Les Cahiers du GERAD ISSN: 0711–2440

A Repeated Sequential Elimination Algorithm for Finding an Upper Bound on the Clique Number

L. Curzi, A. Hertz,
I. Lari

G–2012–80

December 2012

A Repeated Sequential Elimination Algorithm for Finding an Upper Bound on the Clique Number

Luigi Curzi

Dipartimento di Scienze Statistiche
"Sapienza" Università di Roma
Piazzale Aldo Moro 5
00185 Rome, Italy

Alain Hertz

GERAD & Département de mathématiques et de génie industriel
Polytechnique Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) Canada, H3C 3A7
alain.hertz@gerad.ca

Isabella Lari

Dipartimento di Scienze Statistiche
"Sapienza" Università di Roma
Piazzale Aldo Moro 5
00185 Rome, Italy
isabella.lari@uniroma1.it

December 2012

Les Cahiers du GERAD
G–2012–80

Abstract

In this paper a new procedure for finding an upper bound on the clique number of a given graph is described. Gendron, Hertz and St-Louis (2008) proposed a sequential elimination algorithm which, given any method that computes an upper bound on the clique number, improves upon that bound by iteratively reducing the graph. The idea of the new algorithm is to apply the sequential elimination algorithm to the given graph and then apply it again to some subgraphs in order to further improve the obtained bound. A preliminary set of computational results shows that if the new algorithm is associated with a simple but sufficiently accurate method for computing an upper bound on the clique number it can substantially improve the bounds obtained with the Gendron, Hertz and St-Louis algorithm within reasonable execution times.

Key Words: Clique number; Upper bounds.

1 Introduction

A *clique* of a graph $G = (V, E)$ is a complete subgraph of G. The maximum clique problem is the problem of finding a clique of G having maximum number of vertices and the clique number $\omega(G)$ of G is the cardinality of a maximum clique of G. It is well known that the maximum clique problem is NP-hard [6] and that also the problem of finding an approximation of the clique number is hard [9]. Therefore tight upper bounds of the clique number are useful for any exact or heuristic algorithm.

The problem of finding an upper bound on the clique number has been studied by several authors. A simple upper bound has been proposed by Amin and Hakimi [1], who proved that $(3 + \sqrt{9 - 8(n - m)})/2$ is an upper bound of $\omega(G)$, where $n = |V|$ and $m = |E|$. An upper bound can be also found by solving a continuous relaxation of an Integer Linear Program (see e.g.[8] and [1]). Another upper bound can be obtained by formulating the maximum clique problem as an unconstrained 0-1 quadratic program [8] and finding the roof dual of this problem [5]. In [2] Boros et al. proposed a generalization of the roof duality theory and applied it to the maximum clique problem showing that the obtained upper bounds strongly improve those provided by the roof dual.

A coloring of a graph is an assignment of labels or colors to the vertices of the graph in such a way that adjacent vertices have different colors. The number of colors in any graph coloring of G is an upper bound on the clique number of G. The best of such upper bounds is the chromatic number of G, i.e. the minimum number of colors in a coloring of G. The problem of finding or approximating the chromatic number of a graph is NP-hard [9], but any heuristic for the graph coloring problem provides an upper bound on the clique number. Lovász [7] introduced the function $\theta(G)$ and proved that $\omega(G) \leq \theta(G) \leq \chi(G)$. The θ function can be defined in many different ways and can be found in polynomial time by Semidefinite Programming.

Gendron, Hertz and St-Louis [4] proposed a sequential elimination algorithm for finding an upper bound on the clique number which, given an arbitrary function providing an upper bound on the clique number, iteratively reduces the graph for improving the bound. In this paper we propose a repeated sequential elimination algorithm in which the graph reducing procedure is applied in two stages in order to further improve the bound obtained at the end of the sequential elimination algorithm The paper is organized as follows: in Section 2 the Gendron, Hertz and St-Louis procedure is briefly described; in Section 3 the repeated sequential elimination procedure is proposed and in Section 4 the computational results on a set of DIMACS benchmarks are presented.

2 The sequential elimination algorithm

Given a graph G we indicate the vertex set of G as $V(G)$. Given a vertex $v \in V(G)$, the *closed neighborhood* $N_G(v)$ of v in G is the subgraph of G induced by v and the vertices of G adjacent to v. Let $h(G)$ be a function providing an upper bound on the clique number of G. The sequential elimination algorithm (SEA) proposed by Gendron et al. [4] is based on the following ideas.

- *Decomposition of the graph*: for each vertex v of the graph, the function h is computed on the closed neighborhood $N_G(v)$ of v and the maximum among the obtained upper bounds is an upper bound on the clique number of G.
- *Reduction of the graph*: the graph is iteratively reduced by deleting at each iteration the vertex whose closed neighborhood has the minimum value of the function h.

At each iteration of the algorithm an upper bound on the clique number is available and the algorithm terminates when it is not possible to improve the bound.

A pseudocode of the algorithm follows.

---

algorithm `Sequential Elimination (SEA)`

input: An undirected graph $G = (V, E)$.
output: An upper bound $h'(G)$ on the clique number of G.

begin
  let $G' := G$ and $h'(G) := 0$;
  while $h'(G) < \max_{v \in V(G')} h(N_{G'}(v))$ **do**
    choose in G' a vertex s such that $h(N_{G'}(s)) = \min_{v \in V(G')} h(N_{G'}(v))$;
    if $h(N_{G'}(s)) > h'(G)$ then **let** $h'(G) := h(N_{G'}(s))$;
    update G' by removing s and all the edges incident to s;
  return $h'(G)$;
end

At the beginning of each iteration $h'(G)$ is an upper bound on the cardinality of any clique containing one of the removed vertices and $\max_{v \in V(G')} h(N_{G'}(v))$ is an upper bound on the clique number of the current graph G'; if this value is greater than or equal to $h'(G)$, it is also an upper bound on the clique number of G. Otherwise, if $h'(G) > \max_{v \in V(G')} h(N_{G'}(v))$, $h'(G)$ is an upper bound on the clique number of G. In this case it is no longer possible to decrease this bound by further reducing the graph and the algorithm stops.

Gendron, Hertz and St-Louis proved the following theorem which ensures that, under a reasonable assumption on the function h, the bound provided by SEA is no worse than $h(G)$.

Theorem 2.1 *If h is decreasing, i.e. if $h(G) \geq h(G')$ for any induced subgraph G' of G, then $h'(G) \leq h(G)$ [4] .*

3 The repeated sequential elimination algorithm

The repeated sequential elimination algorithm (R-SEA) which is presented in this paper is a variant of SEA and it has been created to improve its performance. Unlike the original algorithm, it is divided into two distinct parts: in the first part SEA is applied to the given graph, while, in the second one, SEA is applied again to some of the generated subgraphs. In the first part the algorithm behaves as the original one but it iterates until the remaining subgraph is a clique. The cardinality of this clique is a lower bound on the clique number. In addition, at each iteration of the first part, two pieces of information are stored: the subgraph induced by the closed neighborhood of the removed vertex and the upper bound for this subgraph. These stored data are compared in the second part, trying to improve at each iteration the upper bound; in particular, the algorithm iteratively apply SEA on a stored subgraph having maximum upper bound until it is possible to improve the bound.

A pseudo-code of the algorithm follows.

---

algorithm `Repeated Sequential Elimination (R-SEA)`

input: An undirected graph $G = (V, E)$.
output: An upper bound $h''(G)$ on the clique number of G.

begin
  (first part)
  let $G' := G$, $h'(G) := 0$ and $k := 1$;
  while G' is not a clique **do**
    choose in G' a vertex s such that $h(N_{G'}(s)) = \min_{v \in V(G')} h(N_{G'}(v))$;
    if $h(N_{G'}(s)) > h'(G)$ then $h'(G) := h(N_{G'}(s))$;
    let $G_k := N_{G'}(s)$ and $U_k := h(N_{G'}(s))$;
    update G' by removing s and all the edges incident to s;
    let $k := k + 1$;
  let $G_k := G'$ and $U_k := V(G')$;
  (second part)
  order the upper bounds $U_1, ..., U_k$ in non increasing order
  and let $i_1, ..., i_k$ such that $U_{i_1} \geq \ldots \geq U_{i_k}$;
  let $h''(G) := 0$ and $j := 1$;
  while $U_{i_j} > h''(G)$ and $j \leq k$ **do**
    apply SEA to G_{i_j} and let $h'(G_{i_j})$ be the obtained upper bound;
    let $h''(G) := \max\{h''(G), h'(G_{i_j})\}$;
    let $j := j + 1$;
  return $h''(G)$;
end

The first part of the algorithm has the same computational complexity of SEA, i.e. $O(n^2 T(n, m))$, where $T(n, m)$ is the computational complexity of the procedure providing the upper bound h. The second part has complexity $O(n^3 T(n, m))$, thus the computational complexity of the overall procedure is $O(n^3 T(n, m))$.

The following results show that R-SEA correctly finds an upper bound on the clique number and, similarly to the original algorithm, if h is decreasing (see theorem 2.1) the obtained bound is no worse than $h'(G)$.

Theorem 3.1 *At the end of R-SEA, $h''(G)$ is an upper bound on the clique number of G.*

Proof. Let $s_1, \ldots, s_k$ be the vertices deleted from G during the first part of the algorithm for obtaining $G_1, \ldots, G_k$, respectively.

The subgraphs $G_1, \ldots, G_k$ are generated in such a way that:

- all cliques of G containing s_1 are induced subgraph of G_1,
- for each $j = 2, \ldots, k-1$, all cliques of G containing s_j but not $s_1, \ldots, s_{j-1}$ are induced subgraphs of G_j,
- the last subgraph G_k is a clique of G containing all vertices of G but $s_1, \ldots, s_{k-1}$.

Therefore a maximum clique of G is a maximum clique of one of the subgraphs $G_1, \ldots, G_k$. It follows that, since U_j is an upper bound on $\omega(G_j)$ for each j, at the end of the first part $h'(G) = \max\{U_1, \ldots, U_k\}$ is an upper bound on $\omega(G)$.

At iteration j of the second part, SEA is applied to the subgraph having the largest upper bound and $h''(G)$ is the maximum among the updated bounds. Hence $\max\{h''(G), U_{i_j}\}$ is an upper bound on the clique number of G. The algorithm stops when $U_{i_j} \leq h''(G)$ and then $h''(G)$ is an upper bound on the clique number and it is impossible to further improve it. □

Theorem 3.2 *If h is decreasing then at the end of R-SEA $h''(G) \leq h'(G)$.*

Proof. At the first iteration of the second part, $h''(G) = 0$ and SEA is applied to the subgraph G_{i_1}. Since h is decreasing

$$h'(G_{i_1}) \leq U_{i_1} = h'(G).$$

Therefore at the end of the first iteration

$$h''(G) = \max\{0, h'(G_{i_1})\} = h'(G_{i_1}) \leq h'(G).$$

At iteration j of the second part

$$h'(G_{i_j}) \leq U_{i_j} \leq U_{i_1} = h'(G).$$

Suppose by induction that at the beginning of this iteration $h''(G) \leq h'(G)$. It follows that the updated value of $h''(G)$, i.e. $\max\{h''(G), h'(G_{i_j})\}$, is still less than or equal to $h'(G)$. □

4 Experimental results

The repeated sequential elimination algorithm has been compared to the original sequential elimination algorithm on a set of DIMACS benchmarks for the maximum clique problem. In our experiments we used three methods for finding the upper bound in the induced subgraphs: a Linear Coloring coloring algorithm and DSATUR [3] that are well-known heuristics for the coloring problem, and a procedure based on the degree sequence of the vertices of the given graph. For each test problem we considered two performance indicators:

- the error, i.e. the difference between the upper bound and the clique number divided by the clique number;
- the execution time of the algorithm.

We chose to run the algorithms on graphs having at most 200 vertices in order to perform a larger number of tests, maintaining the execution times within certain limits. All the experiments were performed on a PC equipped with an AMD Turion 64 X2 dual-core 1.80 Ghz, with 2 GB of RAM and the operating system Debian GNU/Linux and the algorithms have been implemented in Python.

The experimental results are shown in Tables 1 and 2 where R-SEA is compared to SEA. In Table 1 we show the errors and in Table 2 the execution times of the two procedures. We have not included the results for the executions that have breached the time limit of 14000 seconds.

In all the considered cases the bound obtained with the new algorithm is closer to the optimum than the bound produced by the original algorithm and for all three cases (Degree Sequence, Linear Coloring, DSATUR) the average error of R-SEA is about half the error of SEA. Even using a function quite simple and not very accurate as the one based on the degree sequence, on low density graphs, as for example In200-40-13 having density 40%, or on graphs with sufficiently high optimum, as for example In200-60-35, the error of R-SEA is exactly 0; this does not happen with SEA. Using the Linear Coloring algorithm things get even better and the average error is less than half the error achieved by the execution of SEA (0,10 against 0,23). Also using DSATUR, the results are very good (0.04 against 0.14), but for some graphs, in particular for

very dense ones, we were not able to find the upper bound since the execution time exceeded the maximum time of 14000 seconds.

As shown in Table 2, regarding the execution times there is no match, SEA is much faster than R-SEA: about 9 times with Degree Sequence, 6 times with Linear Coloring and 1,5 with DSATUR. In the latter case we counted only graphs for which the execution of R-SEA terminated within the time limit.

At this point you might think that the game not worth the effort due to the excessive increase of the execution times of R-SEA with respect to SEA. But giving a closer look at the tables we can see that the average error obtained using R-SEA with the Linear Coloring function is slightly more than half the error obtained running SEA with DSATUR (0.10 against 0.18); comparing these two cases, the time spent by R-SEA is less than the one spent by SEA: the new algorithm spent on average 303 seconds as opposed to 3176 seconds of the sequential elimination algorithm.

Table 1: in this table the errors obtained by the execution of SEA and R-SEA are compared. In the last two columns some values are missing because the algorithm in those cases took more than 14000 seconds.The last bracketed value of column DSATUR/SEA shows the average error of the cases with execution times less than 14000.

	Degree Sequence		Linear Coloring		DSATUR	
Graph	SEA	R-SEA	SEA	R-SEA	SEA	R-SEA
In200-40-13	1.29	0.00	0.08	0.00	0.00	0.00
In200-40-22	0.35	0.00	0.00	0.00	0.00	0.00
In200-40-33	0.00	0.00	0.00	0.00	0.00	0.00
In200-40-40	0.00	0.00	0.00	0.00	0.00	0.00
In200-40-55	0.00	0.00	0.00	0.00	0.00	0.00
In200-60-15	3.36	1.37	0.73	0.24	0.53	0.13
In200-60-35	0.85	0.00	0.00	0.00	0.00	0.00
In200-60-40	0.61	0.00	0.00	0.00	0.00	0.00
In200-60-50	0.26	0.00	0.00	0.00	0.00	0.00
In200-60-75	0.00	0.00	0.00	0.00	0.00	0.00
In200-80-25	3.73	2.55	0.85	0.58	0.62	-
In200-80-40	1.94	1.19	0.25	0.04	0.16	-
In200-80-55	1.13	0.59	0.00	0.00	0.00	0.00
In200-80-70	0.63	0.20	0.00	0.00	0.00	0.00
In200-80-80	0.42	0.04	0.00	0.00	0.00	0.00
MANN-a9.clq	1.31	1.06	0.13	0.13	0.19	0.13
brock200-1.clq	3.71	2.19	1.00	0.52	0.86	-
brock200-2.clq	2.58	0.67	0.58	0.08	0.42	0.00
brock200-3.clq	3.27	1.40	0.80	0.27	0.67	0.20
brock200-4.clq	3.53	1.71	0.88	0.41	0.76	0.24
c-fat200-1.clq	0.00	0.00	0.00	0.00	0.00	0.00
c-fat200-2.clq	0.00	0.00	0.00	0.00	0.00	0.00
c-fat200-5.clq	0.00	0.00	0.00	0.00	0.00	0.00
hamming6-2.clq	0.63	0.44	0.00	0.00	0.00	0.00
hamming6-4.clq	1.00	0.00	0.25	0.00	0.25	0.00
johnson16-2-4.clq	7.50	5.00	0.63	0.50	0.63	0.50
johnson8-2-4.clq	1.00	0.00	0.25	0.00	0.25	0.00
johnson8-4-4.clq	1.71	0.86	0.07	0.00	0.07	0.00
san200-0.7-1.clq	2.10	1.90	0.07	0.00	0.00	0.00
san200-0.7-2.clq	5.22	4.67	0.28	0.00	0.06	-
san200-0.9-1.clq	1.03	0.70	0.10	0.00	0.00	0.00
san200-0.9-2.clq	1.47	1.10	0.22	0.08	0.13	-
san200-0.9-3.clq	2.39	1.91	0.55	0.41	0.32	-
AVG:	1.61	0.90	0.23	0.10	0.18	0.04
					(0.14)	

Table 2: in this table the execution times in seconds obtained by SEA and R-SEA are compared. In the last column some values are missing because the algorithm in those cases took more than 14000 seconds. The last bracketed value on column DSATUR/SEA shows the average execution time of the cases with execution times less than 14000. Notice that in some cases in which the error is equal to zero, the execution times of R-SEA and SEA are very similar.

	Degree Sequence		Linear Coloring		DSATUR	
Graph	SEA	R-SEA	SEA	R-SEA	SEA	R-SEA
In200-40-13	11.08	29.94	21.19	23.40	497.72	508.22
In200-40-22	12.38	27.10	21.70	21.88	538.21	528.31
In200-40-33	13.70	13.58	22.94	23.01	597.81	590.95
In200-40-40	14.12	13.97	24.14	23.99	667.19	659.90
In200-40-55	15.68	15.43	28.50	28.26	942.39	948.10
In200-60-15	19.96	153.57	46.17	275.57	1886.07	8316.71
In200-60-35	24.20	152.67	57.64	59.87	2348.68	2448.99
In200-60-40	24.81	148.96	57.98	60.24	2386.39	2486.10
In200-60-50	26.77	129.98	60.15	62.16	2548.47	2653.98
In200-60-75	35.18	35.69	67.46	69.35	3293.86	3425.54
In200-80-25	24.21	475.70	76.44	957.23	5844.98	-
In200-80-40	28.70	487.57	77.33	1010.35	5776.70	-
In200-80-55	30.96	473.43	114.58	119.37	7726.06	7989.33
In200-80-70	34.26	490.63	115.60	120.41	7931.74	8325.59
In200-80-80	33.60	472.57	116.17	121.12	8129.43	8541.14
MANN-a9.clq	0.04	0.34	0.24	1.26	4.65	16.62
brock200-1.clq	33.17	445.52	70.63	936.48	4862.42	-
brock200-2.clq	18.80	67.85	33.42	120.74	1047.01	2918.16
brock200-3.clq	25.13	160.05	50.58	304.87	2098.92	9356.67
brock200-4.clq	25.31	243.01	58.06	445.21	2753.77	17256.72
c-fat200-1.clq	1.08	1.28	2.31	2.70	33.17	38.64
c-fat200-2.clq	3.64	3.62	7.88	7.84	179.94	182.22
c-fat200-5.clq	16.33	20.93	34.35	44.91	1513.69	2076.96
hamming6-2.clq	0.03	1.05	0.46	1.34	3.22	49.44
hamming6-4.clq	0.01	0.22	0.25	0.31	3.02	3.66
johnson16-2-4.clq	0.19	20.52	10.45	43.63	405.64	2248.35
johnson8-2-4.clq	0.00	0.03	0.02	0.04	0.18	0.41
johnson8-4-4.clq	0.03	2.67	1.22	3.77	31.79	76.31
san200-0.7-1.clq	44.62	70.73	77.20	468.64	4560.25	4872.75
san200-0.7-2.clq	42.46	71.44	81.60	1085.66	4218.99	-
san200-0.9-1.clq	38.77	511.93	81.10	1016.20	13425.85	13412.87
san200-0.9-2.clq	33.30	512.97	90.62	1231.59	8502.28	-
san200-0.9-3.clq	30.04	495.80	96.46	1318.26	10056.08	-
AVG:	20.08	174.27	48.63	303.32	3176.26 (2427.97)	3701.21

5 Conclusions and future works

The experimental results show that R-SEA is a valid procedure to improve the upper bound on the clique number given by SEA, mainly if it is associated with simple but sufficiently accurate upper bound functions for the maximum clique problem. However the running times are very high and the efficiency of the procedure and its implementation must be improved.

References

[1] Amin A.T. , Hakimi S.L. (1972), Upper bounds of the order of a clique of a graph, SIAM J. Appl. Math., vol. 22, pp. 569–573.

[2] Boros E., Lari I., Simeone B. (2004), Block linear majorants in quadratic 0-1 optimization, Discrete Applied Mathematics, vol. 145, pp. 52–71.

[3] Brélaz, D. (1979), New methods to color the vertices of a graph, Communications of the Assoc. of Comput. Machinery 22, pp. 251–256.

[4] Gendron B., Hertz A., St-Louis P. (2008), A sequential elimination algorithm for computing bounds on the clique number of a graph, Discrete Optimization, vol. 5, pp. 615–628.

[5] Hammer P. L., P. Hansen, Simeone B. (1984), Roof duality, complementation and persistency in quadratic 0-1 optimization, Mathematical Programming, vol. 28, pp. 121–155.

[6] Karp R. M., (1972), Reducibility among combinatorial problems, in Complexity of Computer Computations: Proc. of a Symp. on the Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, Eds., The IBM Research Symposia Series, New York, NY: Plenum Press, pp. 85–103.

[7] Lovász L. (1979), On the Shannon capacity of a graph, IEEE Transactions on Information Theory, vol. 25, pp. 1–7.

[8] Pardalos P. M., Xue J. (1992), The Maximum Clique Problem, Journal of Global Optimization, vol. 4, pp. 301-328.

[9] Zuckerman, D. (2007), Linear degree extractors and the inapproximability of Max Clique and Chromatic Number, Theory of Computing, vol. 3, pp. 103–128.