An Exact Algorithm for the Maximum Cardinality Stable Set Problem *

Carlo Mannino (mannino@iasi.rm.cnr.it)
Dipartimento di Statistica, Probabilità e Statistica Applicata,
Università di Roma La Sapienza, Piazzale Aldo Moro 5, 00185 Roma.

Antonio Sassano
Dipartimento di Informatica e Sistemistica,
Università di Roma La Sapienza, Via Buonarroti 12, 00185 Roma

April 1992, IASI-CNR Report n. 334

Abstract

We describe a new branch-and-bound algorithm for the exact solution of the maximum cardinality stable set problem. The bounding phase is based on a variation of the standard greedy algorithm for finding a colouring of a graph. Two different node-fixing heuristics are also described. Computational tests on random and structured graphs and very large graphs corresponding to "real-life" problems show that the algorithm is competitive with the fastest algorithms known so far.

1 Introduction

We denote by $G = (V, E)$ an undirected graph. V is the set of nodes and E the set of edges. A *stable set* is a subset of V such that no two nodes of the subset are pairwise adjacent. The cardinality of a maximum stable set of G will be denoted by $\alpha(G)$. A *clique* is a subset of V with the property that all the nodes are pairwise adjacent. A *clique covering* is a set of disjoint cliques whose union is equal to V; the cardinality of a minimum clique covering is denoted by $\theta(G)$, and since at most one node of a clique can be contained in

*Submitted to : *NETWORKS*, May 1992

a stable set, the relation $\theta(G) \geq \alpha(G)$ holds. The *complement* of $G = (V, E)$ is a graph $\bar{G} = (\bar{V}, \bar{E})$ such that $\bar{V} = V$ and $(u, v) \in \bar{E}$ iff $(u, v) \notin E$. The *neighbourhood* $N(v)$ of $v \in V$ is the set of all nodes adjacent to v in G. Let $W \subseteq V$. The subgraph $G_W = (W, E_W)$ *induced* by W in G is the graph obtained from G by eliminating all the nodes in $V - W$ and all the edges incident with them. The *Maximum Stable Set Problem* (MSS) consists in finding a stable set in G of maximum cardinality $\alpha(G)$.

Note that if S is a stable set of G, then S is a clique of $\bar{G}$. So, MSS is trivially equivalent to finding a maximum cardinality clique of $\bar{G}$. MSS is known to be NP-hard [11] for arbitrary graphs, while is polynomially solvable for special classes of graphs. This is the case of perfect graphs and t-perfect graphs [14], circle graphs and their complements [12], circular arc graphs and their complements [13, 15], claw-free graphs [17], graphs without long odd cycles [16]. Even though there is little hope to find a polynomial time algorithm for arbitrary graphs, many efforts have been done to implement efficient algorithms for MSS. The standard approach is to use branch-and-bound, trying to obtain the best compromise between the quality of the upper bound and the time spent to compute it in each subproblem. Carraghan and Pardalos [8] describe a partially enumerative algorithm, where the time spent for computing an upper bound is negligible and hence the quality of the bound is poor. More time is spent to produce a better quality bound in the works of Balas and Yu [3], of Babel and Tinhofer [1] [2] and of Friden, Hertz and de Werra [9]. The last algorithm is based on Tabu Search. All these algorithms (excluded [9]) are designed for computing a maximum cardinality clique; due to the aforementioned equivalence, we will refer to them as algorithms for solving a MSS. Except for [8], they all compute upper bounds for $\alpha(G)$ by finding a clique covering $\mathcal{C}$ of G and applying the relation $|\mathcal{C}| \geq \theta(G) \geq \alpha(G)$.

A different approach based on polyhedral theory is used by Nehmauser and Sigismondi in their cutting-plane algorithm [18].

The algorithm we propose is a branch-and-bound algorithm, whose main features are the following:

- A tighter approximation for $\alpha(G)$ based on an extension of the definition of clique covering.
- Two different criteria for fixing nodes "out of" or "in" the stable set.
- Extensive testing. In addition to standard testing over a class of random graphs, the algorithm has been tested on graphs arising from

istances of real problems and on graphs with special structure.

Computational testing has shown that the algorithm favourably compares with all the other algorithms described in the literature, and, in addition, has a robust behaviour on different classes of test problems.

2 Bounding Heuristic

A clique covering is a set of cliques of $G = (V, E)$ with the property that each node of G is contained in exactly one clique. We extend the definition by relaxing the last property. A *generalized clique covering* is a set of cliques $\mathcal{C}$ of G (not necessarily disinct) such that each node is contained in one or more cliques. In the following we denote by c_i the number of cliques containing node $v_i \in V$, and by $\bar{c}$ the smallest c_i, for all i. Clearly a clique covering is a special case of generalized clique covering, where c_i is equal to one for all i.

Theorem 1 *Let $G = (V, E)$ and $\mathcal{C} = \{K_1, \ldots, K_q\}$ be a generalized clique covering of G. Then $\lfloor |\mathcal{C}|/\bar{c} \rfloor \geq \alpha(G)$.*

Proof: Suppose there exists a stable set S such that $|S| \geq \lfloor |\mathcal{C}|/\bar{c} \rfloor + 1$. Since each node of S is contained in at least $\bar{c}$ cliques of $\mathcal{C}$ and $|K_i \cap S| \leq 1$ for each $K_i \in \mathcal{C}$, it follows that $|\mathcal{C}| \geq \bar{c}|S|$. Hence $|\mathcal{C}| \geq \bar{c}(\lfloor |\mathcal{C}|/\bar{c} \rfloor + 1) > \bar{c}(|\mathcal{C}|/\bar{c}) = |\mathcal{C}|$, a contradiction. □

Notice that if $\bar{c} = 1$ the bound corresponds to the one obtained by a clique covering.

The bounding procedure HEUTHETA generates a sequence of generalized clique covering $\mathcal{C}^1, \ldots, \mathcal{C}^\Delta$ with the property that each node of V is contained in exactly t cliques of $\mathcal{C}^t$, for $t = 1, \ldots, \Delta$. By Theorem 1 the value of the upper bound is given by $UB = \min \lfloor \mathcal{C}/t \rfloor$, $t = 1, \ldots, \Delta$. The problem of finding a generalized clique covering is made difficult by the fact that the graph G has a huge number of cliques. It follows that the crucial phase of the bounding procedure is that of generating only a promising family of cliques.

This is done by HEUTHETA in the following way. At the t-th iteration HEUTHETA finds a set of cliques $\mathcal{C}^t$ such that each node is contained in exactly t cliques of $\mathcal{C}^t$. The cliques of $\mathcal{C}^1$ are obtained by generating a clique covering of G and this can be done by any method (greedy or sofisticated as in [2, 3, 9]).

The cliques of $\mathcal{C}^t$, for $t = 2, \ldots, \Delta$, are obtained by exentending cliques of $\mathcal{C}^{t-1}$ and by generating new cliques. This is performed in a greedy fashion by selecting in sequence the nodes $v_1, \ldots, v_n$. Let $\mathcal{C}_i^t$ be the set of cliques generated at the t-th iteration after having selected the nodes $v_1, \ldots, v_i$ ($\mathcal{C}_0^t = \mathcal{C}^{t-1}$). When node v_{i+1} is selected, HEUTHETA checks if there exists a clique $K \in \mathcal{C}_i^t$ such that $K' = K \cup \{v_{i+1}\}$ is a clique of G. If K exists then K is replaced by K' producing $\mathcal{C}_{i+1}^t = \mathcal{C}_i^t - \{K\} \cup \{K'\}$. Otherwise the new clique $K' = \{v_{i+1}\}$ is generated and $\mathcal{C}_{i+1}^t$ is set to $\mathcal{C}_i^t \cup \{K'\}$. The generation of cliques in the t-th iteration terminates when all the nodes have been scanned and $\mathcal{C}^t = \mathcal{C}_n^t$.

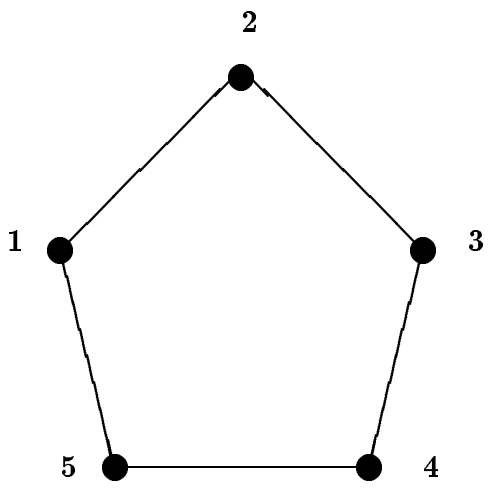

Fig. 1

HEUTHETA

Generate a clique covering $\mathcal{C}^1$. Set $UB = |\mathcal{C}^1|$
for $t = 2$ *to* Δ:
  for $j = 1$ *to* $|V|$
    Select node v_j
    If $\exists K \in \mathcal{C}_{j-1}^t$ such that $K \cup \{v_j\}$ is a clique of G
    Then Set $K' = K \cup \{v_j\}$ and $\mathcal{C}_j^t = \mathcal{C}_{j-1}^t \cup \{K'\} - \{K\}$
    Else Set $K' = \{v_j\}$ and $\mathcal{C}_j^t = \mathcal{C}_{j-1}^t \cup \{K'\}$
  (*end of selection of node* v_j)
  If $UB > \lfloor |\mathcal{C}^t|/t \rfloor$ *Then* $UB = \lfloor |\mathcal{C}^t|/t \rfloor$
(*end of* t*-th iteration*)

Table 1: Variation of UB(Δ)

Δ	UB(Δ)
1	23
2	21
3	20
4	19
7	18
22	17

Consider for example the simple graph of Fig. 2 (where $\alpha(G) = 2$), and suppose $\Delta = 2$. At the first iteration the algorithm produces a clique covering $\mathcal{C}^1 = \{(1,2),(3,4),(5)\}$. This gives $UB^1 = 3$, which is indeed the best possible value for an upper bound produced by any heuristic based on clique covering. When node 1 is selected again we have $\mathcal{C}_1^2 = \{(1,2),(3,4),(5,\mathbf{1})\}$, where the clique $K_3 = (5)$ of $C_0^2 = C^1$ has been replaced by $K_3' = (5,1)$. When node 2 is selected we have $\mathcal{C}_2^2 = \{(1,2),(3,4),(5,\mathbf{1}),(\mathbf{2})\}$ and at the end of the second iteration we obtain $\mathcal{C}^2 = \{(1,2),(3,4),(5,\mathbf{1}),(\mathbf{2},\mathbf{3}),(\mathbf{4},\mathbf{5})\}$. Every node is contained in exactly two cliques of $\mathcal{C}^2$. The value of UB^2 is $\lfloor |\mathcal{C}^2|/2 \rfloor = 2$ and $UB = UB^2 < UB^1$.

In table 1 we show the variation of the upper bound UB(Δ) for increasing values of Δ (the number of iterations of HEUTHETA), when HEUTHETA is applied to a graph with 40 nodes. When Δ is small, UB(Δ) decreases very quickly. UB(Δ) decreases more slowly for higher values of Δ. The limit $\alpha(G) = 17$ is reached when $\Delta = 22$.

3 Enumeration Scheme

In this section we describe the structure of the enumeration scheme and some simple rules which are useful in reducing the size of the branching tree.

Let $G = (V, E)$ be a graph, $V = \{v_1, \ldots, v_n\}$ and let $V_i = \{v_1, \ldots, v_{i-1}\} - N(v_i)$, $i = 1, \ldots, n$. $G_i = (V_i, E_i)$ will then be the subgraph of G induced by V_i and $S^*(G)$ will denote an optimum stable set of G. Obviously $|S^*(G)| = \alpha(G)$. The following simple observation motivates our enumeration scheme.

Fact 1 *For any stable set S of G, $\exists i$ such that $S \subseteq V_i \cup \{v_i\}$ and $v_i \in S$.*

Hence, any stable set S can be thought as the union of $\{v_i\}$ (the node in S with largest index i) with a stable set of G_i. By Fact 1 the problem of finding an optimum stable set of G can be reduced to the solution of $|V|$ optimum stable set problems in the subgraphs $G_1, \ldots, G_n$ induced by $V_1, \ldots, V_n$. Notice that with the i-th subproblem we can associate a set of nodes $I_i = \{v_i\}$ fixed in the stable set and a set $O_i = \{v_{i+1}, \ldots, v_n\} \cup N(v_i)$ of nodes fixed out of the stable set. Moreover, each subproblem has a smaller size than the original problem and can be solved recursively in the same fashion.

In this way it is possible to generate a search tree $\mathcal{T}$, where the *t-th* node $P^t \in \mathcal{T}$ is identified by:

- a set I^t of nodes fixed in the current stable set
- a set O^t of nodes fixed out of the current stable set
- a graph $G^t = (V^t, E^t)$ induced in G by node set $V^t = V - I^t - O^t$.

The subproblems recursively generated by P^t will be denoted by $P_1^t, \ldots, P_{n^t}^t$, $n^t = |V^t|$, where $I_i^t = \{v_i^t\} \cup I^t$ and $O_i^t = O^t \cup N^t(v_i^t) \cup \{v_{i+1}^t, \ldots, v_{n^t}^t\}$, $i = 1, \ldots, n^t$.

Hence Fact 1 provides a systematic way to explore all possible stable sets of G. The resulting search tree $\mathcal{T}$ has a size which grows very fast with $|V|$, becoming soon intractable. In order to limit the growth of $\mathcal{T}$, we use the following fathoming rule.

Let $\hat{S}$ be the best stable set found so far. Let I^t and G^t be, respectively, the set of nodes fixed in the current stable set and the subgraph relative to subproblem P^t. Then there exists a stable set S such that $|S| > |\hat{S}|$ and $I^t \subseteq S$ if and only if $\alpha(G^t) \geq |\hat{S}| - |I^t| + 1 = \bar{\alpha}^t$. In other words, $\bar{\alpha}^t$ is the minimum cardinality of a stable set of G^t which allows to improve the current best solution. If we are able to prove, by the upper bounding heuristic, that $\alpha(G^t) < \bar{\alpha}^t$ then P^t can be fathomed, since every stable set S of G which contains I_t satisfies $|S| \leq |\hat{S}|$.

Another way to limit the growth of the search tree which makes use of some additional informations produced by the bounding heuristic HEUTHETA is described by the following theorem.

Theorem 2 *Let $P^t = (I^t, O^t, \bar{\alpha}^t) \in \mathcal{T}$. Let $\mathcal{C}^1 = \{K_1, \ldots, K_p\}$ be the clique covering generated by the first iteration of HEUTHETA, with $p \geq \bar{\alpha}^t$. Consider $\bar{\alpha}^t - 1$ cliques of $\mathcal{C}^1$, say $K_1, \ldots, K_{\bar{\alpha}^t - 1}$, and let*

$$W = K_1 \cup \ldots \cup K_{\bar{\alpha}^t - 1}. \tag{1}$$

Let the nodes of V^t be ordered in such way that the first $|W|$ nodes coincide with W. Then the first $|W|$ subproblems recursively generated from P^t, and denoted by $P^t_1, \ldots, P^t_{|W|}$, can be fathomed.

Proof: Let $G_W = (W, E_W)$ be the subgraph induced by W. By definition of $W, \mathcal{C}_W = \{K_1, \ldots, K_{\bar{\alpha}^t - 1}\}$ is a clique covering of G_W and then $\alpha(G_W) < \bar{\alpha}^t$. In other words HEUTHETA provides an upper bound for G_W and for all its induced subgraphs which is less than $\bar{\alpha}^t$. Let $S^*(G^t_i)$ be an optimum stable set of G^t_i, by the ordering of the nodes of V^t, we have that $|\{v^t_i\} \cup S^*(G^t_i)| \leq \alpha(G_W) < \bar{\alpha}^t$, $i = 1, \ldots, |W|$ and the subproblems $P^t_1, \ldots, P^t_{|W|}$ can be fathomed. □

In the following we will indicate by $P^t = (I^t, O^t, \bar{\alpha}^t)$ the t-th subproblem, and by S_I the incumbent best solution ($\alpha_I = |S_I|$). The search tree will be represented by a list of not yet solved problems $\mathcal{L}$.

Main algorithm.

Initialization. Set $t = 0$, $G^0 = G$, $I^0 = O^0 = \emptyset$. Find a feasible solution S_f (greedy heuristic) and set $S_I = S_f$. Set $\bar{\alpha}^0 = \alpha_I + 1$. Add $P^0 = (I^0, O^0, \bar{\alpha}^0)$ to the list of unsolved problems $\mathcal{L}$.

1. If $\mathcal{L}$ is empty **STOP**. S_I is an optimum stable set.

2. Extract the first problem $P^t = (I^t, O^t, \bar{\alpha}^t)$ from $\mathcal{L}$.

3. Compute an upper bound UB^t (HEUTHETA). If $UB^t < \bar{\alpha}^t$ go to 1.

4. Let W as in Equation (1). Let $V^t - W = v^t_1, \ldots, v^t_q$.

5. For $j = 1$ to q do

5.1 Set $I^t_j = I^t \cup \{v^t_j\}$. Set $O^t_j = O^t \cup N^t(v^t_j) \cup \{v^t_{j+1}, \ldots, v^t_q\}$.

5.2 If $|I^t_j| > |S_I|$ set $S_I = I^t_j$.

5.3 Set $\bar{\alpha}^t_j = \alpha_I - |I^t_j| + 1$, $V^t_j = V - |I^t_j| - |O^t_j|$.

5.4 If $|V^t_j| \geq \bar{\alpha}^t_j$ add $P^t_j = (I^t_j, O^t_j, \bar{\alpha}^t_j)$ to $\mathcal{L}$.

6. Go to 1.

In table 2 we show the number of problems of the search tree and the corresponding CPU running times, for increasing values of Δ. The algorithm is applied to a graph with 100 nodes and density 10% (where the density

is the number of edges of E over the number of edges in the complete graph with V nodes). For small values of Δ, small increments produce big decrements in the number of problems, while for larger values of Δ the table shows an opposite behaviour. Since most of the time spent by the algorithm is used by the bounding heuristic, increasing values of Δ produce proportional increments in global time, which are opposed by the reduction of the number of subproblems in the search tree.

Table 2: A graph with 100 nodes and density 10%

Δ	Prob	time (sec)
1	1785	5.92
2	1233	4.79
3	972	4.30
4	833	4.11
5	767	4.10
6	707	4.15
7	684	4.20
8	660	4.31
9	635	4.47

4 Node Fixing

Node fixing is a powerful tool to limit the number of solutions which need to be explored for improving the cardinality of the current best stable set.

For a given subproblem of the search tree $\bar{P} = (\bar{I}, \bar{O}, \bar{\alpha})$, let $G = (V, E)$ be the corresponding graph. Our goal is either to find an optimum stable set S^* of G such that $|S^*| \geq \bar{\alpha}$, or to prove that $\alpha(G) < \bar{\alpha}$. In the first case we can update the current best solution, in the second we can fathom $\bar{P}$. The purpose of this section is to describe two procedures which allow us to achieve the above goal by solving a MSS problem on a graph with fewer nodes than G. This is obtained by fixing "in" and "out" of S^* suitable subsets of V. In particular, we will use the concept of "simplicial node" to fix nodes "in" S^* and a slight extension of the procedure HEUTHETA to fix nodes "out" of S^*.

Simplicial nodes. Let S^* be an optimum stable set of G, and $I \subseteq S^*$. Let $W = I \cup N(I)$ and G_W the subgraph induced by $V - W$. Then the following holds:

Fact 2 *$\alpha(G) = \alpha(G_W) + |I|$ and any optimum stable set S^*_W of G_W is such that $S^*_W \cup I$ is an optimum stable set of G.*

By Fact 2, the original goal can be restated as follows: find an optimum stable set S^*_W of G_W or prove that $\alpha(G_W) < \bar{\alpha} - |I|$. We now describe a way to identify a set $I \subseteq V$ such that there exists an optimum stable set S^* of G containing I.

A node v is called *simplicial* if $N(v)$ is a clique. The following is a straightforward property of simplicial nodes:

Fact 3 *Let v be a simplicial node of G. Then there exists an optimum stable set S^* of G such that $v \in S^*$.*

Obviously, if $v \in S^*$, then for all $u \in N(v)$, $u \notin S^*$. Now, let $v_1, \ldots, v_q \in V$, let $V_i = V_{i-1} - \{v_i\} - N(v_i)$ and let $G_i = (V_i, E_i)$ be the subgraph induced by V_i, with $V_0 = V$.

Fact 4 *Suppose v_i simplicial in G_{i-1}, for $i = 1, \ldots, q$. Then there exists an optimum stable set S^* of G such that $I = \{v_1, \ldots, v_q\} \subseteq S^*$.*

The following algorithm finds the set I as defined in Fact 4. By $N(I)$ we denote the neighbourhood of the set I, that is the union of all the neighbourhoods of the nodes belonging to I.

FIND SIMPLICIAL NODES

Set $I = W = \emptyset$.

While $\exists v \in V - W$ such that v is simplicial:

Set $I = I \cup \{v\}$, $W = I \cup N(I)$.

Extending HEUTHETA. Let O be a subset of V such that $O \cap S = \emptyset$ for any stable set S of G such that $|S| \geq \bar{\alpha}$, and let G_O be the subgraph of G induced by the set $V - O$. Since the removal of the set O does not "destroy" any stable set S of G such that $|S| \geq \bar{\alpha}$, we have the following property of the graph G_O:

Fact 5 *If $\alpha(G) \geq \bar{\alpha}$ then $\alpha(G) = \alpha(G_O)$.*

Since G_O is a subgraph of G, if $\alpha(G) \geq \bar{\alpha}$ every optimum stable set of G_O is also an optimum stable set of G. Moreover, $\alpha(G_O) < \bar{\alpha}$ if and only if $\alpha(G) < \bar{\alpha}$.

By Fact 5, the original goal can be restated as follows: find an optimum stable set S_O^* of G_O or prove that $\alpha(G_O) < \bar{\alpha}$. In fact, if $|S_O^*| \geq \bar{\alpha}$ we can update the current optimum. If conversely $\alpha(G_O) < \bar{\alpha}$, we have that $\alpha(G) < \bar{\alpha}$ and the problem $\bar{P}$ can be fathomed.

Next Theorem is the basis for a sligth extension of the algorithm HEUTHETA for computing a set $O \subseteq V$ satisfying the conditions of Fact 5.

Theorem 3 *Let S be a stable set of $G = (V, E)$ such that $|S| \geq \bar{\alpha}$. Let $\mathcal{C}$ be a generalized clique covering of G, with $\bar{c} = \min_i c_i$. Let $v_j \in V$ and let*

- $\mathcal{J}' = \{K | K \in \mathcal{C}$ *such that* $v_j \notin K$ *and* $K \cup \{v_j\}$ *clique of* $G\}$, *and*
- $\mathcal{J}'' = \{K | K \in \mathcal{C}$ *such that* $v_j \in K\}$.

If $\lfloor |\mathcal{C} - \mathcal{J}' - \mathcal{J}''|/\bar{c} \rfloor \leq \bar{\alpha} - 2$ *then* $v_j \notin S$.

Proof: Suppose $v_j \in S$ and let $W = V - \{v_j\} - N(v_j)$. Then there exists a stable set $S_W \subseteq W$ such that $S = \{v_j\} \cup S_W$ and $|S_W| \geq \bar{\alpha} - 1$. If we denote by G_W the subgraph induced by W, then $\alpha(G_W) \geq \bar{\alpha} - 1$. $\mathcal{C} - \mathcal{J}' - \mathcal{J}''$ is a generalized clique covering of G_W such that each node belonging to W is contained in at least $\bar{c}$ cliques. In fact, the cliques of $\mathcal{J}'$ and $\mathcal{J}''$ contain only nodes belonging to $\{v_j\} \cup N(v_j) = V - W$. By Theorem 1 we obtain $\alpha(G_W) \leq \lfloor |\mathcal{C} - \mathcal{J}' - \mathcal{J}''|/\bar{c} \rfloor \leq \bar{\alpha} - 2$, a contradiction. □

We now show how to extend the algorithm HEUTHETA in order to compute the set $\mathcal{J}'$. In the following $\mathcal{C}$ is the family of cliques generated by HEUTHETA, Δ the number of iteration ($\bar{c} = \Delta$) and the set O will denote the nodes fixed out of the stable set. Note that since $|\mathcal{J}''| = \Delta = \bar{c}$, we can limit the search to the set $\mathcal{J}'$, such that $\lfloor |\mathcal{C} - \mathcal{J}'|/\Delta \rfloor \leq \bar{\alpha} - 1$

EXTENSION of HEUTHETA

Set $O = \emptyset$.
for $j = 1$ to $|V|$
  Select node v_j. Let $\mathcal{J}' = \{K | K \in \mathcal{C}$ such that $v_j \notin K$ and $K \cup \{v_j\}$ clique of $G\}$.
  If $\lfloor |\mathcal{C} - \mathcal{J}'|/\Delta \rfloor \leq \bar{\alpha} - 1$ *Then Set* $O = O \cup \{v_j\}$.
(*end of selection of node* v_j)

Example. To illustrate the last fixing procedure, consider again the graph C_5 of Fig. 2 (which can be thought associated with a node $\bar{P}$ of the search tree) and suppose $\bar{\alpha} = 3$ and $\Delta = 1$. For $\Delta = 1$ HEUTHETA applied to C_5 produces a clique covering $\mathcal{C}^1 = \{K_1 = (1,2), K_2 = (3,4), K_3 = (5)\}$, which gives $UB = 3$. Now, if we consider node 1 , then $\mathcal{J}' = \{K_3\}$, and $\lfloor |\mathcal{C} - \mathcal{J}'|/\bar{c} \rfloor = 2 \leq 3 - 1$ and node 1 can be fixed out. The same applies to node 4, and then $O = \{1, 4\}$. We are now able to give the complete description of the MSS algorithm partially introduced in section 3, extended by the fixing procedures.

Main Algorithm.

Initialization. Set $t = 0$, $G^0 = G$, $I^0 = O^0 = \emptyset$. Find a feasible solution S_f (greedy heuristic) and set $S_I = S_f$. Set $\bar{\alpha}^0 = \alpha_I + 1$. Add $P^0 = (I^0, O^0, \bar{\alpha}^0)$ to the list of unsolved problems $\mathcal{L}$.

1. If $\mathcal{L}$ is empty **STOP**. S_I is an optimum stable set.

2. Extract the first problem $P^t = (I^t, O^t, \bar{\alpha}^t)$ from $\mathcal{L}$.

3. Compute an upper bound UB^t (HEUTHETA). If $UB^t < \bar{\alpha}^t$ go to 1.

4. Use the algorithm EXTENSION of HEUTHETA to find a set O of nodes fixed "out". Set $O^t = O^t \cup O$. Set $V^t = V - I^t - O^t$.

5 If $|V^t| < \bar{\alpha}^t$ go to 1.

6. Let W as in Equation (1). Let $V^t - W = v_1^t, \ldots, v_q^t$.

7. For $j = 1$ to q do

7.1 Set $I_j^t = I^t \cup \{v_j^t\}$. Set $O_j^t = O^t \cup N^t(v_j^t) \cup \{v_{j+1}^t, \ldots, v_q^t\}$.

7.2 Use the algorithm FIND SIMPLICIAL NODES to find a set I of nodes fixed "in". Set $I_j^t = I_j^t \cup I$ and $O_j^t = O_j^t \cup N(I)$.

7.3 If $|I_j^t| > |S_I|$ set $S_I = I_j^t$.

7.4 Set $\bar{\alpha}_j^t = \alpha_I - |I_j^t| + 1$, $V_j^t = V - |I_j^t| - |O_j^t|$.

7.5 If $|V_j^t| \geq \bar{\alpha}_j^t$ add $P_j^t = (I_j^t, O_j^t, \bar{\alpha}_j^t)$ to $\mathcal{L}$.

8. Go to 1.

Note that node fixing has two different effects. First, the number of nodes in the search tree decreases. Second, the average time spent to solve each subproblem decreases, as a consequence of the reduced sizes of the subproblems.

5 Computational experience

As mentioned in the introduction, the algorithm has been extensively tested on random graphs, graphs with special structure and graphs corresponding to instances of real problems. Random graphs are the usual test problems for MSS algorithms and have been used by [1, 2, 3, 8, 9]. Nevertheless, our computational experience showed that an algorithm with good performances on random graphs can have very poor performances on structured or "real life" problems. To assess the robustness of our algorithm with respect to the input data, we did a series of experiments with structured (and "real life") problems, which are easy to reproduce. The results are uniformly good. In particular, our algorithm was able to solve in few seconds the infamous covering problems A_{27} and some very large "real life" problems arising from circuit testing. To allow a better comparison with other codes we include also the computing times for these new classes of problems.

The algorithm has been coded in C, and executed on a computer IBM RISC 6000-320.

Except for the case of graphs arising from code generation, the clique covering generated in the first iteration of the bounding heuristic HEUTHETA is found in a greedy fashion, by applying the same steps as in iteration $2, \ldots, \Delta$. Except for the case of graphs arising from circuit testing, the nodes of the graph are ordered by increasing degree, that is if $j > i$, $deg(v_j) \geq deg(v_i)$, where $deg(v) = |N(v)|$.

In all the tables the columns identified by *Nodes* will show the number of nodes of the graph, by $\Delta = x$ the number of problems of the search tree needed for solving an instance when the number of iterations of HEUTHETA (Δ) is equal to x. The time, when given, is expressed in seconds.

Random Graph. For our random graphs we chose the model $\mathcal{G}\{n$,P(edge) $= p\}$ (see [4]) which consists of graphs with node set $V = 1, \ldots, n$ in which the edges are chosen indipendently with probability p. If we define the density of a graph G as the number of edges of $G = (V, E)$ over the number of edges of the complete graph with $|V|$ nodes, then for this class of random graphs the density is very close to p. In Table 3 we compare our algorithm with the fastest algorithm known so far, due to Babel [1]. The number of problems of the search tree is an average of computational results obtained over 25 graphs.

Table 3: Random Graphs

Nodes	p	$\Delta = 5$	$\Delta = 10$	$\Delta = 20$	Ba
100	.1	1580	1282	1156	1698
100	.2	1783	1472	1338	2192
100	.3	827	709	662	1018
100	.4	461	426	403	542
100	.5	198	155	130	252
100	.6	75	63	60	116
100	.7	51	50	50	81
100	.8	48	47	47	60
100	.9	34	31	30	31
200	.2	584171	467891	361620	893158
200	.3	73062	69532	62513	79734
200	.4	15691	13337	11938	15747
200	.5	3310	3008	2896	3721
200	.6	1625	1555	1505	1241
200	.7	356	219	172	526
200	.8	128	127	126	192
200	.9	98	98	98	111
300	.3	2055051	1726808	1568176	1521542
300	.4	176816	151627	138078	183454
300	.5	34680	29161	25993	29256
300	.6	6199	5822	5644	6307
300	.7	2645	2486	2391	1953
300	.8	222	209	208	535
300	.9	185	185	184	196
400	.4	1403653	1180527	1057331	1333978
400	.5	159098	147039	138426	141892
400	.6	19326	15725	14129	24606
400	.7	6407	6284	6152	5662
400	.8	1175	766	543	1462
400	.9	267	267	267	295
500	.5	500155	452937	432533	476392
500	.6	78239	64775	56601	66494
500	.7	12772	12696	12523	13015
500	.8	2804	1923	1451	2740
500	.9	357	354	357	415

Table 4: Code Sequences

d, q, w	Nodes	Dens.	$\Delta = 1$	$\Delta = 5$	$\Delta = 10$	$\Delta = 20$	time
$9, 4, 4$	126	16	51001	28397	21422	17258	414.20
$9, 4, 5$	126	16	42822	20930	14185	10837	323.15
$9, 5, 0$	512	49.9	226064	206651	200446	197219	2831.12

Code Generation. Given two sequences s_1, s_2 of q bits, the *Hamming Distance* $H(s_1, s_2)$ is defined as the number of coordinates in which s_1 and s_2 differ. For example, if $s_1 = 001001$ and $s_2 = 101010$, then the Hamming Distance $H(s_1, s_2) = 3$ since s_1 and s_2 differ in the first, fifth and sixth coordinate. A fundamental problem in coding theory is to find the maximum number of sequences of q bits such that any two sequences have Hamming Distance at least d, where q and d are given parameters. This problem can be easily stated as a MSS on a graph $G = (V, E)$. In fact, if $\mathcal{S}_q$ is the family of all sequences of length q, then $|V| = |\mathcal{S}_q|$, and $v_i, v_j \in V$ are connected by an edge iff $H(s_i, s_j) < d$, where $s_i, s_j \in \mathcal{S}_q$. Clearly, any two nodes in a stable set of G correspond to sequences whose distance is greater than or equal to d.

A third parameter w is often given in applications, where w denotes the number of 1's in a sequence s; in this case $\mathcal{S}_{q,w}$ is the family of all sequences of length q with exactly w 1's and G is built again as described above.

Besides representing instances of real problems, these graphs are standardly generated and can be used for comparisons among different algorithms. Moreover they are computationally difficult.

Computational experience has shown how for this class of graphs it is convenient to use a more sofisticated heuristic for finding a clique covering in the first iteration of HEUTHETA. The one we chose is due to Brelaz [5] and is the base for the bounding in [1, 2]. In table 4, the column identified by d, q, w shows the corresponding values for the parameters, while *Dens.* is the density of the associated graph. The computing time is relative to the case $\Delta = 5$. Observe that we were not able to solve the problems $(q = 8, d = 3, w = 0)$ (the corresponding graph has 216 nodes and density equal to 14.12), $(q = 9, d = 3, w = 0)$ (512 nodes and density 8.81) and $(q = 9, d = 4, w = 0)$ (512 nodes and density 25.24).

Table 5: Steiner Triples

A	Nodes	Dens.	$\Delta = 5$	$\Delta = 10$	$\Delta = 20$	time
A_9	40	7.69	12	9	9	0.01
A_{27}	350	1.02	1933	1779	1710	6.25
A_{45}	1012	0.37	293852	221464	180406	5238

Steiner Triple System. A Steiner Triple System (see [19]) is a set of elements $A = \{a_1, \ldots, a_n\}$ and a family T of subsets $t_i \subseteq A$ called *triples*, such that $|t_i| = 3$ for all i and any couple of distint elements of A is contained in exaclty one triple of T. From a Steiner Triple System we derive a graph $G_S = (V, E)$, with the following rules:

- $V = W \cup U$, where $W = \{w_i : a_i \in A\}$ and $U = u_{ij} : t_i \in T, a_j \in t_i$.
- $E = E_U \cup E_{WU}$, where $E_U = \{(u_{ij}, u_{ik}) : u_{ij}, u_{ik} \in U\}$ and $E_{WU} = \{(w_j, u_{ij}) : w_j \in W \text{ and } u_{ij} \in U\}$.

In other words, the nodes in W represent the elements of A while the nodes in U represent the triples with $|W| = |A|$ and $|U| = 3|T|$. Moreover to each triple t_i corresponds a triangle K_i with nodes in U and each node $w_j \in W$ is connected to the corresponding nodes u_{ij} whenever $a_j \in t_i$.

It is well known that the problem of finding a subset C (called *cover*) of A of minimum cardinality such that $C \cap t_i \neq \emptyset$, for all i, is NP-hard. This problem can be stated as a Set Covering Problem; moreover computational experience has shown how the problem is hard to solve in practice (for A sufficiently large) and in [10] the authors suggest to use it for testing the quality of algorithms for Set Covering. Now, let G_S be the graph above described: it is possible to show that the cardinality of a minimum cover for a Steiner Triple System defined on A is equal to $|V| - 2|T| - \alpha(G_S)$. The size of the problem depends from the size of the set A, and will be indicated by a subscript ($|A_x| = x$). As for the previous class of graphs, these graphs are very easy to generate and can be used for comparisons among algorithms. In table 5, the column identified by A shows the Steiner System from which the instance arises.

Test Generation for Combinatorial Circuit. Finally we show computational results over six graphs which arise from a formulation as MSS

Table 6: Combinatorial Circuits Testing

Circuit	Nodes	Den.	$\Delta = 1$	time
C942.ig9	64	97.32	4	<0.01
C942.ig3	142	96.42	60	0.02
C942.ig0	942	98.24	499	0.35
C1355	1470	99.8	749	0.7
C1908	1864	99.04	1230	1.16
C7552	7542	99.8	3721	9.50

of a problem of testing combinatorial and sequential circuits for faults [6]. Details about this model can be found in [7]. These graphs can have a huge number of nodes and are very dense. To our knowledge these are the largest "real life" instances of MSS problems solved to optimality. In this case the value of the parameter Δ has been set equal to one for all the experiments. The results are shown in table 6.

References

[1] L. Babel. Finding maximum cliques in arbitrary and special graphs. *Computing*, 46/4:321–341, 1991.

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

[3] E. Balas and C.S. Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15:1054–1068, 1986.

[4] Bela Bollobas. *Random Graphs.* Academic Press, 1985.

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

[6] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinatorial benchmark cicruits and a target translator in fortran. *Iscas'85: IEEE Int. Symp. on Circuits and Systems (Kyoto-Japan)*, June 1985.

[7] E. Calia and A. Lioy. Test generation in a distributed enviroment. Technical report, Politecnico di Torino, C.so Duca degli Abbruzzi 24, 10129 Torino-Italy, August 1991.

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

[9] C. Friden, A. Hertz, and D. de Werra. An exact algorititham based on tabu search for finding a maximum indipendent set in graph. *Computers Opns. Res.*, 17(5):375–382, 1990.

[10] D.R. Fulkerson, G.L. Nemhauser, and L.E. Trotter. Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of steiner triple systems. *Mathematical Programming Study*, 2:72–81, 1974.

[11] M.R. Garey and D.S. Johnson. *Computer and Intractability: a Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[12] F. Gavril. Algorithms for a maximum clique and a maximum indipendent set of a circle graph. *Networks*, 3:261–273, 1973.

[13] F. Gavril. Algorithms on circular arc graphs. *Networks*, 4:357–369, 1974.

[14] M. Grotschel, L. Lovasz, and A. Schrijver. Relaxation of vertex packing. *J. of Comb. Theory B*, 40:330–343, 1986.

[15] U. Gupta, D. Lee, and J. Leung. Efficient algorithms for interval graphs and circular arc graphs. *Networks*, 12:459–467, 1982.

[16] W.L. Hsu, Y. Ikura, and G.L. Nemhauser. A polynomial algorithm for maximum weighted vertex packings on graphs without long odd cycles. *Mathematical Programming*, 20:225–232, 1981.

[17] G.J. Minty. On maximal indipendent sets of vertices in claw-free graphs. *J. of Comb. Theory B*, 28:284–304, 1980.

[18] G.L. Nemhauser and G. Sigismondi. A strong cutting plane/branch-and-bound algorithm for node packing. Technical Report J-89-08, Georgia Institute of Technology, Atlanta, Georgia 30332, USA, August 1989.

[19] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley and Sons, Inc., 1988.