A SEQUENTIAL ELIMINATION ALGORITHM FOR COMPUTING BOUNDS ON THE CLIQUE NUMBER OF A GRAPH

Bernard Gendron
Département d'informatique
et de recherche opérationnelle
and
Centre de recherche sur les transports
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal, Québec H3C 3J7

Alain Hertz
Département de mathématiques et de génie industriel
École Polytechnique de Montréal
C.P. 6079, succ. Centre-ville
Montréal, Québec H3C 3A7
and
GERAD
3000, chemin de la Côte-Sainte-Catherine
Montréal, Québec H3T 2A7

Patrick St-Louis
Département d'informatique
et de recherche opérationnelle
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal, Québec H3C 3J7

December 2007

Abstract

We consider the problem of determining the size of a maximum clique in a graph, also known as the clique number. Given any method that computes an upper bound on the clique number of a graph, we present a sequential elimination algorithm which is guaranteed to improve upon that upper bound. Computational experiments on DIMACS instances show that, on average, this algorithm can reduce the gap between the upper bound and the clique number by about 60%. We also show how to use this sequential elimination algorithm to improve the computation of lower bounds on the clique number of a graph.

1 Introduction

In this paper, we consider only undirected graphs with no loops or multiple edges. For a graph G, we denote $V(G)$ its vertex set and $E(G)$ its edge set. The *size* of a graph is its number of vertices. The subgraph of G induced by a subset $V' \subseteq V(G)$ of vertices is the graph with vertex set V' and edge set $\{(u,v) \in E(G) \mid u, v \in V'\}$. A *complete graph* is a graph G such that u and v are adjacent, for each pair $u, v \in V(G)$. A *clique* of G is an induced subgraph that is complete. The *clique number* of a graph G, denoted $\omega(G)$, is the maximum size of a clique of G. Finding $\omega(G)$ is known as the *clique number problem*, while finding a clique of maximum size is the *maximum clique problem*. Both problems are NP-hard [6]. Many algorithms, both heuristic and exact, have been designed to solve the clique number and maximum clique problems, but finding an optimal solution in relatively short computing times is realistic only for small instances. The reader may refer to [3] for a survey on algorithms and bounds for these two problems.

An upper bound on the clique number of a graph is useful in both exact and heuristic algorithms for solving the maximum clique problem. Typically, upper bounds are used to guide the search, prune the search space and prove optimality. One of the most famous upper bounds on the clique number of a graph G is the *chromatic number*, $\chi(G)$, which is the smallest integer k such that a legal k-coloring exists (a legal k-coloring is a function $c : V(G) \rightarrow \{1, 2, \ldots, k\}$ such that $c(u) \neq c(v)$ for all edges $(u, v) \in E(G)$). Finding the chromatic number is known as the *graph coloring problem*, which is NP-hard [6]. Since $\chi(G) \geq \omega(G)$, any heuristic method for solving the graph coloring problem provides an upper bound on the clique number. Other upper bounds on the clique number will be briefly discussed in Section 4 (for an exhaustive comparison between the clique number and other graph invariants, see [1]).

In this paper, we introduce a sequential elimination algorithm which makes use of the *closed neighborhood* $N_G(u)$ of any vertex $u \in V(G)$, defined as the subgraph of G induced by $\{u\} \cup \{v \in V(G) \mid (u, v) \in E(G)\}$. Given an arbitrary upper bound $h(G)$ on $\omega(G)$, the proposed algorithm produces an upper bound $h^*(G)$ based on the computation of $h(N_{G'}(u))$ for a series of subgraphs G' of G and vertices $u \in V(G')$. Under mild assumptions we prove that $\omega(G) \leq h^*(G) \leq h(G)$. As reported in Section 4, our tests on DIMACS instances [9] show that embedding a graph coloring heuristic (i.e., $h(G)$ is an upper bound on $\chi(G)$ produced by a heuristic) within this sequential elimination algorithm reduces the gap between the upper bound and $\omega(G)$ by about 60%.

In Section 2, we present the sequential elimination algorithm in more details and we prove that it provides an upper bound on the clique number of a graph. In Section 3, we discuss how the sequential elimination algorithm can also be used to improve the computation of lower bounds on the clique number. We present computational results on DIMACS instances in Section 4, along with concluding remarks.

2 The Sequential Elimination Algorithm

Assuming $h(G')$ is a function that provides an upper bound on the clique number of any induced subgraph G' of G (including G itself), the sequential elimination algorithm is based on computing this upper bound for a series of subgraphs G' of G. By computing this upper bound for the closed neighborhood $N_G(u)$ of each vertex $u \in V(G)$, one can easily get a new upper bound $h^1(G)$ on $\omega(G)$, as shown by the following proposition.

Proposition 1 $\omega(G) \leq \max_{u \in V(G)} h(N_G(u)) \equiv h^1(G)$.

Proof: Let v be a vertex in a clique of maximum size of G. This implies that $N_G(v)$ contains a clique of size $\omega(G)$. Hence, $\omega(G) = \omega(N_G(v)) \leq h(N_G(v)) \leq \max_{u \in V(G)} h(N_G(u))$. ■

Now let s be any vertex in $V(G)$, and let G_s denote the subgraph of G induced by $V(G) \setminus \{s\}$. We then have $\omega(G) = \max\{\omega(N_G(s)), \max_{u \in V(G_s)} \omega(N_{G_s}(u))\}$. This equality is often used in branch and bound algorithms for the computation of the clique number of G (see for example [12]). By using function h to compute an upper bound on the clique number of $N_G(s)$ as well as on the clique number of the closed neighborhoods of the vertices of G_s, we can obtain another upper bound $h^2_s(G)$ on $\omega(G)$, as demonstrated by the following proposition.

Proposition 2 $\omega(G) \leq \max\{\ h(N_G(s)),\ \max_{u \in V(G_s)} h(N_{G_s}(u))\ \} \equiv h^2_s(G)\ \ \forall s \in V(G)$.

Proof: Consider any vertex $s \in V(G)$. If s belongs to a clique of size $\omega(G)$ in G, then $\omega(G) = \omega(N_G(s)) \leq h(N_G(s)) \leq h^2_s(G)$. Otherwise, there is a clique of size $\omega(G)$ in G_s. By Proposition 1 applied to G_s, we have $\omega(G) = \omega(G_s) \leq \max_{u \in V(G_s)} h(N_{G_s}(u)) \leq h^2_s(G)$. ■

Given the graph G_s, one can repeat the previous process and select another vertex to remove, proceeding in an iterative fashion. This gives the *sequential elimination algorithm* of Figure 1, which provides an upper bound $h^*(G)$ on the clique number of G.

Sequential elimination algorithm

1. Set $G' \leftarrow G$, $G^* \leftarrow G$ and $h^*(G) \leftarrow 0$;
2. Select a vertex $s \in V(G')$;

   If $h^*(G) < h(N_{G'}(s))$ then set $h^*(G) \leftarrow h(N_{G'}(s))$ and $G^* \leftarrow G'$;
3. If $h^*(G) < \max_{u \in V(G')} h(N_{G'}(u))$ then set $G' \leftarrow G'_s$ and go to 2;

   Else STOP : return $h^*(G)$ and G^*.

Figure 1. The sequential elimination algorithm

Notice that the sequential elimination algorithm also returns the subgraph G^* of G for which $h^*(G)$ was updated last. This subgraph will be useful for the computation of a lower bound on $\omega(G)$, as shown in the next section.

Proposition 3 *The sequential elimination algorithm is finite and its output $h^*(G)$ is an upper bound on $\omega(G)$.*

Proof: The algorithm is finite since at most $|V(G)| - 1$ vertices can be removed from G before the algorithm stops. Indeed, if the algorithm enters Step 2 with a unique vertex s in $V(G')$, then $h^*(G) \geq h(N_{G'}(s)) = \max_{u \in V(G')} h(N_{G'}(u))$ at the end of this Step, and the stopping criterion of Step 3 is satisfied.

Let $W \subseteq V(G)$ denote the set of vertices that belong to a maximum clique in G and let G' denote the remaining subgraph of G when the algorithm stops. If $W \subseteq V(G')$, then we know from Proposition 1 applied to G' that $\omega(G) = \omega(G') \leq \max_{u \in V(G')} h(N_{G'}(u)) \leq h^*(G)$. So assume $W \cap (V(G) \setminus V(G')) \neq \emptyset$. Let s be the first vertex in W that was removed from G, and let G'' denote the subgraph of G from which s was removed. Just after removing s, we have $\omega(G) = \omega(G'') = \omega(N_{G''}(s)) \leq h(N_{G''}(s)) \leq h^*(G)$. Clearly, $h^*(G)$ cannot decrease in the remaining iterations, which yields the conclusion. ∎

Under the mild assumption that the upper bound function h is *increasing*, i.e., $h(G') \leq h(G)$ whenever $G' \subseteq G$, we can order the bounds determined by the last three propositions and compare them to $h(G)$, the upper bound computed on G itself.

Proposition 4 *Let s the be the vertex selected at the first iteration of the sequential elimination algorithm. If h is an increasing function, then we have $\omega(G) \leq h^*(G) \leq h^2_s(G) \leq h^1(G) \leq h(G)$.*

Proof: The first inequality, $\omega(G) \leq h^*(G)$, was proved in Proposition 3. To prove the second inequality, $h^*(G) \leq h^2_s(G)$, consider the iteration of the sequential elimination algorithm where $h^*(G)$ was updated last. If this last update happened at the first iteration, we have $h^*(G) = h(N_G(s)) \leq h^2_s(G)$. Otherwise, let s' be the vertex selected for the last update of $h^*(G)$, and let G^* denote the subgraph in which s' was selected. We have $N_{G^*}(s') \subseteq N_{G_s}(s')$ which gives $h^*(G) = h(N_{G^*}(s')) \leq h(N_{G_s}(s')) \leq \max_{u \in V(G_s)} h(N_{G_s}(u)) \leq h^2_s(G)$.
The inequality $h^2_s(G) \leq h^1(G)$ follows from the fact that G_s is a subgraph of G. Indeed, $h^2_s(G) = \max\{h(N_G(s)), \max_{u \in V(G_s)} h(N_{G_s}(u))\} \leq \max\{h(N_G(s)), \max_{u \in V(G_s)} h(N_G(u))\} = \max_{u \in V(G)} h(N_G(u)) = h^1(G)$. Finally, the inequality $h^1(G) \leq h(G)$ is a direct consequence of the hypothesis that h is increasing, since the closed neighborhood of any vertex of G is an induced subgraph of G. ∎

If h is not increasing, the relationships above between the different bounds do not necessarily hold. Consider, for instance, the following upper bound function:

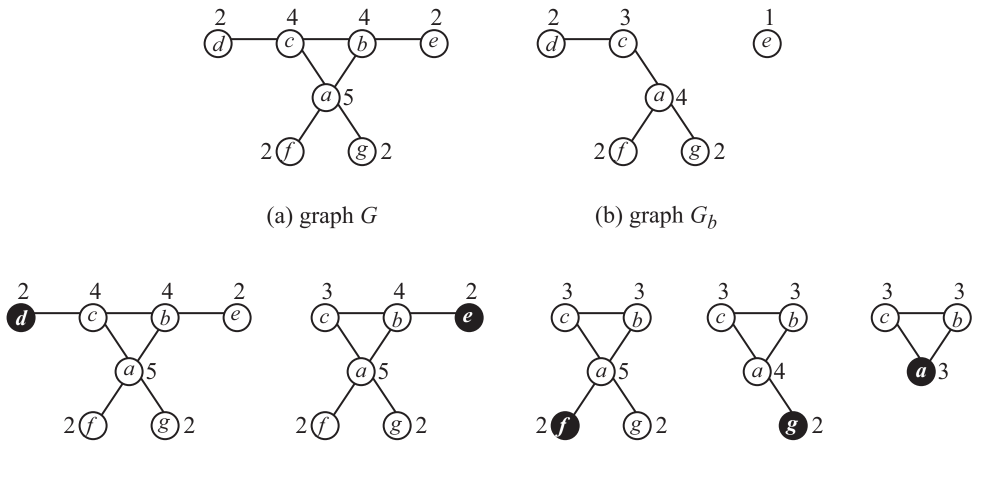
$$h(G) = \begin{cases} \chi(G) & \text{if } |V(G)| \geq 4 \\ |V(G)| & \text{otherwise.} \end{cases}$$

Using this function on a square graph (4 vertices, 4 edges, organised in a square) gives $h(G) = \chi(G) = 2$, while the upper bound computed on the closed neighborhood of any vertex u gives $h(N_G(u)) = |V(N_G(u))| = 3 > h(G)$, which implies $h^*(G) = h^2_s(G) = h^1(G) = 3 > 2 = h(G)$.

Nonetheless, even when h is not increasing, it is easy to modify the bound definitions to obtain the result of the last proposition. For instance, one can replace each bound $h(N_{G'}(u))$ by $\min\{h(G), h(N_{G'}(u))\}$. In practice, this implies that we add computing time at the start of the sequential elimination algorithm to determine $h(G)$, only to prevent an event that is unlikely to happen. This is why we chose not to incorporate this safeguard into our implementation of the sequential elimination algorithm.

To fully describe the sequential elimination algorithm, it remains to specify how to select the vertex s to be removed at every iteration. Since the sequential elimination algorithm updates the value of $h^*(G)$ by setting $h^*(G) \leftarrow \max\{h^*(G), h(N_{G'}(s))\}$, we select the vertex s that minimizes $h(N_{G'}(u))$ over all $u \in V(G')$.

Figure 2 illustrates all bounds when using $h(G) = |V(G)|$. We indicate the value $h(N_{G'}(u))$ for every graph G' and for every vertex $u \in V(G')$. We obviously have $h(G) = 7$. As shown in Figure 2a, $h(N_G(a)) = 5$, $h(N_G(b)) = h(N_G(c)) = 4$, and $h(N_G(u)) = 2$ for $u = d, e, f, g$, and we therefore have $h^1(G) = 5$. Also, we see from Figure 2b that $h(N_{G_b}(a)) = 4$, $h(N_{G_b}(c)) = 3$, $h(N_{G_b}(u)) = 2$ for $u = d, e, f$, and $h(N_{G_b}(g)) = 1$, and this gives an upper bound $h^2_b(G) = 4$. The sequential elimination algorithm is illustrated in Figure 2c. The black vertices correspond to the selected vertices. At the first iteration, one can choose $s = d, e, f$ or g, say d and this gives value 2 to $h^*(G)$. Then vertices e, f and g are removed without modifying $h^*(G)$. Finally, the algorithm selects one of the three vertices in the remaining graph G', say a, and stops since $h^*(G)$ is set equal to $3 = h(N_{G'}(a)) = h(N_{G'}(b)) = h(N_{G'}(c))$. The final upper bound is therefore $h^*(G) = 3$, which corresponds to the size of the maximum clique.



(a) graph G (b) graph G_b

(c) illustation of the decomposition algorithm

Figure 2. Illustration of the upper bounds

Notice that if $h(G) = |V(G)|$, then the sequential elimination algorithm always chooses a vertex with minimum degree in the remaining graph. Hence, it is equivalent to the procedure proposed by Szekeres and Wilf [14] for the computation of an upper bound on the chromatic number $\chi(G)$. For other upper bounding procedures h, our sequential elimination algorithm possibly gives a bound $h^*(G) < \chi(G)$. For example, assume that h is a procedure that returns the number of colors used by a linear coloring algorithm that orders the vertices randomly and then colors them sequentially according to that order, giving the smallest available color to each vertex. We then have $h(G) \geq \chi(G)$. However, for G equal to a pentagon (the chordless cycle on five vertices), $h(N_G(u)) = 2$ for all $u \in V(G)$, which implies $\chi(G) = 3 > 2 = h^*(G) = h^1(G) = h_s^2(G)$ for all $s \in V(G)$.

Given a graph G with n vertices and m edges, the computational complexity of the sequential elimination algorithm is $O(n^2\ T(n,m))$, where $T(n,m)$ is the time taken to compute $h(G)$ on G. Since $h^1(G)$ and $h_s^2(G)$ can both be computed in $O(n\ T(n,m))$, significant improvements in the quality of the upper bounds need to be observed to justify this additional computational effort. Our computational results, presented in Section 4.1, show that this is indeed the case. Before presenting these results, we will see how to use the sequential elimination algorithm to also improve the computation of lower bounds on the clique number of a graph.

3 Using the Sequential Elimination Algorithm to Compute Lower Bounds

In this section we show how to exploit the results of the sequential elimination algorithm to compute lower bounds on the clique number of a graph. To this end, we make use of the following proposition.

Proposition 5 *Let $h^*(G)$ and G^* be the output of the sequential elimination algorithm. If $h^*(G) = \omega(G)$, then G^* contains all cliques of maximum size of G.*

Proof: Suppose there exists a clique of size $\omega(G)$ in G that is not in G^*, and let t^* be the iteration where $h^*(G)$ was updated last. At some iteration t', prior to t^*, some vertex s' belonging to a maximum clique of G was removed from some graph $G' \subseteq G$. Hence $h^*(G) \geq h(N_{G'}(s')) \geq \omega(N_{G'}(s')) = \omega(G)$ at the end of iteration t'. Since $h^*(G)$ is updated (increased) at iteration t^*, we have $h^*(G) > \omega(G)$ at the end of iteration t^*, a contradiction. ■

Notice that when $h^*(G) > \omega(G)$, it may happen that $\omega(G^*) < \omega(G)$. For example, for the left graph G of Figure 3, with $h(G) = |V(G)|$, the sequential elimination algorithm first selects $s = a$ or b, say a, and $h^*(G)$ is set equal to 3. Then b is removed without changing the value of $h^*(G)$. Finally, one of the 6 remaining vertices is selected, the bound $h^*(G)$ is set equal to 4, and the algorithm stops since there are no vertices with $h(N_{G'}(u)) > 4$ in the remaining graph G'. Hence G^* has 6 vertices and $\omega(G^*) = 2 < 3 = \omega(G)$. According to Proposition 5, this is possible only because $h^*(G) = 4 > 3 = \omega(G)$.

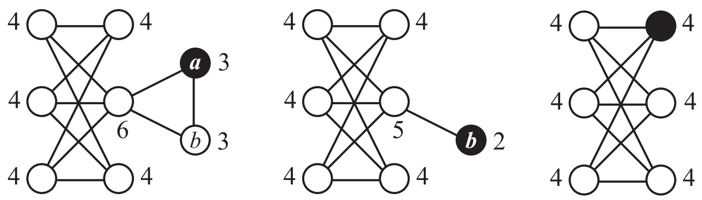


Figure 3. A graph G with $\omega(G^*) < \omega(G)$

In order to obtain a lower bound on $\omega(G)$, Proposition 5 suggests to determine G^*, and to run an algorithm (either exact or heuristic) to get a lower bound on the clique number of G^*. Clearly, the size of such a clique is a lower bound on the clique number of G. This process is summarized in Figure 4, where $\ell(G)$ is any known procedure that computes a lower bound on the clique number of a graph G, while $\ell^*(G)$ is the new lower bound produced by our algorithm.

Lower bounding procedure

1. Run the sequential elimination algorithm on G to get G^*;
2. Set $\ell^*(G) \leftarrow \ell(G^*)$.

Figure 4. Algorithm for the computation of a lower bound on $\omega(G)$

When G^* is small enough, an exact algorithm can be used at Step 2 to determine $\omega(G^*)$, while it can be too time consuming to use the same exact algorithm to compute $\omega(G)$. However, for many instances, the iteration where $h^*(G)$ is updated last is reached very early, as will be shown in the next section, and using an exact algorithm at Step 2 is often not realistic. We will observe in the next section that even if ℓ is a heuristic lower bounding function, it often happens that $\ell^*(G) = \ell(G^*) > \ell(G)$. Notice that such a situation can only happen if ℓ is not a decreasing function (i.e., $\ell(G')$ is possibly larger than $\ell(G)$ for a subgraph $G' \subset G$). This is illustrated with ℓ equal to the well-known greedy algorithm MIN [7] described in Figure 5; we then use the notation $\ell(G)$ =MIN(G).

Procedure MIN

Set $G' \leftarrow G$ and $K \leftarrow \emptyset$;
While $G' \neq \emptyset$ do
  Set $s \leftarrow \text{argmax}_{u \in V(G') \setminus K} |N_{G'}(u)|$;
  Set $G' \leftarrow N_{G'}(s)$ and $K \leftarrow K \cup \{s\}$;
Return $|K|$.

Figure 5. Greedy lower bounding algorithm for the clique number

Algorithm MIN applied on the graph G of Figure 6 returns value 2, since vertex a has the largest number of neighbors and is therefore chosen first. The sequential elimination algorithm with $h(G) = |V(G)|$ first chooses $s = b, c$ or d, say b, which gives value 2 to $h^*(G)$. Then c, d and a are removed without changing the value of $h^*(G)$. Finally, one of the vertices e, f or g is selected, which gives $h^*(G) = 3$, and the algorithm stops. Hence, G^* is the triangle induced by vertices e, f and g, and procedure MIN applied to this triangle returns value 3. In summary, we have $\ell^*(G) = \ell(G^*) = 3 > 2 = \ell(G)$. Notice also that even if MIN and h are not very efficient lower and upper bounding procedures (since $\ell(G) = 2 < 3 = \omega(G) < 6 = h(G)$), they help getting better bounds. In our example, we have $\ell^*(G) = \ell(G^*) = 3 = h^*(G)$, which provides a proof that $\omega(G) = 3$.

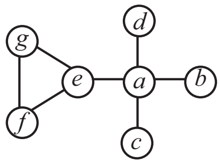


Figure 6. A graph G with $\ell^*(G) > \ell(G)$

4 Computational Results

The objective of our computational experiments is twofold. First, we analyze the effectiveness of the sequential elimination algorithm when using different upper bound functions h. Second, we present the lower bounds obtained by running several maximum clique algorithms (exact and heuristic) on the graph G^* resulting from the sequential elimination algorithm (using an effective upper bound function h). All our tests were performed on 93 instances used in the second DIMACS challenge [9]. Most instances come from the maximum clique section, though we also included a few instances from the graph coloring section, since we use graph coloring heuristics as upper bound functions. The characteristics of the selected instances can be found in the next subsection.

4.1 Computing Upper Bounds

Since the sequential elimination method produces a bound $h^*(G)$ with a significant increase in computing time when compared to the computation of $h(G)$, we did not use hard-to-compute upper bound functions like Lovasz' theta function [10, 11] (which gives a value between the clique number and the chromatic number). Although there is a polynomial time algorithm to compute this bound, it is still time-consuming (even for relatively small instances) and difficult to code.

The most trivial bounds we tested are the number of vertices $h_a(G) = |V(G)|$ and the size of the largest closed neighborhood $h_b(G) = \max_{u \in V(G)} |V(N_G(u))|$. Apart from these loose bounds, we obtained tighter bounds by computing upper bounds on the chromatic number with three fast heuristic methods. The simplest is the linear coloring algorithm (which we denote $h_c(G)$), which consists in assigning the smallest available color to each vertex, using the order given in the file defining the graph. The second graph coloring method we tested is DSATUR [4] (denoted $h_d(G)$), a well-known greedy algorithm which iteratively selects a vertex with maximum saturation degree and assigns to it the smallest available color, the saturation degree of a vertex u being defined as the number of colors already used in $N_G(u)$. Finally, the last method we tested (denoted $h_e(G)$) starts with the solution found by DSATUR and runs the well-known tabu search algorithm Tabucol [8, 5], performing as many iterations as there are vertices in the graph (which is a very small amount of iterations for a tabu search).

Let G be a graph with n vertices and m edges. As mentioned at the end of Section 2, the computational complexity of the sequential elimination algorithm is $O(n^2\, T_x(n, m))$, where $T_x(n, m)$ is the time taken to compute $h_x(G)$ on G (and x is any letter between a and e). The functions $h_x(G)$ defined above can easily be implemented so that $T_a(n, m) \in O(n)$, $T_b(n, m) \in O(m)$, $T_c(n, m) \in O(m)$, $T_d(n, m) \in O(n^2)$, and $T_e(n, m) \in O(n^3)$. The last bound is easily derived, since Tabucol is initialized with DSATUR, which gives the bound $h_d(G) \leq n$, and a neighbor solution is obtained by changing the color of one vertex. Thus, the neighborhood is explored in time $O(nh_d(G))$, and since we perform n iterations, we obtain a worst-case complexity of $O(n^2 h_d(G)) \subseteq O(n^3)$.

Tables 1 and 2 give the detailed results obtained when using the sequential elimination algorithm with these five upper bound functions. The first column (Problem) indicates the name of the problem instance taken from the DIMACS site; the second column (W)

gives the size of the largest known clique; the remaining columns indicate the upper bounds $h_x(G)$ and $h_x^*(G)$ computed by each of the five functions on the original graph and when using the sequential elimination algorithm.

To analyze these results, we use the following *improvement ratio*, which is a value in the interval [0,1]:

$$I_x = \frac{h_x(G) - h_x^*(G)}{h_x(G) - W}.$$

A value of 0 indicates that the bound $h_x^*(G)$ does not improve upon $h_x(G)$, while a value of 1 corresponds to the case where $h_x^*(G) = W$, i.e., the maximum possible improvement (if W is indeed the clique number) is achieved by the sequential elimination algorithm. We have discarded the cases where the upper bound function applied to G already found a maximum clique, since then there is no possible improvement to be gained by using the sequential elimination algorithm. Table 3 displays the improvement ratios (in %) averaged for each family of graphs and for all instances. The first column (Problem) indicates the family of graphs, each being represented by the first characters identifying them, followed by a star (*). The remaining columns show the average improvement for the five upper bound functions.

Our initial conjecture was that the improvement would be inversely correlated to the quality of the upper bound function, i.e., the worst the function, the more room for improvement there is, hence the most improvement should be obtained. The results do not verify this conjecture, since the best upper bound functions show better improvements than the worst ones. Indeed, the worst functions, $h_a(G)$ and $h_b(G)$, display improvements of 50% and 43%, respectively, while the best functions, $h_c(G)$, $h_d(G)$ and $h_e(G)$, reach improvements of 55%, 60% and 64%, respectively. Even among the graph coloring algorithms, we observe an inverse relationship, i.e., as the effectiveness of the method increases, the improvement values also increase. At first, we thought this phenomenon might be due to the fact that for some families of graphs average improvements were reaching 100%, thus influencing the overall average improvement more than it should. But a similar progression can be observed for most families of graphs. Another explana-

		$\lvert V(G)\rvert$		$\max_{u\in V(G)} \lvert V(N_G(u))\rvert$		Linear coloring		DSATUR		DSATUR + Tabucol	
Problem	W	$h_a(G)$	$h^*_a(G)$	$h_b(G)$	$h^*_b(G)$	$h_c(G)$	$h^*_c(G)$	$h_d(G)$	$h^*_d(G)$	$h_e(G)$	$h^*_e(G)$
brock200_1	21	200	135	166	114	59	42	53	38	47	35
brock200_2	12	200	85	115	54	36	19	31	17	30	16
brock200_3	15	200	106	135	76	45	28	39	24	35	23
brock200_4	17	200	118	148	90	49	32	43	29	41	27
brock400_1	27	400	278	321	226	102	75	93	68	89	62
brock400_2	29	400	279	329	229	100	74	93	68	90	62
brock400_3	31	400	279	323	227	103	75	92	68	83	63
brock400_4	33	400	278	327	228	100	74	91	68	90	62
brock800_1	23	800	488	561	345	144	96	137	88	128	80
brock800_2	24	800	487	567	347	144	96	134	88	122	81
brock800_3	25	800	484	559	346	143	96	133	87	123	80
brock800_4	26	800	486	566	346	148	96	136	87	124	81
c-fat200-1	12	200	15	18	15	12	12	15	12	14	12
c-fat200-2	24	200	33	35	33	24	24	24	24	24	24
c-fat200-5	58	200	84	87	84	68	58	84	58	83	58
c-fat500-1	14	500	18	21	18	14	14	14	14	14	14
c-fat500-10	126	500	186	189	186	126	126	126	126	126	126
c-fat500-2	26	500	36	39	36	26	26	26	26	26	26
c-fat500-5	64	500	93	96	93	64	64	64	64	64	64
c1000_9	68	1000	875	926	814	319	283	305	266	276	238
c125_9	34	125	103	120	100	57	49	52	44	47	41
c2000_5	16	2000	941	1075	518	226	120	210	110	198	102
c2000_9	77	2000	1759	1849	1625	592	519	562	492	492	442
c250_9	44	250	211	237	200	98	86	92	78	82	71
c4000_5	18	4000	1910	2124	1028	402	215	377	200	365	186
c500_9	57	500	433	469	408	184	156	164	144	149	131
dsjc1000_5	15	1000	460	552	263	127	68	115	61	109	57
dsjc500_5	13	500	226	287	134	72	39	65	35	61	33
gen200_p0_9_44	44	200	168	191	161	76	63	62	53	44	44
gen200_p0_9_55	55	200	167	191	161	80	68	71	60	62	55
gen400_p0_9_55	55	400	337	376	320	127	110	102	81	55	55
gen400_p0_9_65	65	400	337	379	321	136	118	118	99	65	65
gen400_p0_9_75	75	400	337	381	322	143	124	118	103	75	79
hamming10-2	512	1024	1014	1014	1006	512	512	512	512	512	512
hamming10-4	40	1024	849	849	724	128	121	85	71	85	70
hamming6-2	32	64	58	58	54	32	32	32	32	32	32
hamming6-4	4	64	23	23	8	8	5	7	5	7	5
hamming8-2	128	256	248	248	242	128	128	128	128	128	128
hamming8-4	16	256	164	164	110	32	27	24	17	24	16
johnson16-2-4	8	120	92	92	68	14	13	14	13	14	13
johnson32-2-4	16	496	436	436	380	30	29	30	29	30	29
johnson8-2-4	4	28	16	16	8	6	5	6	5	6	5
johnson8-4-4	14	70	54	54	42	20	17	17	14	14	14
keller4	11	171	103	125	76	37	18	24	17	22	15
keller5	27	776	561	639	459	175	50	61	49	59	43
keller6	59	3361	2691	2953	2350	781	126	141	122	141	118

Table 1: Upper bounds obtained with five upper bound functions $h_x(G)$

		$\|V(G)\|$		$\max_{u\in V(G)} \|V(N_G(u))\|$		Linear coloring		DSATUR		DSATUR + Tabucol	
Problem	W	$h_a(G)$	$h_a^*(G)$	$h_b(G)$	$h_b^*(G)$	$h_c(G)$	$h_c^*(G)$	$h_d(G)$	$h_d^*(G)$	$h_e(G)$	$h_e^*(G)$
latin_square_10.col	90	900	684	684	636	213	144	132	108	119	105
le450_15a.col	15	450	25	100	18	22	15	17	15	17	15
le450_15b.col	15	450	25	95	17	22	15	16	15	16	15
le450_15c.col	15	450	50	140	29	30	15	23	15	23	15
le450_15d.col	15	450	52	139	29	31	15	24	15	23	15
le450_25a.col	25	450	27	129	26	28	25	25	25	25	25
le450_25b.col	25	450	26	112	25	27	25	25	25	25	25
le450_25c.col	25	450	53	180	39	37	25	29	25	29	25
le450_25d.col	25	450	52	158	38	35	25	28	25	28	25
le450_5a.col	5	450	18	43	7	14	5	10	5	10	5
le450_5b.col	5	450	18	43	7	13	5	9	5	9	5
le450_5c.col	5	450	34	67	12	17	6	10	5	9	5
le450_5d.col	5	450	33	69	12	18	5	12	5	11	5
MANN_a27	126	378	365	375	363	135	135	140	137	135	131
MANN_a45	345	1035	1013	1032	1011	372	370	369	367	363	353
MANN_a81	1100	3321	3281	3318	3279	1134	1134	1153	1146	1135	1124
MANN_a9	16	45	41	42	39	18	18	19	18	18	17
p_hat1000-1	10	1000	164	409	84	69	24	52	20	52	19
p_hat1000-2	46	1000	328	767	289	148	89	109	76	109	74
p_hat1000-3	68	1000	610	896	554	230	160	187	134	187	132
p_hat1500-1	12	1500	253	615	126	95	33	74	28	74	27
p_hat1500-2	65	1500	505	1154	451	213	133	157	113	157	112
p_hat1500-3	94	1500	930	1331	840	326	231	270	195	270	194
p_hat300-1	8	300	50	133	28	29	11	22	9	21	9
p_hat300-2	25	300	99	230	90	56	34	42	29	42	28
p_hat300-3	36	300	181	268	166	85	59	69	51	69	49
p_hat500-1	9	500	87	205	46	45	16	32	13	32	13
p_hat500-2	36	500	171	390	152	87	54	66	46	66	45
p_hat500-3	50	500	304	453	279	131	94	108	78	107	76
p_hat700-1	11	700	118	287	62	53	19	40	16	40	16
p_hat700-2	44	700	236	540	213	114	71	85	60	85	59
p_hat700-3	62	700	427	628	389	171	120	143	102	141	100
san1000	15	1000	465	551	400	47	21	24	16	15	15
san200_0_7_1	30	200	126	156	108	49	32	42	30	31	30
san200_0_7_2	18	200	123	165	113	35	24	23	18	18	18
san200_0_9_1	70	200	163	192	156	92	75	73	70	70	70
san200_0_9_2	60	200	170	189	161	86	75	75	63	62	60
san200_0_9_3	44	200	170	188	161	73	65	64	53	48	44
san400_0_5_1	13	400	184	226	155	29	14	21	13	21	13
san400_0_7_1	40	400	262	302	224	81	54	59	43	45	40
san400_0_7_2	30	400	260	305	217	67	49	47	36	30	30
san400_0_7_3	22	400	254	308	203	59	42	29	26	22	22
san400_0_9_1	100	400	345	375	325	163	138	135	116	109	100
sanr200_0_7	18	200	125	162	100	52	36	47	32	42	30
sanr200_0_9	42	200	167	190	158	82	70	74	64	69	59
sanr400_0_5	13	400	178	234	108	62	33	56	29	50	27
sanr400_0_7	21	400	259	311	201	91	64	83	58	78	53

Table 2: Upper bounds obtained with five upper bound functions $h_x(G)$

Problem	$I_a(\%)$	$I_b(\%)$	$I_c(\%)$	$I_d(\%)$	$I_e(\%)$
brock*	41	40	45	47	50
c-fat*	93	23	100	100	100
c*	27	27	30	32	33
dsjc*	56	54	54	56	57
gen*	20	20	33	47	100
hamming*	25	25	38	62	67
johnson*	29	35	31	43	25
keller*	30	31	83	37	47
latin*	27	8	56	57	48
le*	96	93	99	100	100
MANN*	6	5	2	19	45
p_hat*	66	63	62	64	66
san*	36	29	61	79	100
sanr*	39	40	44	47	48
ALL	50	43	55	60	64

Table 3: Average improvement for each family of graphs

tion would be that if two functions show different results when applied to G but identical results when embedded into the sequential elimination algorithm, then the improvement would be better for the best function because the denominator is smaller. When we look at the detailed results, however, we notice that in general, the better the function, the better the results obtained by the sequential elimination algorithm.

Computing times are reported in Tables 4 and 5. The times needed to compute $h_x(G)$ and $h_x^*(G)$ appear in columns T_x and T_x^*, respectively. All times are in seconds and were obtained on an AMD Opteron(tm) 275/2.2 GHz Processor. A zero value means that the bound was obtained in less than 0.5 seconds, while times larger than 10 hours (i.e., 36000 seconds) are reported as “>10h”. It clearly appears that the best upper bounds are obtained with the most time consuming procedures. For example, for the c2000_5 instance, Tabucol makes it possible to compute $h_e^*(G) = 102$ while $h_a^*(G) = 941$, $h_b^*(G) = 518$, $h_c^*(G) = 120$, and $h_d^*(G) = 110$, but such an improvement requires an increase of the computing time from 0 second for $h_a^*(G)$ and several minutes for $h_b^*(G)$, $h_c^*(G)$, and $h_d^*(G)$, to more than 5 hours for $h_e^*(G)$. Given the computational complexity analysis performed earlier, these experimental results can be easily explained, since the sequential elimination with Tabucol requires a worst-case time of $O(n^5)$.

			$\|V(G)\|$		$\max_{u \in V(G)} \|V(N_G(u))\|$		Linear coloring		DSATUR		DSATUR + Tabucol	
Problem	n	m	T_a	T_a^*	T_b	T_b^*	T_c	T_c^*	T_d	T_d^*	T_e	T_e^*
brock200_1	200	14834	0	0	0	0	0	0	0	0	0	6
brock200_2	200	9876	0	0	0	0	0	0	0	0	0	1
brock200_3	200	12048	0	0	0	0	0	0	0	0	0	3
brock200_4	200	13089	0	0	0	0	0	0	0	0	0	3
brock400_1	400	59723	0	0	0	1	0	1	0	5	0	92
brock400_2	400	59786	0	0	0	1	0	1	0	3	0	71
brock400_3	400	59681	0	0	0	1	0	0	0	4	0	82
brock400_4	400	59765	0	0	0	1	0	1	0	5	0	66
brock800_1	800	207505	0	0	0	10	0	6	0	63	2	1466
brock800_2	800	208166	0	0	0	10	0	6	0	33	3	855
brock800_3	800	207333	0	0	0	8	0	6	0	32	2	669
brock800_4	800	207643	0	0	0	9	0	7	0	54	2	786
c-fat200-1	200	1534	0	0	0	0	0	0	0	0	0	0
c-fat200-2	200	3235	0	0	0	0	0	0	0	0	0	0
c-fat200-5	200	8473	0	0	0	0	0	0	0	0	0	1
c-fat500-1	500	4459	0	0	0	0	0	0	0	0	0	0
c-fat500-10	500	46627	0	0	0	0	0	0	0	1	0	19
c-fat500-2	500	9139	0	0	0	0	0	0	0	0	0	0
c-fat500-5	500	23191	0	0	0	0	0	0	0	0	0	3
C1000_9	1000	45079	0	0	0	16	0	20	0	162	9	12241
C125_9	125	6963	0	0	0	0	0	0	0	0	0	1
C2000_5	2000	999836	0	0	0	153	0	100	0	803	30	20034
C2000_9	2000	1799532	0	0	0	119	0	134	0	1522	141	>10h
C250_9	250	27984	0	0	0	0	0	0	0	1	0	37
C4000_5	4000	4000268	0	1	0	1454	0	909	1	6906	178	>10h
C500_9	500	112332	0	0	0	2	0	2	0	11	1	1115
DSJC1000_5	1000	249826	0	0	0	18	0	12	0	83	2	880
DSJC500_5	500	62624	0	0	0	1	0	1	0	5	0	42
gen200_p0_9_44	200	17910	0	0	0	0	0	0	0	1	0	11
gen200_p0_9_55	200	17910	0	0	0	0	0	0	0	1	0	8
gen400_p0_9_55	400	71820	0	0	0	2	0	1	0	19	0	53
gen400_p0_9_65	400	71820	0	0	0	1	0	1	0	13	0	104
gen400_p0_9_75	400	71820	0	0	0	1	0	1	0	6	0	630
hamming10-2	1024	518656	0	0	0	6	0	6	0	57	4	3189
hamming10-4	1024	434176	0	0	0	6	0	5	0	394	1	3102
hamming6-2	64	1824	0	0	0	0	0	0	0	0	0	0
hamming6-4	64	704	0	0	0	0	0	0	0	0	0	0
hamming8-2	256	31616	0	0	0	0	0	0	0	0	0	12
hamming8-4	256	20864	0	0	0	0	0	0	0	1	0	5
johnson16-2-4	120	5460	0	0	0	0	0	0	0	0	0	0
johnson32-2-4	496	107880	0	0	0	1	0	1	0	3	0	22
johnson8-2-4	28	210	0	0	0	0	0	0	0	0	0	0
johnson8-4-4	70	1855	0	0	0	0	0	0	0	0	0	0
keller4	171	9435	0	0	0	0	0	0	0	0	0	1
keller5	776	225990	0	0	0	3	0	20	0	164	0	2157
keller6	3361	4619898	0	0	0	269	0	11518	1	>10h	14	>10h

Table 4: Computing times in seconds obtained with five upper bound functions $h_x(G)$

			$\|V(G)\|$		$\max_{u\in V(G)} \|V(N_G(u))\|$		Linear coloring		DSATUR		DSATUR + Tabucol	
Problem	n	m	T_a	T_a^*	T_b	T_b^*	T_c	T_c^*	T_d	T_d^*	T_e	T_e^*
latin_square_10.col	900	307350	0	0	0	4	0	10	0	109	5	2207
le450_15a.col	450	8168	0	0	0	0	0	0	0	0	0	1
le450_15b.col	450	8169	0	0	0	0	0	0	0	0	0	1
le450_15c.col	450	16680	0	0	0	0	0	0	0	0	0	2
le450_15d.col	450	16750	0	0	0	0	0	0	0	0	0	2
le450_25a.col	450	8260	0	0	0	0	0	0	0	0	0	1
le450_25b.col	450	8263	0	0	0	0	0	0	0	0	0	1
le450_25c.col	450	17343	0	0	0	1	0	0	0	1	0	3
le450_25d.col	450	17425	0	0	0	1	0	0	0	1	0	2
le450_5a.col	450	5714	0	0	0	0	0	0	0	0	0	0
le450_5b.col	450	5734	0	0	0	0	0	0	0	0	0	0
le450_5c.col	450	9803	0	0	0	0	0	0	0	0	0	0
MANN_a27	378	70551	0	0	0	0	0	0	0	2	0	86
MANN_a45	1035	533115	0	0	0	9	0	10	0	83	5	10104
MANN_a81	3321	5506380	0	0	0	293	0	361	1	9544	177	>10h
MANN_a9	45	918	0	0	0	0	0	0	0	0	0	0
p_hat1000-1	1000	122253	0	0	0	30	0	14	0	45	0	144
p_hat1000-2	1000	244799	0	0	0	191	0	77	0	569	1	3176
p_hat1000-3	1000	371746	0	0	0	33	0	26	0	390	1	4209
p_hat1500-1	1500	284923	0	1	0	145	0	70	0	298	1	862
p_hat1500-2	1500	568960	0	1	0	953	0	366	0	3408	2	22368
p_hat1500-3	1500	847244	0	0	0	145	0	131	0	2325	4	30069
p_hat300-1	300	10933	0	0	0	0	0	0	0	0	0	2
p_hat300-2	300	21928	0	0	0	2	0	1	0	3	0	17
p_hat300-3	300	33390	0	0	0	1	0	1	0	3	0	29
p_hat500-1	500	31569	0	0	0	2	0	1	0	3	0	11
p_hat500-2	500	62946	0	0	0	12	0	6	0	25	0	171
p_hat500-3	500	93800	0	0	0	3	0	4	0	21	0	262
p_hat700-1	700	60999	0	0	0	7	0	4	0	12	0	42
p_hat700-2	700	121728	0	0	0	47	0	20	0	107	0	733
p_hat700-3	700	183010	0	0	0	15	0	11	0	90	1	1075
san1000	1000	250500	0	0	0	6	0	28	0	133	0	397
san200_0_7_1	200	13930	0	0	0	0	0	0	0	0	0	2
san200_0_7_2	200	13930	0	0	0	0	0	0	0	0	0	5
san200_0_9_1	200	17910	0	0	0	0	0	0	0	1	0	16
san200_0_9_2	200	17910	0	0	0	0	0	0	0	0	0	14
san200_0_9_3	200	17910	0	0	0	0	0	0	0	1	0	6
san400_0_5_1	400	39900	0	0	0	0	0	1	0	4	0	11
san400_0_7_1	400	55860	0	0	0	0	0	1	0	6	0	82
san400_0_7_2	400	55860	0	0	0	0	0	1	0	10	0	43
san400_0_7_3	400	55860	0	0	0	0	0	1	0	18	0	10
san400_0_9_1	400	71820	0	0	0	1	0	1	0	7	0	446
sanr200_0_7	200	13868	0	0	0	0	0	0	0	0	0	4
sanr200_0_9	200	17863	0	0	0	0	0	0	0	0	0	9
sanr400_0_5	400	39984	0	0	0	1	0	0	0	3	0	12
sanr400_0_7	400	55869	0	0	0	1	0	1	0	3	0	51

Table 5: Computing times in seconds obtained with five upper bound functions $h_x(G)$

4.2 Computing Lower Bounds

In this section, we present the results obtained when computing lower bounds using G^*, the graph obtained at the iteration where $h^*(G)$ was updated last in the sequential elimination algorithm. We use DSATUR ($h_d(G)$) as upper bound function, since it shows a good balance between solution effectiveness and computational efficiency. We tested four maximum clique algorithms to compute lower bounds:

- An exact branch-and-bound algorithm, dfmax [9] (available as a C program on the DIMACS ftp site [2]), performed with a time limit of five hours. We denote the lower bound obtained by this algorithm when applied to G and G^* as $l_a(G)$ and $l_a(G^*)$, respectively.
- A very fast greedy heuristic, MIN [7] (already described in Section 3). We denote the lower bounds obtained by this algorithm when applied to G and G^* as $l_b(G)$ and $l_b(G^*)$, respectively.
- The penalty-evaporation heuristic [13], which, at each iteration, inserts into the current clique some vertex i, removing the vertices not adjacent to i. The removed vertices are penalized in order to reduce their potential of being selected to be inserted again during the next iterations. This penalty is gradually evaporating. We denote the lower bounds obtained by this algorithm when applied to G and G^* as $l_c(G)$ and $l_c(G^*)$, respectively.
- An improved version of the above penalty-evaporation heuristic, summarized in Figure 7. We denote the lower bounds obtained by this algorithm when applied to G and G^* as $l_d(G)$ and $l_d(G^*)$, respectively.

Improved penalty-evaporation heuristic

1. Set $IPE(G) \leftarrow 0$ and $G' \leftarrow G$;
2. Run the penalty-evaporation heuristic on G' to get a clique K;
   If $|K| > IPE(G)$ then set $IPE(G) \leftarrow |K|$;
3. For all $u \in K$ do

   Run the penalty-evaporation heuristic on $N_{G'}(u)$ to get a clique K';
   If $|K'| > IPE(G)$ then set $IPE(G) \leftarrow |K'|$, $K \leftarrow K'$, and restart Step 3;

4. Remove K from G' (i.e., set G' equal to the subgraph induced by $V(G') \setminus K$);
   If G' is not empty then go to 2 else STOP: return $IPE(G)$.

Figure 7. Improved penalty-evaporation heuristic [13]

The results obtained on the same instances as in Section 4.1 are presented in Tables 6 and 7. Of the 93 instances, we removed those where G and G^* coincide, which left 72 instances. The first column (Problem) indicates the name of each problem; the second and third columns show the number of vertices in G (n) and G^* (n^*), respectively; the fourth column (W) gives the size of the largest known clique; the fifth and sixth columns ($l_a(G)$ and $l_a(G^*)$) show the results obtained by dfmax (with a time limit of five hours) when applied to G and G^*, respectively (a + sign indicates the algorithm was stopped because of the time limit, so G or G^* might contain a clique of size larger than the given value); the remaining columns give the lower bounds generated by the three maximum clique heuristic methods, when applied to G and G^*.

Column $l_a(G^*)$ indicates that dfmax has determined $\omega(G^*)$ within the time limit of five hours for 41 out the 72 instances. By comparing columns W, $l_a(G)$ and $l_a(G^*)$ on these instances, we observe that $\omega(G^*) = \omega(G) = W$ in 38 cases, while $\omega(G^*) < \omega(G)$ in one case (instance p_hat1500-1) and $\omega(G^*) = W$ and $\omega(G)$ is not known in two cases (instances c-fat500-10 and san200_0_9_1). We do not report computing times since the aim of the experiments is to compare the quality of the lower bounds on G and G^*. We find however interesting to mention that for 22 of the 38 instances with $\omega(G^*) = \omega(G)$, dfmax has determined the clique number, both in G and G^*, in less than 1 second. For the 16 other instances, the decrease in computing time is in average equal to 47%.

For the 31 instances for which $\omega(G^*)$ is not known, we deduce from columns W, $l_a(G^*)$ and $l_d(G^*)$ that $\omega(G^*) \geq W$ in 24 cases. The status of the seven remaining instances is yet unknown. Also, $l_a(G) < l_a(G^*)$ for 11 out of these 31 instances while $l_a(G) > l_a(G^*)$ for 6 of them (and $l_a(G) = l_a(G^*)$ for the 14 remaining instances).

Furthermore, let $\frac{|V(G)|-|V(G^*)|}{|V(G)|}$ be the reduction ratio between the number of vertices in graphs G and G^*. The mean reduction ratio for the 72 instances is 20%, among which

				dfmax		MIN		Penalty-evaporation		Imp. Penalty-evaporation	
Problem	n	n^*	W	$l_a(G)$	$l_a(G^*)$	$l_b(G)$	$l_b(G^*)$	$l_c(G)$	$l_c(G^*)$	$l_d(G)$	$l_d(G^*)$
brock200_1	200	198	21	21	21	14	16	20	20	20	21
brock200_2	200	197	12	12	12	7	8	10	11	11	12
brock200_3	200	199	15	15	15	10	11	14	13	14	14
brock200_4	200	195	17	17	17	11	13	16	16	17	17
brock400_1	400	399	27	27+	27+	19	19	23	23	25	24
brock400_2	400	399	29	29+	29+	20	18	23	24	29	25
brock400_3	400	399	31	31+	31+	20	17	24	25	31	31
brock800_3	800	799	25	21+	21+	15	15	19	20	22	22
c-fat200-1	200	90	12	12	12	12	12	12	12	12	12
c-fat200-2	200	24	24	24	24	24	24	24	24	24	24
c-fat200-5	200	116	58	58	58	58	58	58	58	58	58
c-fat500-1	500	140	14	14	14	14	14	14	14	14	14
c-fat500-10	500	252	126	124+	126	126	126	126	126	126	126
c-fat500-2	500	260	26	26	26	26	26	26	26	26	26
c-fat500-5	500	128	64	64	64	64	64	64	64	64	64
c1000_9	1000	997	68	53+	52+	51	51	64	64	67	67
c2000_5	2000	1999	16	16+	16+	10	10	15	15	16	16
c250_9	250	242	44	41+	42+	35	36	44	44	44	44
c4000_5	4000	3998	18	17+	17+	12	12	16	17	18	17
c500_9	500	498	57	47+	47+	42	47	56	54	57	57
dsjc1000_5	1000	998	15	15	15	10	10	14	14	15	15
dsjc500_5	500	498	13	13	13	10	10	12	12	13	13
gen200_p0_9_44	200	197	44	44+	44+	32	32	44	44	44	44
gen200_p0_9_55	200	196	55	55	55	36	37	55	55	55	55
gen400_p0_9_55	400	388	55	43+	44+	42	44	51	51	53	52
gen400_p0_9_65	400	398	65	43+	43+	40	40	65	52	65	65
gen400_p0_9_75	400	398	75	45+	45+	45	47	75	75	75	75
hamming10-4	1024	1023	40	32+	34+	29	27	40	40	40	40
hamming8-4	256	114	16	16	16	16	11	16	16	16	16
keller5	776	770	27	24+	25+	15	15	26	27	27	27
keller6	3361	3338	59	42+	45+	32	36	39	43	59	59

Table 6: Lower bounds $l_x(G)$ and $l_x(G^*)$

36 instances have a reduction ratio of more than 5%. If we focus on these 36 instances, we find 35 instances with $\omega(G^*) \geq W$ and one (p_hat1500-1) with $\omega(G^*) < W$.

In general, it seems preferable to perform a heuristic method on G^* rather than on G. When MIN (l_b) is used, there are 25 instances with better results on G^* and only 14 instances with better results on G (for the other instances, we obtain the same results on G and G^*). Similarly, the penalty-evaporation method (l_c) is better when applied to G^* in 14 cases, and better when applied to G for 7 instances. For the last method, the situation is reversed, since $l_d(G^*) > l_d(G)$ in only two cases, while $l_d(G^*) < l_d(G)$ for 7 instances. For one of these 7 instances, we know that $\omega(G^*) < \omega(G)$, while for four other instances, we do not know whether G^* contains a maximum clique of G or not.

				dfmax		MIN		Penalty-evaporation		Imp. Penalty-evaporation	
Problem	n	n^*	W	$l_a(G)$	$l_a(G^*)$	$l_b(G)$	$l_b(G^*)$	$l_c(G)$	$l_c(G^*)$	$l_d(G)$	$l_d(G^*)$
latin_square_10.col	900	88	90	90+	81+	90	90	90	90	90	90
le450_15a.col	450	335	15	15	15	5	6	15	15	15	15
le450_15b.col	450	341	15	15	15	8	8	15	15	15	15
le450_15c.col	450	430	15	15	15	7	7	15	15	15	15
le450_15d.col	450	434	15	15	15	5	5	15	15	15	15
le450_25a.col	450	217	25	25	25	11	9	25	25	25	25
le450_25b.col	450	237	25	25	25	13	12	25	25	25	25
le450_25c.col	450	376	25	25	25	7	8	25	25	25	25
le450_25d.col	450	373	25	25	25	8	9	25	25	25	25
le450_5a.col	450	392	5	5	5	4	5	5	5	5	5
le450_5b.col	450	388	5	5	5	4	3	5	5	5	5
le450_5c.col	450	449	5	5	5	3	3	5	5	5	5
p_hat1000-1	1000	665	10	10	10	7	9	10	10	10	10
p_hat1000-2	1000	470	46	43+	41+	38	40	46	46	46	46
p_hat1000-3	1000	853	68	49+	48+	57	57	67	68	68	68
p_hat1500-1	1500	752	12	12	11	8	7	12	11	12	11
p_hat1500-2	1500	690	65	46+	48+	54	59	65	65	65	65
p_hat1500-3	1500	1263	94	53+	54+	75	81	94	94	94	94
p_hat300-1	300	245	8	8	8	7	7	8	8	8	8
p_hat300-2	300	169	25	25	25	23	20	25	25	25	25
p_hat300-3	300	279	36	36	36	31	31	36	36	36	36
p_hat500-1	500	372	9	9	9	6	8	9	9	9	9
p_hat500-2	500	257	36	36	36	29	33	36	36	36	36
p_hat500-3	500	448	50	44+	48+	42	42	49	50	50	50
p_hat700-1	700	487	11	11	11	7	7	11	11	11	11
p_hat700-2	700	324	44	44	44	38	42	44	44	44	44
p_hat700-3	700	621	62	50+	51+	55	53	62	62	62	62
san1000	1000	970	15	10+	10+	8	8	8	8	15	10
san200_0_7_1	200	30	30	30	30	16	30	17	30	30	30
san200_0_7_2	200	189	18	18+	18+	12	12	13	13	18	18
san200_0_9_1	200	70	70	48+	70	45	70	45	70	70	70
san200_0_9_3	200	198	44	44+	36+	31	30	36	43	44	43
san400_0_5_1	400	13	13	13	13	7	13	8	13	13	13
san400_0_7_1	400	390	40	22+	20+	21	20	21	20	22	22
san400_0_7_2	400	398	30	17+	17+	15	15	18	18	30	30
san400_0_7_3	400	376	22	17+	22+	12	12	17	17	22	22
san400_0_9_1	400	393	100	49+	50+	41	39	54	100	100	100
sanr200_0_7	200	197	18	18	18	14	14	18	18	18	18
sanr200_0_9	200	198	42	40+	40+	35	36	42	42	42	42
sanr400_0_5	400	395	13	13	13	10	9	13	12	13	13
sanr400_0_7	400	397	21	21	21	16	16	21	20	21	21

Table 7: Lower bounds $l_x(G)$ and $l_x(G^*)$

5 Conclusion

In this paper, we have presented a sequential elimination algorithm to compute an upper bound on the clique number of a graph. At each iteration, this algorithm removes one vertex and updates a tentative upper bound by considering the closed neighborhoods of the remaining vertices. Given any method to compute an upper bound on the clique number of a graph, we have shown, under mild assumptions, that the sequential elimination algorithm is guaranteed to improve upon that upper bound. Our computational results on DIMACS instances show significant improvements of about 60%. We have also shown how to use the output of the sequential elimination algorithm to improve the computation of lower bounds.

It would be interesting to apply the sequential elimination algorithm to other upper bound functions to see if similar trends can be observed. The development of other heuristic methods based on the sequential elimination algorithm is another promising avenue for future research.

References

[1] M. Aouchiche. *Comparaison automatisée d'invariants en théorie des graphes*. PhD thesis, École polytechnique de Montréal, 2006.

[2] D. Applegate and D. Johnson. Dfmax source code in C. A World Wide Web page. ftp://dimacs.rutgers.edu/pub/challenge/graph/solvers/.

[3] I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo. The maximum clique problem. In D. Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 4, pages 1–74. Kluwer Academic Publishers, 1999.

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

[5] P. Galinier and A. Hertz. A survey of local search methods for graph coloring. *Computers and Operations Research*, 33(9):2547–2562, 2006.

[6] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] J. Harant, Z. Ryjacek, and I. Schiermeyer. Forbidden subgraphs and MIN-algorithm for independence number. *Discrete Mathematics*, 256(1-2):193–201, 2002.

[8] A. Hertz and D. de Werra. Using tabu search for graph coloring. *Computing*, 39:345–351, 1987.

[9] D.S. Johnson and M.A. Trick, editors. *Cliques, Coloring, and Satisfiability – Second DIMACS Implementation challenge*, volume 26 of *DIMACS – Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, 11–13 October 1993.

[10] D.E. Knuth. The sandwich theorem. *Electronic Journal of Combinatorics*, 1:48, 1994.

[11] László Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, 25(1):1–7, 1979.

[12] P.M. Pardalos and J. Xue. The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328, 1994.

[13] P. St-Louis, B. Gendron, and J.A. Ferland. A penalty-evaporation heuristic in a decomposition method for the maximum clique problem. *Working paper*, Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada.

[14] G. Szekeres and H.S. Wilf. An inequality for the chromatic number of graphs. *Journal of Combinatorial Theory*, 4:1–3, 1968.