1$, since $v_j \in V_q$ has neighbours of colors $1, \ldots, q-1, q+1, \ldots, t$, but none of color q. If v_j has to be colored it is assigned $c(v_j) = q$. The saturation degrees of the uncolored vertices don't change any longer because the vertices from V_q are not adjacent to v_j and all other vertices already have a neighbour of color q. It follows $\tilde{\chi}(G) = t$, together $\tilde{\omega}(G) = \tilde{\chi}(G)$ and depth$(\mathcal{S}\mathcal{T}) = 0$. □

Theorem 5 dealt with graphs containing no complete graphs of cardinality r minus one edge. A result of the same kind can be obtained for graphs without complete graphs K_r missing two adjacent edges. Those forbidden subgraphs will be denoted by $K_r - P_3$. Obviously $K_3 - P_3 = \bar{P}_3$. Graphs without $K_4 - P_3$ are also known as *paw*-free graphs.

Theorem 7: *Let $r \in \mathbb{N}, r \geq 3$. If G has no subgraph $K_r - P_3$ then depth$(\mathcal{S}\mathcal{T}) \leq r - 3$.*

Proof: A graph is P_3-free iff it is disjoint union of complete graphs. Therefore the set of $\bar{P}_3$-free graphs is identical to the set of complete multipartite graphs. If G has no subgraph $K_r - P_3$ then for every clique $G(Cl)$, $|Cl| = r - 3$, the common neighbourhood of the vertices of Cl is $\bar{P}_3$-free. The theorem follows with Lemma 4. □

Lemma 5: *For bipartite graphs depth$(\mathcal{S}\mathcal{T}) = 0$.*

Proof: Let $G(V^i)$ be a connected component of G with $|V^i| \geq 2$. Then $\tilde{\omega}(G(V^i)) = 2$ with corresponding clique $G(Cl)$, $Cl = \{v_1, v_2\}$. v_1 gets color 1, v_2 color 2. While $G(V^i)$ is not completely colored, there always exists a vertex of saturation degree 1. This vertex can be colored with 1 or 2. Suppose a vertex has saturation degree 2. Then G would contain an odd cycle and would not be bipartite. Therefore $\tilde{\chi}(G(V^i)) = 2$. □

The proof shows that the coloring is optimal for bipartite graphs. Furthermore the well known characterization of this graphs as graphs with chromatic number at most 2 becomes evident. Lemma 5 is useful in regard to the study of planar graphs.

A graph has an *embedding* in the plane if it can be drawn such that no two edges intersect. Graphs with existing embeddings are said to be *planar*. An embedding divides the plane into connected areas, called *faces*. One of them is not bounded. It is called *outer* face. A planar graph is *outerplanar* if it has an embedding where all vertices touch the outer face.

A *series-parallel* graph is a partial subgraph of a 2-tree. The set of *2-trees* is recursively defined in the following way. A K_3 is a 2-tree. If G is a 2-tree and (u, v) an edge of G then the graph which results from G by adding a vertex w and edges (u, w), (v, w) is a 2-tree.

Lemma 6: *If G is series-parallel then depth($\mathcal{ST}$) ≤ 1.*

Proof: A graph H' is said to be a *subdivision* of a graph H if H' is obtained from H by replacing some edges by paths. It is well known [11] that a graph is series-parallel iff it has no subdivision of the K_4 as a partial subgraph.

Consider the *s-wheels* $K_2(K_1, C_s)$, $s \geq 4$ (see Fig. 2(a)). Deleting the edges (u, v_i), $i = 4, \ldots, s$, and replacing the path $(v_3, v_4, \ldots, v_s, v_1)$ by the edge (v_3, v_1) (see (b), (c)) yields a complete graph with four vertices. Thus s-wheels contain a subdivision of the K_4. As a consequence the subgraph induced by the neighbourhood of any vertex in G does not contain a K_3 and a cycle C_s of particularly odd length. $G(N(v))$ is bipartite for all $v \in V$. With Lemma 5 the assertion follows. □

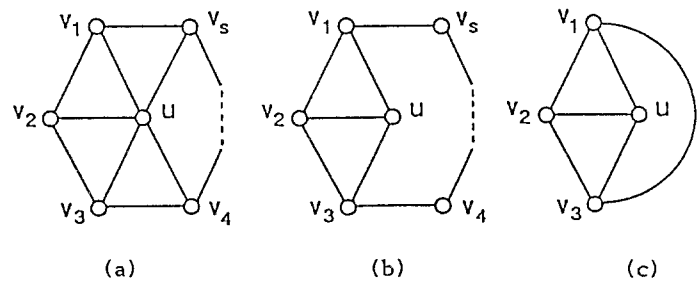

Figure 2

Corollary 1: *If G is outerplanar then depth($\mathcal{ST}$) ≤ 1.*

Proof: Outerplanar graphs are series-parallel [9]. □

Lemma 7: *If G is planar then depth($\mathcal{ST}$) ≤ 2.*

Proof: G is planar iff it has neither a subdivision of the K_5 nor one of the $K_{3,3}$ as a partial subgraph [3]. Similar to the proof of the previous lemma it can be shown that the graphs $K_2(K_2, C_s)$, $s \geq 4$, contain a subdivision of the K_5 (replace vertex u by two adjacent vertices u_1 and u_2). This implies that $G(N(v_i) \cap N(v_j))$ is bipartite for all pairs of adjacent vertices v_i, v_j. □

As both K_5 and $K_{3,3}$ contain subdivisions of K_4, series-parallel graphs are planar [9]. For planar graphs the well known Eulerian formula $m \leq 3n - 6$ holds. Thus according to Theorem 4 the complexity of algorithm BB1 for series-parallel and outerplanar graphs is at most $O(n^2)$, for planar graphs $O(n^3)$. However this results can be improved substantially.

The computation of bounds for the original problem $P(U_0, V_0)$ requires time $O(m + n) = O(n)$. The effort for all problems of distance 1 is of $O(|N(v_1)|) + \cdots +$

$O(|N(v_n)|) = O(n)$, since the graph $G(V_t)$ belonging to $P(U_t, V_t)$, $U_t = \{v_i\}$, has at most $|N(v_i)|$ vertices and due to planarity $O(|N(v_i)|)$ edges. Replacing $P(U_0, V_0)$ by $P(U_t, V_t)$ shows that all successors of $P(U_t, V_t)$ are treated in time $O(|N(v_i)|)$ which summed over all problems of distance 1 gives $O(n)$. It follows:

Theorem 8: *Algorithm* BB1 *solves the maximum clique problem for planar graphs in* $O(n)$.

Hence the algorithm is optimal concerning complexity. Another $O(n)$-method is known in literature [34] which lists all maximal cliques and looks for a largest among them.

In all results for algorithm BB1 achieved so far any of the two branching rules could be applied. For the remainder of subsection 4.1 we restrict to rule I.

Graphs containing no subgraphs C_s, $s \geq 4$, are called *chordal*. They can also be characterized by the existence of a *perfect elimination scheme*, i.e. an ordering $(v_1, v_2, \ldots, v_n)$ of the vertices such that the subgraphs induced by $N(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ are complete [6]. Let $(v_n, v_{n-1}, \ldots, v_1)$ be the order in which the vertices of a chordal graph are colored by procedure *Bounds1*.

Proposition: *Given* $v_i, v_j, v_p \in V, i > j > p$, *with* $(v_i, v_p) \in E$ *and* $(v_i, v_j) \notin E$, *there exists a* $v_q \in V, q > j$, *with* $(v_j, v_q) \in E, (v_p, v_q) \notin E$.

Proof: Proceed from the situation $c(v_r) \neq 0$, $j < r \leq n$, and $c(v_r) = 0$, $1 \leq r \leq j$. v_i is already colored, v_p not yet and v_j is the vertex colored next. If $\text{satdeg}(v_j) > \text{satdeg}(v_p)$ then a $v_q \in N(v_j)$ exists with a color $c(v_q)$ that does not occur in $N(v_p)$. Consequently $v_q \notin N(v_p)$. If $\text{satdeg}(v_j) = \text{satdeg}(v_p)$ then due to the tie breaking rule $\text{cdeg}(v_j) \geq \text{cdeg}(v_p)$. Since $v_i \in N(v_p)$ and $v_i \notin N(v_j)$ there is a vertex v_q, $c(v_q) \neq 0$, which is adjacent to v_j but not to v_p. □

Lemma 8: *For a chordal graph the reverse coloring order* $(v_1, \ldots, v_n)$ *is a perfect elimination scheme.*

Proof: Considering the proposition it can be argumented in an analogous way as in the proof of Theorem 4.8 in [8] where a perfect elimination scheme is constructed by a lexicographic breadth first search. □

It is obvious that the proposition remains true if the graph is colored by *Bounds2*. Thus Lemma 8 holds for both bounding procedures.

Given $P(U_0, V_0)$ the branching rule I produces subproblems $P(U_t, V_t)$ where $U_t = \{v_t\}$ and $G(V_t)$, $t = 1, \ldots, n$, are complete graphs. This shows:

Theorem 9: *For chordal graphs* $\text{depth}(\mathcal{SF}) \leq 1$ *if branching rule* I *is applied.*

The complexity of BB1 for this important class of graphs is $O(m \cdot n + n^2)$ whereas the best known algorithm is of $O(m + n)$ [35]. Here it should be mentioned that the presented solution method does not always reach the complexity of eventually existing special algorithms. This indeed cannot be demanded because we don't have a method that is constructed for one special class. Rather it shall solve the problem uniformly fast for all possible graphs.

Given a family of intervals of the real line a graph whose vertices represent this intervals with two vertices being adjacent iff the corresponding intervals intersect is called an *interval graph.*

Corollary 2: *If G is an interval graph then depth$(\mathcal{ST}) \leq 1$ with branching rule* I.

Proof: Interval graphs are chordal [8]. □

G is a *split graph* if there exists a partition of the vertex set V into sets V_1, V_2, $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, such that V_1 induces a clique and V_2 is independent.

Corollary 3: *If G is a split graph then depth$(\mathcal{ST}) \leq 1$ with branching rule* I.

Proof: Split graphs can be characterized as graphs without subgraphs C_s, $\bar{C}_s$, $s \geq 4$ [8]. Thus they are chordal. □

We now introduce an extension of the concept of chordality. For fixed $r \in \mathbb{N}$ a graph G is defined to be *r-chordal* if the subgraph induced by the common neighbourhood of any r pairwise adjacent vertices is chordal, or equivalently, if G is $K_2(K_r, C_s)$-free for all $s \geq 4$. Obviously a chordal graph is r-chordal for every r. The 1-chordal graphs are identical to the graphs without wheels. Examples for 1- and 2-chordal graphs are series-parallel and planar graphs (see proofs of Lemma 6 and 7). Now Theorem 9 immediately implies:

Theorem 10: *For r-chordal graphs depth$(\mathcal{ST}) \leq r + 1$ if branching rule* I *is applied. Particularly depth$(\mathcal{ST}) \leq 2$ for graphs without wheels.*

A generalization of interval graphs are *circular arc graphs.* Here the vertices represent arcs of a circle. Two vertices are adjacent iff the arcs have nonempty intersection. Every interval graph is a circular arc graph (choose one point to the left and one to the right of all intervals and bend the part of the axis in between to a circle), but the converse is not true (C_4 is a circular arc but not an interval graph). A circular arc graph is called *Helly circular arc graph* if there exists a representation by arcs where no three arcs cover the whole circle.

Theorem 11: *Helly circular arc graphs are 1-chordal.*

Proof: Let $\{A_v | v \in V\}$ be the set of arcs in a representation with the demanded property. For any $v \in V$ there is a point P_v on the circle with $P_v \notin \bigcup_{u \in N(v)} A_u$. If the circle is cut in this point and rolled out on the real axis an interval representation for $G(N(v))$ results. This shows that $G(N(v))$ is an interval graph and consequently a chordal graph. □

Theorem 12: *Line graphs are 2-chordal.*

Proof: It is easy to verify that in a line graph the common neighbourhood of any pair of adjacent vertices induces a C_s-free graph, $s \geq 4$. □

4.2. Extended Clique Search

The versions BB1 and BB2 of algorithm BB differ only in the tie breaking rule used during the clique search in procedures *Bounds1/2*. Since this rule has been of no

importance in the past subsection (except Lemma 8) all results obtained for BB1 also hold for BB2. If extended clique search is applied then the algorithm is polynomial for some further graph classes.

Connected graphs without subgraphs C_s, $s \geq 3$, are called *trees*. A *forest* is the disjoint union of trees. Since forests are bipartite with Lemma 5 $\text{depth}(\mathcal{S}\mathcal{T}) = 0$ holds.

Lemma 9: *If G is the complement of a forest then $\text{depth}(\mathcal{S}\mathcal{T}) = 0$.*

Proof: Let v_1 be a vertex of maximal degree in G. Since forests contain vertices of degree at most 1, $\deg(v_1) \in \{n-2, n-1\}$ holds. As a consequence v_1 is a vertex of a maximum clique $G(MCl)$ in G. Let $\{v_1, \ldots, v_l\} \subseteq MCl$. $G_l := G(\bigcap_{i=1}^{l} N(v_i))$ is the complement of a forest. If v_{l+1} has maximal degree in G_l then by the same argument as above v_{l+1} is contained in a maximum clique of G_l. This shows that the clique constructed by the extended clique search is not only maximal but maximum and $\tilde{\omega}(G) = \omega(G)$.

It remains to proof that $\tilde{\chi}(G) = \omega(G)$. The vertices $v_1, v_2, \ldots, v_{\omega(G)}$ of the maximum clique $G(MCl)$ are colored 1, 2, ..., $\omega(G)$. Label the remaining vertices by $v_{\omega(G)+1}, \ldots, v_n$ such that first the vertex not adjacent to v_1 appears (if it exists) then the only left vertex not adjacent to v_2 (if it exists) and so on. Due to maximality of MCl all vertices of V are captured. For $\omega(G) < i \leq n$ let $v_{k(i)} \in MCl$ be the vertex with smallest index not adjacent to v_i. If $i < j$ then $k(i) < k(j)$. Every vertex $v_i \notin MCl$ is assigned a color $c(v_i) \geq k(i)$. Suppose v_j is the first vertex with a color greater than $k(j)$. In the moment of being colored v_j is adjacent to a vertex $v_l \notin MCl$, $c(v_l) = k(j)$. From $c(v_l) \geq k(l)$ and the choice of v_j follows $k(l) = k(j)$, a contradiction, since $j \neq l$. Therefore every vertex v_i, $i > \omega(G)$, is colored $k(i)$. The total number of colors used in G is $\omega(G)$. □

An immediate consequence of this lemma is:

Theorem 13: *Let $r \in \mathbb{N}$. If a graph G has no subgraph $K_2(K_r, \bar{C}_s)$ for all $s \geq 3$ then $\text{depth}(\mathcal{S}\mathcal{T}) \leq r$.*

In the previous subsection it was stated that line graphs are 2-chordal. Thus according to Theorem 10 $\text{depth}(\mathcal{S}\mathcal{T}) \leq 3$ if branching rule I is applied. For algorithm BB2 with any of the two branching rules a sharper result can be achieved.

Theorem 14: *For line graphs $\text{depth}(\mathcal{S}\mathcal{T}) \leq 2$ holds.*

Proof: The subgraph induced by the common neighbourhood of any two adjacent vertices is $\bar{C}_s$-free, $s \geq 3$. This can easily be varified for $s = 3, 4$. Every $\bar{C}_s$, $s \geq 6$, contains a C_4, furthermore $\bar{C}_5 = C_5$. Since line graphs are 2-chordal it is also true for $s \geq 5$. The assertion follows with Theorem 13. □

For split graphs too, the estimation of the size of the search tree given in Corollary 3, $\text{depth}(\mathcal{S}\mathcal{T}) \leq 1$, is valid only for branching rule I. In the following we show that using extended clique search no branching is performed, thus yielding an $O(m+n)$-algorithm.

Theorem 15: *If G is a split graph then $\text{depth}(\mathcal{S}\mathcal{T}) = 0$.*

Proof: Let V_1, V_2 be a partition of V such that V_1 induce a clique and V_2 is independent. First we show that for every vertex v of maximal degree there exists a maximum clique containing v (since subgraphs of split graphs are split graphs again this implies $\tilde{\omega}(G) = \omega(G)$). Suppose this is not true. Then for every maximum clique $G(MCl)$ there are vertices $u_1, u_2 \in MCl$ with $u_1, u_2 \notin N(v)$. Due to the partition $v \in V_2$ and $u_i \in V_1$ for at least one $i \in \{1,2\}$ holds. Let $u_1 \in V_1$. Since $N(v) \subseteq V_1$ and $N(u_1) \supseteq V_1 - \{u_1\}$ it is $N(v) \subseteq N(u_1)$. u_2 is adjacent to u_1 but not to v, therefore $N(v) \subset N(u_1)$ in contradiction to the choice of v.

Consider now the coloring. The first $\omega(G)$ vertices induce a maximum clique and are colored $1, 2, \ldots, \omega(G)$. We have to check whether one of the remaining vertices may be assigned a color greater than $\omega(G)$. Let u be any vertex of V_2 with $u \notin MCl$. Since $N(u) \subset V_1$ the degree of u is smaller than $\omega(G)$. Hence one color from $\{1, 2, \ldots, \omega(G)\}$ is free. There exists at most one vertex $v \in V_1$ which is not in MCl. If v has no neighbours in V_2 then $\deg(v) < \omega(G)$ and the situation is the same as for u. Otherwise let $w \in N(v) \cap V_2$. Obviously $w \notin MCl$. Suppose w is colored before v. Then $\text{satdeg}(w) \geq \text{satdeg}(v) = \omega(G) - 1$ holds at the moment w is colored, implying $N(w) = V_1$. This is not possible because $V_1 \cup \{w\}$ would induce a clique with more than $|MCl|$ vertices. Consequently v is colored prior to all it's neighbours in V_2 and is assigned a color not greater than $\omega(G)$. It follows $\tilde{\chi}(G) = \omega(G)$. □

We are now turning to the class of P_4-free graphs, also known as *cographs*. This graphs (as well as for instance chordal and bipartite graphs) are *perfect*, i.e. $\omega(H) = \chi(H)$ for all $H \subseteq G$ holds. It can be shown [12] that every sequential coloring method yields an optimal coloring for P_4-free graphs. In [23] an $O(n)$-solution method for the maximum clique problem is presented. Since it seems not possible to proof polynomiality of algorithm BB we consider P_4-free graphs with additional properties. From them new classes with polynomial solvable maximum clique problem can be deduced.

Lemma 10: *If G is P_4- and S_3-free then $\text{depth}(\mathcal{ST}) = 0$.*

Proof: It suffices to show that every vertex v of maximal degree is in a maximum clique. Then $\tilde{\omega}(G) = \omega(G)$ and with $\tilde{\chi}(G) = \chi(G) = \omega(G)$ for P_4-free graphs the lemma follows.

Suppose the contrary. Let $G(MCl)$ be a maximum clique and $W = V - \{v\} - N(v)$ with $|W| = k$. Then

(i) $W \subseteq MCl$ holds

Otherwise, there is a $w \in W$, $w \notin MCl$. Due to maximality of MCl vertices u_1, $u_2 \in MCl$ exist with $(u_1, v) \notin E$ and $(u_2, w) \notin E$. $S_3 \not\subseteq G$ implies $u_1 \neq u_2$. Since $G(\{u_1, v, w\}) \neq S_3$, $G(\{u_2, v, w\}) \neq S_3$ it is $(u_1, w) \in E$ and $(u_2, v) \in E$. But then v, u_2, u_1, w induce a path P_4.

(ii) A vertex $v^* \in N(v)$ exists with $N(v) \cup N(v^*) \neq V$ and $N(v) - \{v^*\} \neq N(v^*) - \{v\}$. Let $w \in W$. Since $\deg(w) \leq \deg(v)$ there are (beneath v) at least $k - 1$ vertices v_2, $v_3, \ldots, v_k \notin N(w)$. From (i) $G(W)$ is complete, thus $v_i \notin W, i = 2, \ldots, k$. $G(\{v, v_2, \ldots, k\})$ is complete too, otherwise w together with two non adjacent vertices of this graph

induce a S_3. If $N(v) - \{v_i\} = N(v_i) - \{v\}$ for $i = 2, \ldots, k$ then $G(MCl - W \cup \{v, v_2, \ldots, v_k\})$ would be a clique with $|MCl| - |W| + k = |MCl|$ vertices, thus a maximum clique containing v. This proves (ii).

Now let v^* be as in (ii) and $u \notin N(v) \cup N(v^*)$. From $\deg(v) \geq \deg(v^*)$ and $N(v) - \{v^*\} \neq N(v^*) - \{v\}$ follows the existence of a vertex u^* which is adjacent to v but not to v^*. $G(\{v^*, u^*, u\}) \neq S_3$ implies $(u, u^*) \in E$. But then the graph induced by v^*, v, u^*, u is a P_4. Hence the assumption is not true, there is a maximum clique containing v. □

P_4- and S_3-free graphs can also be characterized as graphs whose complements are P_4-free bipartite. This follows from $\bar{P}_4 = P_4$, $\bar{S}_3 = K_3$ and the fact that P_4- and K_3-free graphs possess no odd cycles.

Theorem 16: *Let $r \in \mathbb{N}$. If a graph G has no subgraphs $K_2(K_r, P_4)$ and $K_2(K_r, S_3)$ then $depth(\mathcal{SF}) \leq r$.*

Proof: Straightforward. □

A graph without subgraph $K_{1,3}$ is called *claw*-free. The maximum clique problem is NP-complete for this class of graphs [10]. For that reason it is of some interest to find large subclasses for which the problem is solvable in polynomial time. One such subclass is attained if additionally to $K_{1,3}$ no cycles of odd length and their complements are allowed [29]. The above theorem yields another class for $r = 1$, namely the graphs containing no $K_{1,3}$ and $K_2(K_1, P_4)$ (also known in literature as *gem*).

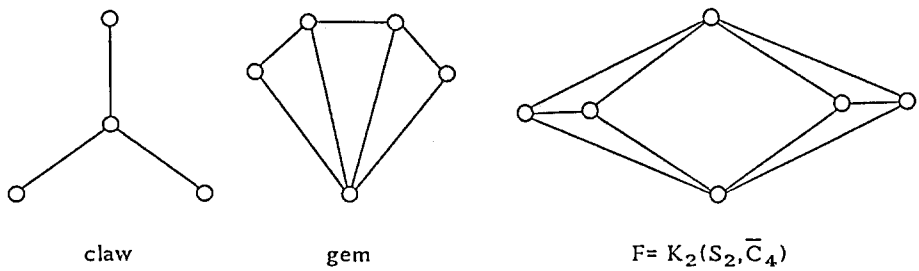

Figure 3

Lemma 11: *If a graph G is P_4- and F-free (see Fig. 3) then $depth(\mathcal{SF}) = 0$.*

Proof: Let G^i be a connected component of G with $\omega(G^i) = \omega(G)$ and v a vertex of maximal degree in G^i. Suppose no maximum clique exists containing v. Then there are vertices v_1, v_2 from a maximum clique $G(MCl) \subseteq G^i$ with v_1, $v_2 \notin N(v)$. G^i is connected and P_4-free. Therefore it contains the subgraph given in Fig. 4(a). Since $\deg(v) \geq \deg(v_1)$ a vertex w_1 exists which is adjacent to v but not to v_1. It is $(w_1, u) \in E$ and $(w_1, v_2) \notin E$, otherwise w_1, v, u, v_1 or v, w_1, v_2, v_1 induce a P_4 (see Fig. 4(b)). Since $\deg(v) \geq \deg(u)$ there is a vertex w_2 with $(w_2, v) \in E$ and $(w_2, u) \notin E$. $G(\{w_2, v, u, v_1\}) \neq$

P_4 and $G(\{w_2, v, u, v_2\}) \neq P_4$ implies the existence of edges (w_2, v_1) and (w_2, v_2). At last $(w_1, w_2) \in E$, otherwise w_2, v_1, u, w_1 induce a P_4. This construction, however, results in the graph F, a contradiction.

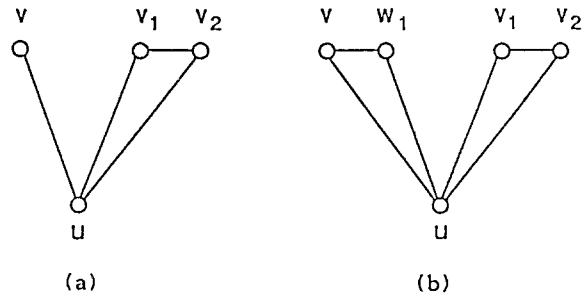

Figure 4

Therefore the maximal clique built in the bounding procedure is maximum, $\tilde{\omega}(G^i) = \tilde{\omega}(G) = \omega(G)$. Since G is P_4-free, $\tilde{\chi}(G) = \omega(G)$ follows and the lemma is proved. □

Theorem 17: *Let $r \in \mathbb{N}$. For graphs without subgraphs $K_2(K_r, P_4)$ and $K_2(K_r, F)$ depth$(\mathcal{S}\mathcal{T}) \leq r$ holds.*

Proof: Straightforward. □

5. Computational Experience

The presented algorithm has been implemented in PASCAL and extensively tested on random graphs of varying cardinality n and edge density p (probability of the existence of an edge). If n is fixed then p can be interpreted as a measure for the difficulty of a problem. With increasing p the 'hardness' grows. In the following tables for each pair (n, p) the average values of running time resp. number of nodes in the search tree of 5 graphs is stated. The problems were run on a CDC Cyber 995.

Table 1. Computational results on random graphs with 100 vertices

edge density p	clique number ω	CPU-time [ms] (number of search tree nodes) BB1(I)	BB1(II)	BB2(I)	BB2(II)
0.1	4	82 (78)	66 (31)	115 (80)	101 (39)
0.2	5	141 (102)	119 (60)	182 (82)	162 (45)
0.3	6	231 (145)	196 (81)	292 (98)	201 (61)
0.4	8	378 (198)	328 (116)	586 (191)	473 (87)
0.5	9	1287 (759)	794 (252)	1588 (606)	1227 (223)
0.6	11–12	3577 (1602)	2031 (542)	5238 (1195)	3431 (477)
0.7	14–15	10478 (3244)	5406 (1018)	11440 (1662)	7904 (750)
0.8	19–21	47737 (9045)	17691 (2192)	90862 (6382)	34733 (1859)
0.9	29–31	197596 (18199)	28401 (1698)	246309 (6707)	79979 (1752)

Table 2. Computational results on random graphs

vertex number n	edge density p	clique number ω	CPU-time [ms] (number of search tree nodes) BALAS-YU		BB	
50	0.1	3	29	(8)	14	(6)
	0.2	4	41	(24)	24	(20)
	0.3	5	57	(36)	30	(20)
	0.4	6	82	(53)	42	(24)
	0.5	7–8	133	(88)	55	(27)
	0.6	8–9	222	(132)	106	(52)
	0.7	11	425	(213)	143	(54)
	0.8	13–15	1228	(512)	185	(51)
	0.9	19–22	3295	(879)	132	(22)
100	0.1	4	120	(38)	66	(31)
	0.2	5	190	(66)	119	(60)
	0.3	6	324	(128)	196	(81)
	0.4	8	630	(254)	328	(116)
	0.5	9	1704	(698)	794	(252)
	0.6	11–12	5069	(1744)	2031	(542)
	0.7	14–15	18987	(5019)	5406	(1018)
	0.8	19–21	152089	(29363)	17691	(2192)
	0.9	29–31	1726262	(175148)	28401	(1698)
200	0.1	4–5	536	(111)	324	(111)
	0.2	6	1079	(291)	669	(192)
	0.3	7–8	2702	(872)	1623	(526)
	0.4	9–10	9027	(2738)	4779	(1241)
	0.5	11	38079	(10567)	17318	(3721)
	0.6	13–14	232199	(51895)	87325	(15747)
	0.7	18–19	2279269	(367230)	630097	(79734)
	0.8	24	—		10761173	(893158)
300	0.1	5	1363	(238)	880	(196)
	0.2	6	3489	(970)	2130	(535)
	0.3	8	12415	(3736)	6991	(1953)
	0.4	10	54650	(14655)	28492	(6307)
	0.5	12–13	353138	(81258)	152195	(29256)
	0.6	15–16	3499705	(608089)	1242186	(183454)
	0.7	21	—		16221377	(1521542)
400	0.1	5	2707	(378)	1743	(295)
	0.2	6–7	9011	(2677)	5234	(1462)
	0.3	8	38864	(10900)	22791	(5662)
	0.4	10	219499	(53682)	113352	(24606)
	0.5	12–13	2016208	(443262)	831199	(141892)
	0.6	16	—		10043446	(1333978)
500	0.1	5	4663	(565)	2998	(415)
	0.2	7	18028	(4603)	10568	(2740)
	0.3	8–9	95838	(24824)	57560	(13015)
	0.4	10–11	699648	(167462)	327797	(66494)
	0.5	14	6234793	(971992)	3082303	(476392)

Table 1 gives results for the versions BB1 and BB2 of algorithm BB provided with both branching rule I and II. The applied node selection rule is depth first search. Compared to BB1 the version BB2 generally yields a considerable reduction of the number of nodes in the search tree. However the time additionally spent to compute the vertex degrees (which is necessary to perform extended clique search) outweighs this advantage. The running time is higher to some extent. Branching rule II is clearly superior to rule I both concerning the size of the search tree and the running time. In all BB1 together with rule II seems to be the most efficient version. This version (in Table 2 denoted by BB for short) was compared to the fastest algorithm known so far which is due to Balas and Yu [15]. The main idea of their method is to find large subgraphs (they choose chordal graphs) where the problem is polynomially solvable. Table 2 states the performances for graphs with 50–500 vertices (if the vertex number is larger then only sparse graphs can be treated, but these problems are not the most interesting ones). For running times greater than $3 \cdot 10^6$ ms we contented with one example. If no entry is made the time limit of $2.2 \cdot 10^7$ ms was exceeded. Among the examples there was none where the running time of *BB* is higher than that of BALAS-YU. The quotient of the running times grows from 1.5 for sparse graphs to about 60 for dense graphs. Similar results hold for the size of the search trees. Thus especially for difficult problems a remarkable efficiency improvement is achieved.

References

[1] Babel, L.: Ein Branch and Bound–Verfahren zur Lösung des Maximum Clique Problems. Dissertation, TU München (1990).

[2] Babel, L., Tinhofer, G.: A branch and bound algorithm for the maximum clique problem. ZOR—Methods and Models of Operations Research *34*, 207–217 (1990).

[3] Berge, C.: Graphs and hypergraphs, Amsterdam: North Holland (1973).

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

[5] Deo, N.: Graph theory with applications to engineering and computer science. New York: Prentice Hall (1974).

[6] Fulkerson, D. R., Gross, O. A.: Incidence matrices and interval graphs. Pacific J. Math. *15*, 835–855 (1965).

[7] Garey, M. R., Johnson, D. S.: Computers and intractability. Freeman, N. Y. (1979).

[8] Golumbic, M. C.: Algorithmic graph theory and perfect graphs. New York: Academic Press (1980).

[9] Johnson, D. S.: The NP-completeness column: an ongoing guide. J Algorithms *6*, 434–451 (1985).

[10] Johnson, D. S.: The NP-completeness column: an ongoing guide. J Algorithms *8*, 438–448 (1987).

[11] Wald, J. A., Colbourn, C. J.: Steiner trees, partial 2-trees, and minimum IFI networks. Networks *13*, 159–167 (1983).

[12] de Werra, D.: Heuristics for graph colorings. In: Tinhofer, G., (ed.) Computational graph theory. Computing Suppl 7, 191–208. Wien New York: Springer (1990).

[13] Augustson, J. G., Minker, J.: An analysis of some graph theoretical cluster techniques. J. of the ACM *17*, 571–588 (1970).

[14] Balas, E., Samuelsson, H.: A node covering algorithm. Naval Res. Log. Quart. *24*, 213–233 (1977).

[15] Balas, E., Yu, C. S.: Finding a maximum clique in an arbitrary graph. SIAM J. Computing *15*, 1054–1068 (1986).

[16] Bron, C., Kerbosch, J.: Finding all cliques of an undirected graph. Comm. of the ACM *16*, 575–577 (1973).

[17] Friden, C., Hertz, A., de Werra, D.: TABARIS: an exact algorithm based on tabu search for finding a maximum independent set in a graph. Preprint, ORWP 3 (1989).

[18] Gerhards, L., Lindenberg, W.: Clique detection for nondirected graphs: Two new algorithms. Computing *21*, 295–322 (1979).

[19] Loukakis, E.: A new backtracking algorithm for generating the family of maximal independent sets of a graph. Comp. Math. Appl. *9*, 583–589 (1983).
[20] Loukakis, E., Tsouros, C.: Determining the number of internal stability of a graph. Intern. J. Comp. Math. *11*, 207–220 (1982).
[21] Nemhauser, G. L., Trotter, L. E.: Vertex packings: structural properties and algorithms. Math. Programming *8*, 232–248 (1975).
[22] Tarjan, R. E., Trojanowski, A. E.: Finding a maximum independent set. SIAM J. Computing *6*, 537–546 (1977).
[23] Corneil, D. G., Perl, Y., Stewart, L. K.: A linear recognition algorithm for cographs. SIAM J. Computing *14*, 929–934 (1985).
[24] Gavril, F.: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM J. Computing *1*, 180–187 (1972).
[25] Gavril, F.: Algorithms for a maximum clique and maximum independent set of a circle graph. Networks *3*, 261–273 (1973).
[26] Gavril, F.: Algorithms on circular-arc-graphs. Networks *4*, 357–369 (1974).
[27] Golumbic, M. C., Hammer, P. L.: Stability in circular-arc-graphs. J Algorithms *9*, 314–320 (1988).
[28] Gupta, U., Lee, D., Leung, J.: Efficient algorithms for interval graphs and circular-arc-graphs. Networks *12*, 459–467 (1982).
[29] Hsu, W.-L., Ikura, Y., Nemhauser, G. L.: A polynomial algorithm for maximum weighted vertex packings on graphs without long odd cycles. Mathematical Programming *20*, 225–232 (1981).
[30] Hsu, W.-L., Nemhauser, G. L.: Algorithms for maximum weight cliques, minimum weighted clique covers and minimum colorings of claw-free perfect graphs. Ann Discrete Math. *21*, 357–369 (1984).
[31] Masuda, S., Nakajima, K.: An optimal algorithm for finding a maximum independent set of a circular-arc-graph. SIAM J. Computing *17*, 41–52 (1988).
[32] Masuda, S., Nakajima, K., Kashiwabara, T., Fujisawa, T.: Efficient algorithms for finding maximum cliques of an overlap graph. Networks *20*, 157–171 (1990).
[33] Minty, G. J.: On maximal independent sets of vertices in claw-free graphs. J. Comb. Theory B *28*, 284–304 (1980).
[34] Papadimitriou, C. H., Yannakakis, M.: The clique problem for planar graphs. Inform. Proc. Letters *13*, 131–133 (1981).
[35] Rose, D. J., Tarjan, R. E., Lueker, G. S.: Algorithmic aspects of vertex elimination on graphs. SIAM J: Computing *5*, 266–283 (1976).
[36] Rotem, D., Urrutia, J.: Finding maximum cliques in circle graphs. Networks *11*, 269–278 (1981).

Luitpold Babel
Mathematisches Institut
Technische Universität München
Arcisstraße 21
D-W-8000 München 2
Federal Republic of Germany