Finding a maximum independent set in time $O(2^{n/4})$

J. M. Robson

January 2001

Abstract

We improve previous algorithms for finding the size of the largest independent set of a graph in three ways: firstly a more detailed analysis of the possible local structures around a vertex where the algorithm will branch recursively, much of the most detailed analysis being done by computer, secondly more use of auxiliary functions which consider independent sets subject to certain extra conditions and thirdly a better analysis of the improvements achieved by a dynamic programming variant. These improvements reduce the constant c in the $O(2^{cn})$ time bound to less than 0.25.

1 Introduction

We consider the NP-hard problem of finding a maximum independent (or stable) set in a graph. As in previous papers ([4], [3]) we concentrate on determining the size of such a set, that is the *stability number* of the graph. Actually finding a set of this size cannot be much harder than finding the size and, in fact, our algorithms for the size can easily be modified to give algorithms of the same complexity for finding one set.

Previous papers have given algorithms with proven bounds on the run time for n vertex graphs of $O(a^n)$ with a < 2 often expressed as $O(2^{cn})$, first for c < 1/3 ([4] and then c » 0.304 ([2]) and c » 0.276 ([3]. In this paper we reduce this bound to just under c=0.25 (a » 1.18882). [1] obtains a better bound for the case of graphs of degree bounded by 3.

1.1 Notation

G is a graph (V,E). We write G-v or G-S for the subgraph induced on V-v or V-S for v Î V or S Ì V.
We write N(v) for the set of neighbours of a vertex v and [`N](v) for N(v)+v. $N^2(v)$=N(N(v))-[`N](v) and $[`N]^2(v)$=[`N]([`N](v)). These notations are also used with a set of vertices rather than the single vertex v.
We write $d_G(v)$ for the degree of v in G and omit the G when no ambiguity can arise.
We often abbreviate maximum independent set as m.i.s.

1.2 Structure

1.2.1 Degree less than 9 plus memorisation

We start by describing an algorithm stab computing the stability number of graphs of degree at most 9 and whose run time is $O(a^n)$ for a » 1.202431, that is 2^{cn} for c » 0.265955 on graphs of size n. Then we apply the technique of *memorisation* or dynamic programming used in [3] to deduce an algorithm running in time $O(a¢^n)$ for a¢ » 1.188805 ($O(2^{c¢n})$ for c¢ » 0.249512), still on graphs of degree at most 9. Finally we give an algorithm achieving the same bound a¢ on general graphs which has a simple structure as long as the degree is more than 9 and calls the second algorithm whenever it is applied to a subgraph of degree £ 9.

1.2.2 Assumption that best case is chosen

Except for trivial cases, the algorithm considers the neighbourhood of a chosen vertex v, decides that there must be a maximum independent set obtained in one of m ways by including certain vertices IN_i in the independent set and excluding certain others EX_i, finds recursively the size of the stable set obtained in each way $|IN_i|$+stab(G-IN_i-EX_i) and takes the maximum of these as i ranges from 1 to m. There are a large number of cases depending on the structure found in the neighbourhood of v. For each case, we claim that the time taken is bounded by $c_{case}2^{cn}$ for a constant c_{case} £ 1 and prove this recursively by considering the cases which might arise in the recursive calls. The algorithm always chooses the case with the smallest value of c_{case}, that is in considering any case we can assume that no ``better'' cases are present in the graph although it is convenient to present the cases in an order different from the numerical order of the constants.

1.2.3 Constrained auxiliary functions

The detailed structure of cases distinguished by the algorithm is somewhat simplified by isolating themes which can be exploited in a number of places.

One situation which arises frequently is illustrated by the simple case of a vertex v of degree 3 and having neighbours w, x and y. If any independent set contained none of these four vertices or contained only w or x or y, it could be modified to give an independent set at least as large by adding v and dropping w or x or y if they were present. Hence we can say that there is a m.i.s. either containing v (and hence not w, x or y) or not containing v but containing at least two of {w,x,y}. Hence in a recursive call stab(G-v) we are not concerned by the result unless every m.i.s. of G-v contains at least two of w, x and y. If this condition is satisfied our function must return the true stability number; otherwise we require only that the value returned is no greater than the true stability number. Hence we define a first auxiliary function stab¢ which has as parameters a graph G and a number i ³ 1 of pairs each consisting of a subset S_i of the vertices of G and an integer n_i; if every maximum independent set of G contains at least n_i vertices from S_i for every i, the function returns the stability number of G; otherwise it returns a value less than or equal to the stability number.

Next we define a second auxiliary function stab$^{¢¢}$ for the even more constrained case where we are interested only in independent sets containing *exactly* one of two specified vertices. Precisely: stab$^{¢¢}$(G,x,y) is only defined where x and y are two vertices of G not connected by an edge; if every maximum independent set of G contains either x or y but not both, the function returns the stability number of G; otherwise it returns a value less than or equal to the stability number.

Finally we remark that the gain from the memorisation technique of Section 4 depends on the fact that all graphs considered are induced subgraphs of the original graph but occasionally it is useful to consider a graph obtained by adding edges between one vertex and a number of others. Accordingly we define stab$^{¢¢¢}$(G,x) which returns the stability number of G but does so in such a way that every call of stab, stab¢ or stab$^{¢¢}$ has as parameter an induced subgraph of G which does not contain the vertex x.

For each case of each of these auxiliary functions we define a constant a_{case} and claim that the computation time is £ $a_{case}2^{cn}$. All the constants a_{case} £ 1 except for those associated with the function stab¢¢¢.

We give equations for each of these constants. The form of these equations is that of C assignment statements because our latex file will be directly incorporated into a C program which evaluates the constants and finds the smallest value of c for which the argument is valid. In this C program, alpha is a pointer into a table containing powers of a and max is a function such that max(n,x_1,¼,x_n) is the maximum of x_1,¼,x_n (for the values of n used). The constants for the function stab all have names beginning c and those for the auxiliary functions begin a. The rest of the names are obtained in an obvious way from the labels of the enumerations in Sections 2 and 3.

1.2.4 Basic idea of min degree and max neighbour degree and not regular

The normal action of the algorithm is to choose a pair A and B of neighbouring vertices so as to minimise d(A), ensure that d(B) > d(A) and lastly maximise the number of neighbours of A with the same degree d(A). For reasons discussed in [3], the analysis can ignore the case of disconnected and regular graphs, so we can always find such a pair.

2 Unconstrained

1. **Dominance**
   If there are two vertices A and B such that [`N](A) Ì [`N](B), we say that A *dominates* B and there is a maximum independent set not containing B. Calculate stab(G)=stab(G-B).

   a. c1a=alpha[-1];(= 0.831648).

   b. Note that a vertex of degree 1 dominates its neighbour and that, in this case the call stab(G-B) has A an isolated vertex giving c1b = c9*alpha[-1] ; = 0.691639.

   c1=c1a;(= 0.831648).

2. **Degree 2**
   If a vertex A has degree 2, consider its neighbours, say B and B¢. There is a m.i.s. containing A (and neither B nor B¢) or both B and B¢. We write x for |[`N]$^2$(A)-[`N](A)| and y for |[`N]$^3$(A)-[`N]^2(A)|

   a. **B and B¢ are connected**
      No independent set can contain B and B¢ and we have stab(G)=1+stab(G-[`N](A)
      c2a=alpha[-3];(= 0.575200).
      Otherwise the independent set containing A can be assumed to contain two elements of |[`N]$^2$(A)-[`N](A)| since, if not, it cannot be larger than all independent sets containing B and B¢. this gives us
      stab(G)=max(1+stab¢(G-[`N](A),([`N]^2(A)-[`N](A),2)),2+stab(G-[`N]^2(A)).

   b. if x £ 1, the stab¢ call is vacuous and we have
      c2b=alpha[-3];(= 0.575200).

   c. x=2
      The stab¢ call can be replaced by 2+stab(G-[`N]^3(A)),

      i. **if y=1**
         the stab call can be dropped giving
         c2ci=alpha[-6];(= 0.330855).

      ii. **else**
         c2cii=alpha[-7]+alpha[-5];(= 0.672986).

      c2c=max(2,c2ci,c2cii);(= 0.672986).

   d. x=3
      c2d=a3*alpha[-3]+alpha[-6];(= 0.728686)

   e. x=4
      c2e=a4*alpha[-3]+alpha[-7];(= 0.727813)

   f. x=5
      c2f=a5*alpha[-3]+alpha[-8];(= 0.724186)

   g. x=6
      c2g=alpha[-3]+alpha[-9];(= 0.765508)

   h. x > 6
      c2h=alpha[-3]+alpha[-10];(= 0.733469)

   Hence c2=max(7,c2a,c2b,c2c,c2d,c2e,c2f,c2g);(= 0.765508).

3. **Degree 3**
   Similarly to the previous case, if d(A)=3, then there is a m.i.s. containing either A or at least two neighbours of A. By the choice of A and B, any neighbour of A with degree 3 has a neighbour with degree no more than d(B).

   a. **In N(A) (for some vertex A with d(A)=3) a vertex v is connected to the other two vertices**
      A dominates v so stab(G)=stab(G-v) with d_{G-v}(A)=2.
      c3a=c2*alpha[-1];(= 0.636633).

   b. **All vertices of N(A) have degree at least 4**
      stab(G)=max(1+stab(G-[`N](A)),stab¢(G-A,(N(A),2))), where in the second call all vertices of G-A have degree at least 3
      c3b=alpha[-4]+a3i*alpha[-1];(= 0.861185).
      Or if there are two non-adjacent vertices of degree 3
      c3bx=c3*alpha[-4]+a3i*alpha[-1];(= 0.794781).

c. **Some vertex A of degree 3 has neighbours v,w,x with d(v) > 3 and (w,x) an edge**
stab(G)=max(1+stab(G-[`N](A)),stab¢(G-[`N](v),({w,x},1)))
c3c=alpha[-4]+a1*alpha[-5];(= 0.809219).

d. **A has neighbours v,w,x with d(v)=d(w)=3**
We must have d(x) > 3 by the choice of A.
stab(G)=max(1+stab(G-[`N](v)),stab¢(G-v),(N(v),2))) where, in the first call d(w) £ 2 and in the second all vertices have degree ³ 2
c3d=max(3,c9,c1b,c2)*alpha[-4]+max(3,a3e,a3f,a3h)*alpha[-1];(= 0.833059).
The situation now is symmetric and we change our notation: A has a neighbour also of degree 3 and we call these two vertices A_1 and A_2; the two other neighbours of A_i we call $B_{i,j}$ (j=1,2) and the four vertices $B_{i,j}$ are distinct.

e. **At most one edge ($B_{i,1}$,$B_{i,2}$)**
W.l.o.g. we suppose that ($B_{2,1}$,$B_{2,2}$) is not an edge.

i. **Some $B_{i,j}$ has degree £ 4**
stab(G)=max(1+stab(G-[`N]($A_{i¢}$)),stab¢(G-$A_{i¢}$),(N($A_{i¢}$),2)))
c3ei=max(4,c9,c1b,c2,c3)*alpha[-4]+max(3,a3e,a3f,a3h)*alpha[-1];(= 0.847188).

ii. **Some $B_{i,j}$ has degree ³ 7**
stab(G)=max(1+stab(G-[`N]($B_{i,j}$)),stab(G-$B_{i,j}$))
c3eii=max(3,c9,c1b,c2)*alpha[-8]+c2*alpha[-1];(= 0.826941).
In the remaining cases we write e for the number of edges ($B_{1,1}$,$B_{2,j}$) assuming w.l.o.g. that this quantity is at least as great for $B_{1,1}$ as for $B_{1,2}$.

iii. **d($B_{1,1}$)=5**

A. **e=0**
stab(G)=max(1+stab(G-[`N]($A_2$)),stab¢(G-$A_2$,({$A_1$,$B_{2,1}$,$B_{2,2}$},2))) where d($B_{1,1}$) < 5 in G-[`N](A_2) and, in G-A_2, d(A_1)=2 and A_1 has no common neighbours with $B_{2,1}$ or $B_{2,2}$.
c3eiiiA=max(5,c9,c1b,c2,c3,c4)*alpha[-4]+a3hiy*alpha[-1];(= 0.797866).

B. **e > 0**
stab(G)=max(1+stab(G-[`N](A_2)),stab¢(G-A_2,({A_1,$B_{2,1}$,$B_{2,2}$},2))).
where d($B_{1,1}$) £ 3 in G-[`N](A_2)
c3eiiiB=max(4,c9,c1b,c2,c3)*alpha[-4]+max(2,a3f,a3h)*alpha[-1];(= 0.847188).

c3eiii=max(2,c3eiiiA,c3eiiiB);(= 0.847188).

iv. **d($B_{1,1}$)=6**

A. **e £ 1**
stab(G)=max(1+stab(G-[`N]($B_{1,1}$)),stab(G-$B_{1,1}$)) where 1 £ d($A_2$) £ 2 in G-[`N]($B_{1,1}$) and d(A_1)=2 in G-$B_{1,1}$.
c3eivA=max(2,c1b,c2)*alpha[-7]+c2*alpha[-1];(= 0.847267).

B. **e=2**
stab(G)=max(1+stab(G-[`N]($A_2$)),stab¢(G-$A_2$,({$A_1$,$B_{2,1}$,$B_{2,2}$},2))) where d($B_{1,1}$)=3 in G-[`N](A_2).
c3eivB=c3*alpha[-4]+max(2,a3f,a3h)*alpha[-1];(= 0.847188).

c3eiv=max(2,c3eivA,c3eivB);(= 0.847267).

c3e=max(4,c3ei,c3eii,c3eii,c3eiv);(= 0.847267).

f. **Two edges ($B_{1,1}$,$B_{1,2}$) and ($B_{2,1}$,$B_{2,2}$)** For each $B_{i,j}$ we define d¢($B_{i,j}$) as the degree of $B_{i,j}$ in the graph G-[`N]($A_{i¢}$) (i¢ ¹ i) and then we define m as the minimum of the four d¢($B_{i,j}$). In the next three cases i and j refer to (one of) the $B_{i,j}$ giving this minimum.

i. **m £ 3**
stab(G)=max(1+stab(G-[`N]($A_{i¢}$)),1+stab¢(G-$A_{i¢}$,(N($A_{i¢}$),2)))
c3fi=c3*alpha[-4]+a3e*alpha[-1];(= 0.808342).

ii. **m=4**
In this and the following subcase we use the same calculation:
stab(G)=max(1+stab(G-[`N]($A_{i¢}$)),1+stab¢(G-[`N](A_i),($B_{i¢,1}$,$B_{i¢,2}$),1)))
c3fii=c4*alpha[-4]+a1e*alpha[-4];(= 0.828620).

iii. **m > 4** c3fiii=alpha[-4]+a1ex*alpha[-4];(= 0.794903).

c3f=max(3,c3fi,c3fii,c3fii);(= 0.828620).

c3=max(6,c3a,c3b,c3c,c3d,c3e,c3f);(= 0.861185).
Or if there are two non-adjacent vertices of degree 3
c3x=max(6,c3a,c3bx,c3c,c3d,c3e,c3f);(= 0.847267).

4. **Degree 4**
For degrees 4 to 7, apart from a few special cases, we always use the same method. Choose a vertex of minimum degree d and having the minimum possible number of neighbours of degree d. Call its neighbours B_1 to B_d where the order is chosen in a way to be described below. We use refinements on the simple calculation:
stab(G)=1+max(stab(G-[`N](A)),$max_{i=1}^{d-1}$(stab¢(G-[`N](B_i)-$È_{j=1}^{i-1}B_j$),($È_{j=i+1}^{d}B_j$,1))). A range of refinements may be used depending on the structure of the graph around A, more precisely on the subgraph G_B induced on {B_1,¼B_d}, on the degrees of all B_i and, in some critical cases, the numbers of common neighbours of pairs and triples of elements of B.

a. **The minimum degree in G-[`N](A)**
If m > 1 vertices B_i have degree equal to d, then , by the choice of A, each of them has at least m-1 neighbours different from A and of the same degree d. If any of these B_i does not have m-1 neighbours in G_B whose degree in G was d, then G-[`N](A) has a vertex of degree at most d-1 and the same is true also for a G-[`N](B_j) if (B_i,B_j) is and edge and B_i and B_j have no common external neighbours.

b. **The minimum degree in G-[`N](B_i)-$È_{j=1}^{i-1}B_j$**

In G-[`N](B_i)-$È_{j=1}^{i-1}B_j$ we have an upper bound on the degree of each B_j ($j > i$) excluding any neighbours of B_i. This is $d_G(B_j)$ minus one for A and for each B_k which is a neighbour of B_j and either has $k < i$ or is also a neighbour of B_i.

The c_m (m £ 8) which can be justified by this argument is only used if it is smaller than the a_m suggested by the stab¢(,($È_{j=i+1}^{d}B_j$,1)) option.

c. **Neighbours of previous B_j**

For $i > 1$, if no maximum independent set is included in those considered previously, then some maximum independent set includes at least two neighbours of each B_j for $j < i$. We have an upper bound, say b, on the number of such neighbours in the graph G-[`N](B_i)-$È_{j=1}^{i-1}B_j$. This gives the possibility of using a constant corresponding to the condition ``2 out of b''; or ``1 out of b'' depending on whether B_i is itself a neighbour of B_j. Again we use the smallest such constant if it is smaller than one already justified or we use a constant corresponding to two conditions with disjoint sets if the information on common external neighbours justifies it.

d. **Explicit choice of a vertex after i**

We can replace stab¢(G-[`N]($B_i$)-$È_{j=1}^{i-1}B_j$),($È_{j=i+1}^{d}B_j$,1) by $å_{j=i+1}^{d}$ stab($G$-[`N]({B_i,B_j})-$È_{k=1}^{j-1}B_k$)) and then further replace some or all of these stab calls by stab¢ calls using the fact that there is a m.i.s. containing two neighbours of every B_k (i ¹ k < j) not adjacent to B_i or B_j.

e. **Last three neighbours**

The last two terms in the $\max_{i=1}^{d-1}$ corresponding to taking at least two from {B_{d-2},B_{d-1},B_d} can be replaced by stab¢(G-$È_{j=1}^{d-3}B_j$-N(B_{d-2})ÇN({B_{d-1},B_d})-N(B_{d-1})ÇN(B_d),({B_{d-2},B_{d-1},B_d},1)).

The bound provable by the use of these techniques on each of the recursive calls may depend on the order of the B_i. For a given structure around A, we evaluate the bound for each of the d! orders and the algorithm chooses an order giving the lowest result.

The worst case bound on the performance is computed by carrying out this choice for each possible structure around A. The possible structures are precomputed and one example of each isomorphism class is inserted into a list and the function evaluate() calculates the constant for each element of the appropriate list and returns the maximum. In all cases we bound the number of common neighbours of two elements B_i and B_j of B by the observations that (1) if (B_i,B_j) is an edge, B_i has a neighbour not in [`N]($B_j$), since otherwise dominance would occur and (2) if ($B_i$,$B_j$) is not an edge, $B_i$ has two neighbours not in [`N](B_j), since otherwise $d_{G-[`N](B_j)}(B_i)$ £ 1 and so a m.i.s. containing B_j can be assumed to contain also B_i; hence we could consider stable sets containing either B_i or neither of B_i,B_j; for c_d this would give $a^{-(d+1)}+a^{-2}c_{d-2}$.

Initially the possible structures listed have only the information about the induced subgraph and the degree of each B_i; as the program progresses, structures found to give the worst case are gradually refined by the addition of information about common neighbours of pairs and then triples of vertices B. The final number of structures is greatest at more than 69000 for d=7.

a. **A vertex B in N(A) has d(B) ³ 8**

stab(G)=max(stab(G-B),1+stab¢(G-[`N](B),(N(A)-B,1)))
c4a=alpha[-1]*c3+alpha[-9]*a2;(= 0.896284).

b. **Subdominance**

If N(A)={B_1,B_2,B_3,B_4} and there are edges (B_1,B_2) and (B_1,B_3), there is a m.i.s. containing either B_4 and one other B_i or neither B_1 nor B_4.

i. **there is also an edge (B_2,B_3)**

There is a m.i.s. containing either A or B_4
stab(G)=1+max(stab(G-[`N](A)),stab(G-[`N](B_4)))
c4bi=2*alpha[-5];(= 0.795661).

ii. **else**

stab(G)=max(1+stab¢(G-[`N](B_4),({B_1,B_2,B_3},1)),stab(G-B_1-B_4)), where d(A)=2 in the call of stab and {B_1,B_2,B_3} is not a triangle in the stab¢ call.
c4bii=alpha[-5]*a2y+alpha[-2]*c2;(= 0.902169).

c4b=max(2,c4bi,c4bii);(= 0.902169).

c. **Special case**

If all vertices in N(A) have degree 5 and no two of them are connected or have a common neighbour other than A, choose two of them arbitrarily to be B_2 and B_3
stab(G)=max(1+stab(G-[`N](A)),1+stab(G-[`N](B_1)),2+stab¢(G-B_1-[`N]({$B_2$,$B_3$}),N($B_1$-A,2)),2+stab¢(G-$B_1$-$B_3$-[`N]({B_2,B_4}),N(B_1-A,2),N(B_3-A,2)),2+stab¢(G-B_1-B_2-[`N]({$B_3$,$B_4$}),N($B_1$-A,2),N($B_2$-A,2))) where in the call of stab(G-[`N](B_1)) there are two non adjacent degree 4 vertices (B_2 and B_3 for instance).

i. **A vertex v of N^2(A) has $d_{G-[`N](\{B_2,B_3\})}$(v)=1**

Take v's common neighbour with A to be B_4. The call stab(G-[`N]({$B_2$,$B_3$})) can be replaced by 1+stab(G-[`N]({B_2,B_3})-v-B_4)
c4ci=alpha[-5]+alpha[-6]*c4+alpha[-14]*a4+2*alpha[-13]*a4x;(= 0.906128).

ii. **Otherwise**

In the call stab¢(G-B_1-[`N]({B_2,B_3}),N(B_1-A,2)), there is no isolated vertex in N(B_1)-A
c4cii=alpha[-5]+alpha[-6]*c4+alpha[-12]*max(2,a4b,a4c)+2*alpha[-13]*a4x;(= 0.930032).

c4c=max(2,c4ci,c4cii);(= 0.930032).

d. **Otherwise**

c4d=max(2,evaluate(degree4),alpha[-5]+alpha[-2]*c2);(= 0.936534). (where the last term in the max corresponds to the discussion preceding the case c4a.

c4=max(4,c4a,c4b,c4c,c4d);(= 0.936534).

5. **Degree 5**

a. **A vertex B in N(A) has d(B) ³ 9**
stab(G)=max(stab(G-B),1+stab(G-[`N](B)))
c5a=alpha[-1]*c4+alpha[-10];(= 0.937136).

b. **Subdominance** (edges (B_1,B_i) for i=2,3,4)
stab(G)=max(1+stab(G-[`N](B_5)),stab(G-B_1-B_5)), where d(A)=3 in the second call of stab.
c5c=alpha[-6]+alpha[-2]*c3;(= 0.926484).

c. **Some B_i and B_j have a common neighbour X outside [`N](A) with d(X) ³ 8**
stab(G)=max(stab(G-X),1+stab(G-[`N](X))) where d(A) is 4 in the first call and 2 in the second.
c5e=alpha[-1]*c4+alpha[-9]*c2;(= 0.924549).

d. **Otherwise**
c5d=max(2,evaluate(degree5),alpha[-6]+alpha[-2]*c3);(= 0.974557).

c5=max(4,c5a,c5c,c5d,c5e);(= 0.974557).

6. **Degree 6**

   a. **Subdominance** (edges (B_1,B_i) for i=2,3,4,5; similar to case c4b)
   c6b=alpha[-7]+alpha[-2]*c4;(= 0.922899).

   b. **Otherwise**
   c6e=max(2,evaluate(degree6),alpha[-7]+alpha[-2]*c4);(= 0.999984).

   c6=max(2,c6b,c6e);(= 0.999984).

7. **Degree 7**: c7=max(2,evaluate(degree7),alpha[-8]+alpha[-2]*c5);(= 0.988848)

8. **Degree 8**: A has a neighbour B with degree 9.
   stab(G)=max(1+stab(G-[`N](B)),stab(G-B)) where d_{G-B}(A)=7
   c8=alpha[-10]+alpha[-1]*c7;(= 0.980643)

9. **Unconnected**: if G is not connected, we simply call stab on each connected component and sum the results.

   a. if there is a connected component consisting of a single isolated vertex v, stab(G)=1+stab(G-v)
   c9a=alpha[-1];(= 0.831648).

   b. otherwise time £ k $\max_{i=2}^{n-2}(a^i+a^{n-i})$ £ $a^n(a^{-2}+a^{2-n})$, giving c9b=alpha[-2]+alpha[-12];(= 0.801104) since we can assume n ³ 14.

   c9=max(2,c9a,c9b);(= 0.831648).

10. **Miscellaneous cases** If the graph is regular we can simply choose any vertex v and use stab(G)=max(1+stab(G-[`N](v)),stab(G-v)). [3] explains why these cases can be ignored in the timing analysis.

    If any pair of vertices v and w are not connected but |N(v)-N(w)| £ 1, there is a m.i.s. either containing both v and w or not containing w. Thus we can use stab(G)=max(2+stab(G-[`N](v,w)),stab(G-w)) if appropriate. If in addition |N(w)-N(v)| £ 1, we have the better possibility stab(G)=max(2+stab(G-[`N](v,w)),stab(G-v-w)). We can improve the calls stab(G-w) and stab(G-v-w) to stab¢(G-w,(N(v),2)) and stab¢(G-v-w,(N(v)ÈN(w),3)) respectively.

3 Constrained

1. **One out of two**
   Suppose that the set S is {A,B} with d(A) £ d(B).

   a. d(A)=0
   stab¢(G,(S,1))=1+stab(G-A)
   a1a=alpha[-1];(= 0.831648).

   b. d(A)=1
   stab¢(G,(S,1))=1+stab(G-[`N](A))
   a1b=alpha[-2];(= 0.691639).

   c. **(A,B) is an edge and d(A)=2**
   Let x be the other neighbour of A and G¢ be G-A-B with added edges between x and each remaining element of N(B).
   stab¢(G,(S,1))=1+$\mathrm{stab}^{¢¢¢}$(G¢,x).
   a1c=a7*alpha[-2];(= 0.804032).

   d. **{A,B} is an edge and d(A)=3**
   We assume that A and B have no common neighbour (case a1f).

      i. **d(B) > 3**
      stab¢(G,({A,B},1))=1+max(stab(G-[`N](A)),stab¢(G-[`N](B),(N(A)-B,1)))
      a1di=alpha[-4]+alpha[-5]*a1;(= 0.809219).

      ii. **d(B)=3** Of the four neighbours of A and B, let C be the one with minimum degree in G-[`N](B) or G-[`N](A). Assume wlog. that C is in fact in N(A) and that d(C) in G-[`N](B) is d; let the other two neighbours of B be D and E.

         A. d £ 3
         stab¢(G,({A,B},1)=1+max(stab(G-[`N](B)),stab¢(G-[`N](A),({D,E},1)))
         a1diiA=alpha[-4]*max(4,c1,c2,c3,c9)+alpha[-4]*a1;(= 0.809791).

         B. d = 4
         stab¢(G,({A,B},1))=max(1+stab(G-[`N](B)),2+stab(G-[`N]({A,D})),2+stab(G-[`N](A,E))
         a1diiB=alpha[-4]*c4+alpha[-4]*2*alpha[-5];(= 0.828620).

C. d > 4
We use the same calculation as in a1diiB. We no longer have the c4 from the low degree of C but this is compensated by the fact that |[`N]({A,D})| ³ 10 and similarly for E.
a1diiC=alpha[-4]+alpha[-4]*2*alpha[-6];(= 0.794903).

a1dii=max(3,a1diiA,a1diiB,a1diiC);(= 0.828620).

a1d=max(2,a1di,a1dii);(= 0.828620).

e. **{A,B} is an edge and d(A) > 3**
1+stab¢(G,(S,1))=max(stab(G-[`N](A)),stab(G-[`N](B))).
a1e=2*alpha[-5];(= 0.795661).
or if in fact d(A) > 4
a1ex=2*alpha[-6];(= 0.661710).

f. **A and B have a common neighbour v**
stab¢(G,(S,1))=stab ¢(G-v,(S,1)).
a1f=a1*alpha[-1];(= 0.691639).

g. **d(B)=2 (and so also d(A)=2)**
Let the neighbours of A be C and D. The independent sets to be considered contain either A or B, C and D.

i. **(C,D) is an edge**
stab¢(G,(S,1))=1+stab(G-[`N](A)).
a1gi=alpha[-3];(= 0.575200).

ii. **All neighbours of C and D apart from A are also neighbours of B**
Any independent set containing A can be transformed into one at least as large by removing A and any neighbours of B and adding {B,C,D}. stab¢(G,(S,1))=3+stab(G-[`N]({A,B})) a1gii=alpha[-6];(= 0.330855).

iii. **Otherwise**
stab¢(G,(S,1))=max(1+stab(G-[`N](A)),3+stab(G-[`N]({B,C,D}))) where, in the first call the graph has a vertex (B) of degree 2.
a1giii=c2*alpha[-3]+alpha[-7];(= 0.715475).

a1g=max(3,a1gi,a1gii,a1giii);(= 0.715475).

h. Otherwise
stab¢(G,(S,1))=1+max(stab(G-[`N](B)),stab¢(G-[`N](A)-B,(N(B),2))).

i. d(A),d(B)=2,3: a1hi=c2*alpha[-4]+a3*alpha[-4];(= 0.697047).

ii. d(A),d(B)=2,4: a1hii=c2*alpha[-5]+a4*alpha[-4];(= 0.680994).

iii. d(A),d(B)=2,5: a1hiii=c2*alpha[-6]+a5*alpha[-4];(= 0.665232).

iv. d(A),d(B)=2, > 5: a1hiv=c2*alpha[-7]+alpha[-4];(= 0.688998).

v. d(A),d(B)=3,3: a1hv=c3*alpha[-4]+a3*alpha[-5];(= 0.687115).

vi. d(A),d(B)=3,4: a1hvi=c3*alpha[-5]+a4*alpha[-5];(= 0.655681).

vii. d(A),d(B)=3, > 4: a1hvii=c3*alpha[-6]+alpha[-5];(= 0.682758).

viii. d(A),d(B)=4, 4: a1hviii=c4*alpha[-5]+a4*alpha[-6];(= 0.632951).

ix. d(A),d(B) ³ 4, > 4: a1hix=alpha[-6]+alpha[-6];(= 0.661710).

a1h=max(9,a1hi,a1hii,a1hiii,a1hiv,a1hv,a1hvi,a1hvii,a1hviii,a1hix);(= 0.697047).

a1=max(8,a1a,a1b,a1c,a1d,a1e,a1f,a1g,a1h);(= 0.831648).

If there are two disjoint ``1 out of 2'' conditions, either there is an isolated vertex v and one condition remains in G-[`N](v) or the m.i.s. must contain two vertices from the four in the two sets and case a_{4a} (an isolated vertex) cannot occur.

a1y=max(3,alpha[-1]*a1,a4b,a4c);(= 0.762660)

2. **One out of three**

a. **some element of S has degree 4 or less** stab¢(G,(S,1))=stab(G)
a2a=max(5,c9,c1b,c2,c3,c4);(= 0.936534).

b. **all elements of S have degree at least 5**

i. **Some elements s_1 and s_2 are not adjacent**
stab¢(G,(S,1))=1+max(stab(G-[`N]($s_1$)),stab(G-[`N](s_2)-s_1), stab(G-[`N](s_3)))
a2bi=2*alpha[-6]+alpha[-7];(= 0.936865).

ii. **All elements of S have degree 5**

A. **2 elements of S have a common neighbour not in S** Say s_2 and s_3 have a common neighbour u.
stab¢(G,(S,1))=max(1+stab(G-[`N](s_1)),stab¢(G-s_1-u,({s_2,s_3},1)))
a2biiA=alpha[-6]+alpha[-2]*a1;(= 0.906055)

B. **A neighbour of s_2 has degree < =5 in G-[`N](s_1)**
stab¢(G,(S,1))=1+max(stab(G-[`N]($s_1$)), stab¢(G-[`N](s_2),(N(s_1)-s_2-s_3,1)), stab¢(G-[`N](s_3),(N(s_1)-s_2-s_3,1),(N(s_2)-s_1-s_3,1)))
a2biiB=alpha[-6]*(c5+a2+a2x);(= 0.946261).

C. **Otherwise**
stab¢(G,(S,1))=1+max(stab(G-[`N]($s_2$)), stab¢(G-[`N](s_1),(N(s_2)-s_1-s_3,1)), stab¢(G-[`N](s_3),(N(s_1)-s_2-s_3,1),(N(s_2)-s_1-s_3,1)))
a2biiC=alpha[-6]*(1+a2z+a2x);(= 0.914713).

a2bii=max(3,a2biiA,a2biiB,a2biiC);(= 0.946261).

iii. **otherwise**
stab¢(G,(S,1))=1+max(stab(G-[`N]($s_1$)),stab(G-[`N](s_2)),stab(G-[`N](s_3)))
a2biii=2*alpha[-6]+alpha[-7];(= 0.936865).

a2b=max(3,a2bi,a2bii,a2biii);(= 0.946261).

Or, if there are two ``one out of three'' conditions with disjoint sets S and S¢, stab¢(G,(S,1),(S¢,1))=1+$\text{Max}_{i=1}^{3}$stab¢(G-[`N](s_i),(S¢,1))
a2bx=3*alpha[-6]*a2;(= 0.939226).

a2=max(2,a2a,a2b);(= 0.946261).
Or, if there are two ``one out of three'' conditions with disjoint sets S and S¢, a2x=max(2,a2a,a2bx);(= 0.939226).
Or, if S is not a triangle, a2y=max(2,a2a,a2bi);(= 0.936865).
Or, if all vertices of S have degree at least 6, a2z=3*alpha[-7];(= 0.825465).

3. **Two out of three**

   a. **There is a vertex A Î S of degree 0**
   stab¢(G,(S,2))=1+stab¢(G-A,(S-A,1))
   a3a=a1*alpha[-1];(= 0.691639).
   or if there are two ``two out of three'' conditions with disjoint sets S_1 and S_2 we choose an S_i without a vertex of degree 0 unless each has one (v_i)
   stab¢(G,(S_1,2),(S_2,2))=2+stab¢(G-v_1-v_2,(S_1-v_1,1),(S_2-v_2,1))
   a3ax=a1y*alpha[-2];(= 0.527485).

   b. **There is a vertex A Î V(G)-S of degree 1**
   stab¢(G,(S,2))=stab¢(G-[`N](A),(S-N(A),2))
   a3b=max(2,alpha[-2],a3)*alpha[-2];(= 0.478364).

   c. **S is a 3-clique**
   stab¢(G,(S,2))=0

   d. **S has two edges (A,B) and (A,C) (but excluding case g)**
   stab¢(G,(S,2))=2+stab(G-[`N](B)-[`N](C))
   a3d=alpha[-4];(= 0.478364).

   e. **S has one edge (A,B) (but excluding case g)**
   stab¢(G,(S,2))=1+stab¢(G-[`N](C),({A,B},1))
   a3e=max(4,a1b,a1c,a1d,a1e)*alpha[-3];(= 0.476623).

   f. **S has two vertices A and B with a common neighbour v**
   stab¢(G,(S,2))=stab¢(G-v,(S,2)).
   a3f=max(6,a3b,a3d,a3e,a3f,a3g,a3h)*alpha[-1];(= 0.478364).

   g. **S has a vertex A of degree 1**
   stab¢(G,(S,2))=1+stab¢(G-[`N](A),(S-A,1)).
   a3g=a1*alpha[-2];(= 0.575200).
   or a3gx=a3*alpha[-2];(= 0.478364).

   h. **None of the above**
   Let A be an element of S with minimum degree. stab¢(G,(S,2))=max(1+stab¢(G-[`N](A),(S-A,1)),2+stab¢(G-A-[`N](S-A),(N(A),2))), where the first call has S-A a set of two vertices each of degree at least d(A) and with no edge between them or a common neighbour.

      i. d(A)=2:
      (Better calculation of stab¢) stab¢(G,(S,2))=max(1+stab$^{¢¢}$(G-[`N](A),S-A),2+stab(G-[`N](S-A))).
      Many of the worst cases of stab$^{¢¢}$ can only occur when the size of [`N](S-A) is large:
      a3hi=max(14,/*a6d cannot occur because no common neighbours*/a6e*alpha[-3]+c2*alpha[-6],/*switching on C instead of A*/a6e*alpha[-4]+c3*alpha[-6],a6g*alpha[-3]+c2*alpha[-8],a6h*alpha[-3]+c2*alpha[-9],a6i*alpha[-3]+c2*alpha[-10],a6j*alpha[-3]+c2*alpha[-
      (= 0.523332).
      If we know only that A has no common neighbours with B and C the other elements of S (but not that B and C have no common neighbours) and that the minimum m of d(B), d(C) is greater than 2, we can do better. Either B and C have a common neighbour v giving stab¢(G,(S,2))=stab¢(G-v,(S,2)) where d(A)=2 in G-v and only cases a3h and a3f can occur or stab¢(G,(S,2))=max(1+stab$^{¢¢¢}$(G-[`N](A),{B,C},2+stab(G-[`N]({B,C}))) and, depending on the subcase of stab$^{¢¢¢}$ which occurs we have lower bounds on |[`N]({B,C})| and d(A)=2 in G-[`N]({B,C})
      If m=3,
      a3hix=max(2,max(2,a3h,a3f)*alpha[-1],max(3,a6k*alpha[-3]+c2*alpha[-8],a6l*alpha[-3]+c2*alpha[-9],a6m*alpha[-3]+c2*alpha[-10]));(= 0.491711),
      If m ³ 4,
      a3hiy=max(2,a3hix*alpha[-1],max(5,a6n*alpha[-3]+c2*alpha[-10],a6o*alpha[-3]+c2*alpha[-11],a6p*alpha[-3]+c2*alpha[-12],a6q*alpha[-3]+c
      (= 0.420685).

      ii. d(A)=3: We use the same method as in the no-common-neighbour part of cases a3hix and a3hiy:
      a3hii=max(3,c3*alpha[-8]+max(6,a6k,a6n,a6o,a6p,a6q,a6r)*alpha[-4],c3*alpha[-9]+a6l*alpha[-4],c3*alpha[-10]+a6m*alpha[-4]);(= 0.460316).

      iii. d(A) ³ 4:
      stab¢(G,({A,B,C},2))=max(1+stab¢(G-[`N](A),({B,C},1)),2+stab(G-A-[`N]({B,C})))
      a3hiii=a1h*alpha[-5]+alpha[-11];(= 0.408931).

   a3h=max(3,a3hi,a3hii,a3hiii);(= 0.523332).

   i. **Degrees of all vertices in S are at least 3**

      i. **Two elements A and B of S are neighbours**
      stab¢(G,({A,B,C},2))=1+stab¢(G-[`N](C),({A,B},1))
      a3ii=a1*alpha[-4];(= 0.397831).

ii. **Two elements of S have a common neighbour v**
stab¢(G,({A,B,C},2))=1+stab¢(G-v,({A,B,C},2))
a3iii=max(2,a3f,a3h)*alpha[-1];(= 0.435228).

Otherwise we must be in case 3h.
a3i=max(4,a3hii,a3hiii,a3ii,a3iii);(= 0.460316).

a3=max(8,a3a,a3b,a3d,a3e,a3f,a3g,a3h,a3i);(= 0.691639).
or if S does not contain any isolated vertex or there are two ``two out of three'' conditions with disjoint sets S
a3y=max(7,a3b,a3d,a3e,a3f,a3g,a3h,a3i);(= 0.575200).
a3z=max(8,a3ax,a3b,a3d,a3e,a3f,a3gx,a3h,a3i);(= 0.527485).

4. **Two out of four** We use the same evaluate function as in the previous section. Now the structures do not have the central node A but otherwise the computation is similar. Here, in the list of possible structures around S, we bound the number of common neighbours of s_i and s_j simply by min(d(s_i),d(s_j)) (minus one if (s_i,s_j) is an edge).

   a. **G has a vertex v of degree 0**
   stab¢(G,(S,2))=1+stab¢(G-v,(S-v,2-|SÇ{v}|))
   a4a=max(2,a2,a4)*alpha[-1];(= 0.786956).
   or if there are two ``two out of four'' conditions with disjoint sets S
   a4ax=a4*alpha[-1];(= 0.654471).

   b. **S has a vertex v of degree £ 2**
   stab¢(G,(S,2))=stab(G)
   a4b=max(5,c1b,c2b,a4bi,a4bii,a4biii);(= 0.702559).
   The worse cases of c2 are dealt with by considering the number w of elements of N(v)ÇS. The call stab(G-[`N]$^2$(v)) has the condition (S-v,2-w) and the call stab(G-[`N](v)) has a condition (S-N(v)ÇS,1) where this is stronger than the simple ([`N]$^2$(v)-[`N](v),2). If S-N(v)ÇS was a single vertex v¢ with $d_{G-[`N](v)}$(v¢)=0, we would have used instead the much stronger stab¢(G,(S,2))=2+stab(G-[`N]({v,v¢})) so any call stab¢(G-[`N](v),(S-N(v)ÇS,1)) where S-N(v)ÇS,1) = {v¢} is replaced by 1+stab¢(G-[`N]({v,v¢}),[`N]$^2$(v)-v¢,2-|[`N]^2(v)Ç{v¢}|) with the time bound for this call given by a^{-5} multiplied by the appropriate constant for the stab¢ call. This gives the following bounds where x as in c2 is |[`N]$^2$(v)-[`N](v)|, the maximum of three is over w=0,1,2 in that order and the first summand is always that for G-[`N](v).

      i. x=3
      a4bi=max(3,a3y*alpha[-3]+a3*alpha[-6],a3y*alpha[-3]+a1*alpha[-6],max(2,a1,a3)*alpha[-5]+alpha[-6]);(= 0.661710).

      ii. x=4
      a4bii=max(3,a4*alpha[-3]+a3*alpha[-7],a4*alpha[-3]+a1*alpha[-7],max(2,a2,a4)*alpha[-5]+alpha[-7]);(= 0.681490).

      iii. x ³ 5
      a4biii=max(3,a2*alpha[-3]+a3*alpha[-8],a1*alpha[-3]+a1*alpha[-8],alpha[-5]+alpha[-8]);(= 0.702559).
      If there are two ``two out of 4'' conditions with disjoint sets S, the second is still valid in the call stab(G-[`N](v)) giving
      a4biiix=max(3,a4*alpha[-3]+a3*alpha[-8],a1*alpha[-3]+a1*alpha[-8],alpha[-5]+alpha[-8]);(= 0.668672) and
      a4bx=max(5,c1b,c2b,a4bi,a4bii,a4biii);(= 0.702559).

   c. **All vertices of S have degree ³ 3**
   a4c=evaluate(twofrom4);(= 0.762660).

   a4=max(3,a4a,a4b,a4c);(= 0.786956).
   or if there are two ``two out of four'' conditions with disjoint sets S
   a4x=max(3,a4ax,a4bx,a4c);(= 0.762660)
   or similarly if there are a ``two out of four'' and a ``two out of five'' condition with disjoint sets S
   a4y=max(3,a5*alpha[-1],a4b,a4c);(= 0.762660)

5. **Two out of five**

   a. **G has a vertex of degree £ 3**
   stab¢(G,(S,2))=stab(G,S)
   a5a=max(4,c9a,c1b,c2,c3);(= 0.861185).

   b. **d(s_1) > 7**
   stab¢(G,(S,2))=max(1+stab¢(G-[`N](s_1),(S-s_1,1)),stab¢(G-s_1,(S-s_1,2))) a5b=alpha[-9]+a4c*alpha[-1];(= 0.824573).

   c. **All degrees ³ 4 but £ 6**
   a5c=evaluate(twofrom5);(= 0.851424).

   a5=max(3,a5a,a5b,a5c);(= 0.861185).

6. **Two out of six** Setting this to 1 doesn't change result so we can eliminate it unless there is scope for improving result by reducing it substantially. The best that can possibly be hoped for would be a5 so here goes! That didn't make any difference either, so this section goes requiring renumbering of a6, a7, including any a6 and a7 in graph.dec etc.

7. **Exactly one out of two**
We need consider only independent sets containing x and two neighbours of y or vice versa. We switch on the degrees d(x) and d(y), assuming w.l.o.g. that d(x) £ d(y). Except where otherwise noted we use the simple calculation stab$^{¢¢¢}$(G,{x,y})=1+max(stab¢(G-[`N](x)-y,(N(y),2)),stab¢(G-[`N](y)-x,(N(x),2)).

   a. **both degrees < 2**
   stab$^{¢¢}$(G,x,y)=0.

   b. **d(x)=0**
   stab$^{¢¢}$(G,x,y)=1+stab(G-[`N](x)-y)
   a6b=alpha[-2];(= 0.691639).

   c. **d(x)=1**
   stab$^{¢¢}$(G,x,y)=1+stab(G-[`N](x)-y)
   a6c=alpha[-3];(= 0.575200).

d. **x and y have a common neighbour v**
stab$^{¢¢}$(G,x,y)=stab$^{¢¢}$(G-v,x,y)
a6d=max(16,a6c,a6d,a6e,a6f,a6g,a6h,a6i,a6j,a6k,a6l,a6m,a6n,a6o,a6p,a6q,a6r)*alpha[-1];(= 0.507296).

e. **d(x)=2, d(y)=2**
W.l.o.g. assume that |[`N]$^2$(x)-[`N]({x,y})| = b £ |[`N]$^2$(y)-[`N]({x,y})|.

i. b=0
y+[`N](x) dominates x+[`N](y).
stab$^{¢¢}$(G,x,y)=3+stab(G-[`N]({x,y})).
a6ei=alpha[-6];(= 0.330855).

ii. b=1
The independent sets consist of 3 elements of [`N]({x,y}) together with an independent set of G-[`N]({x,y}) which may contain [`N]$^2$(x)-[`N]({x,y}) or part of [`N]$^2$(y)-[`N]({x,y}) but not both. Let z be the single element of [`N]$^2$(x)-[`N]({x,y}) and G¢ be the graph obtained by adding to G-[`N]({x,y}) edges between z and every element of [`N]^2(y)-[`N]({x,y}).
stab$^{¢¢}$(G,x,y)=3+stab$^{¢¢¢}$(G¢,z),
a6eii=a7*alpha[-6];(= 0.384620).

iii. b ³ 2
stab$^{¢¢}$(G,x,y)=3+max(stab(G-[`N](x)-[`N]^2(y)),
stab(G-[`N](y)-[`N]^2(x)))
a6eiii=2*alpha[-8];(= 0.457665).

a6e=max(3,a6ei,a6eii,a6eiii);(= 0.457665)

f. **d(x)=2, d(y)=3**
a6f=a3*alpha[-4]+alpha[-7];(= 0.606010).

g. **d(x)=2, d(y)=4**
a6g=a4*alpha[-4]+alpha[-8];(= 0.605284).

h. **d(x)=2, d(y) = 5**

i. **[`N]$^2$(x)-[`N](x) Ì N(y)**
An independent set containing x and two neighbours of y is no better than one containing y and the two neighbours of x so we consider only those containing three neighbours of y and therefore two out of an arbitrary four of these neighbours.
a6hi=a4*alpha[-4]+alpha[-9];(= 0.566760).

ii. **Otherwise** a6hii=a5*alpha[-4]+alpha[-10];(= 0.570229).

a6h=max(2,a6hi,a6hii);(= 0.570229).

i. **d(x)=2, d(y) = 6**
As in the previous case
a6i=max(2,a5*alpha[-4]+alpha[-10],alpha[-4]+alpha[-11]);(= 0.609989).

j. **d(x)=2, d(y) > 6**
a6j=alpha[-4]+alpha[-11];(= 0.609989).

k. **d(x)=3, d(y)=3**: We consider the neighbours of x, say a, b, c and those of y, say d, e, f and their degrees in the induced subgraphs G-[`N](y)-x and G-[`N](x)-y respectively.

i. **Two of a, b, c or two of d, e, f have degree 0 in the subgraphs**
Without loss of generality suppose that the vertices concerned are a, b. We need consider only independent sets containing either x, d, e,f or y and two from a, b, c.
stab$^{¢¢}$(G,{x,y})=max(4+stab(G-[`N](x)-[`N]^2(y)),1+stab¢(G-[`N](y)-x,(N(x),2))).
a6ki=alpha[-8]+a3*alpha[-5];(= 0.503987).

ii. **else**
a6kii=2*a3*alpha[-5];(= 0.550310).

a6k=max(2,a6ki,a6kii);(= 0.550310).

l. **d(x)=3, d(y)=4**
a6l=a4*alpha[-5]+a3*alpha[-6];(= 0.541908).

m. **d(x)=3, d(y) ³ 5**
a6m=alpha[-5]+a3*alpha[-7];(= 0.588139).

n. **d(x)=4, d(y)=4**
a6n=2*a4*alpha[-6];(= 0.520737).

o. **d(x)=4, d(y)=5**
a6o=a5*alpha[-6]+a4*alpha[-7];(= 0.501462).

p. **d(x)=4, d(y) > 5**
a6p=alpha[-6]+a4*alpha[-8];(= 0.510936).

q. **d(x)=5, d(y)=5**
a6q=2*a5*alpha[-7];(= 0.473919).

r. **d(x) > 4, d(y) > 5**
a6r=alpha[-7]+alpha[-8];(= 0.503987).

a6=max(17,a6b,a6c,a6d,a6e,a6f,a6g,a6h,a6i,a6j,a6k,a6l,a6m,a6n,a6o,a6p,a6q,a6r);
(= 0.691639).

8. **A vertex with added edges**
   There is a m.i.s. containing either x or two neighbours of x.

   a. **d(x) £ 1**
      stab$^{¢¢¢}$(G,x)=1+stab(G-[`N](x))
      a7a=alpha[-1];(= 0.831648).

   b. **d(x)=2**
      stab$^{¢¢¢}$(G,x)=max(1+stab(G-[`N](x)),2+stab(G-[`N]^2(x))
      a7b=2*alpha[-3];(= 1.150400).

   c. **d(x)=3**
      stab$^{¢¢¢}$(G,x)=max(1+stab(G-[`N](x)),stab¢(G-x,(N(x),2))
      a7c=alpha[-4]+a3*alpha[-1];(= 1.053564).

   d. **d(x)=4**
      stab$^{¢¢¢}$(G,x)=max(1+stab(G-[`N](x)),stab¢(G-x,(N(x),2)))
      a7d=alpha[-5]+a4*alpha[-1];(= 1.052301).

   e. **d(x) > 4**
      stab$^{¢¢¢}$(G,x)=max(1+stab(G-[`N](x)),stab(G-x))
      a7e=alpha[-6]+alpha[-1];(= 1.162503).

   a7=max(5,a7a,a7b,a7c,a7d,a7e);(= 1.162503).

4 Memorisation

A useful gain in time is obtained at the cost of using exponential space by remembering the result found for any induced subgraph already analysed and avoiding doing the same calculation several times. This gain is evaluated by observing that the number of calls of the algorithm on graphs of size greater than m is O($2^{c(n-m)}$) so that if m is chosen so that the number of induced subgraphs of size £ m is also O($2^{c(n-m)}$) then so is the overall time.

4.1 Connected induced subgraphs

In fact it suffices to consider connected induced subgraphs and to consider graphs with degree bounded by 9 since the performance bounds we will obtain are easily proved recursively for all graphs if they hold for degree £ 9 graphs. A bound on the number of small connected induced subgraphs is then obtained by considering the number of 8-ary trees of size m (where a 8-ary tree is either empty or has a root node and 8 subtrees; note that the order of the subtrees is important). Relative to a given root node and an ordering of the edges at each vertex, the tree is interpreted as a spanning tree of the subgraph.

This bound overestimates the number of subgraphs for three reasons:

- Where the tree indicates that an edge is to be followed, the graph may have no edge; this is not significant since we want an upper bound on the number of connected subgraphs and adding more edges can only increase the number.
- An edge may lead to a vertex which has already been visited.
- A connected subgraph may have many spanning trees.

For the second of these three phenomena we have only a probabilistic estimate of its effect on the mean performance of the algorithm. For the third we can show a definite improvement in the worst case. We rely on the fact that the time taken by the algorithm is bounded by a polynomial times the number of graphs considered where two or more recursive calls are made. Thus, we need only count subgraphs with minimum degree at least 2 and no occurrences of dominance of the types mentioned in Section 1. In the following section we will assume that subgraphs are of this type.

4.2 Weighted Spanning Trees

Suppose a spanning tree of a subgraph has l leaves. Since each of these vertices has degree at least 2 *qua* subgraph vertex, we can obtain other spanning trees (also with at most l leaves) by attaching any leaf v to another neighbour and breaking the resulting cycle just below the first ancestor of v (in the original spanning tree) which is either of degree greater than 2 in the spanning tree or is its root. If we start with a spanning tree which minimises the number of leaves, no two leaves can be adjacent in the subgraph and this process of obtaining new spanning trees can be done without interference for any subset of the leaves and results in a different spanning tree in each case.

Hence if each 8-ary tree is given a weight equal to 2^{-l} where l is its number of leaves, the sum of the weights for all spanning trees of a subgraph with minimum number m of leaves over all its spanning trees, is at least 2^m 2^{-m} i.e. at least 1.

The number of 8-ary trees with size m and m_i nodes of outdegree i (0 £ i £ 8) is (

m

$m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7,m_8$

)$Õ_{i=0}^{8}$(

8

i

)^{m_i}/m provided the m_i satisfy $å_{i=0}^{8}$ m_i=m and $å_{i=0}^{8}$ im_i=m-1. If we fix the proportion of tree nodes with out-degree i to x_i+O(1/m) and let m tend to infinity, the sum of the weights of trees of size m is to within a polynomial factor ($2^{-x_0}Õ_{i=0}^{8}$((

8

i

)/x_i)^{x_i})^m. A simple calculation shows that subject to the constraints $å_{i=0}^{8}$ x_i=1 and $å_{i=0}^{8}$ ix_i=1 the constant is maximum at about 16.403 when the x_i are about
(0.282189, 0.487715, 0.184391, 0.039836, 0.005379, 0.000465, 0.000025, 0.000001, 0.000000). Hence summed over all possible m_i the sum is bounded by 16.403^m ×poly(m). This can be compared with the unweighted sum (the number of trees) which is about 20.371998^m.

This raises the level at which the memorisation becomes effective (number of subgraphs possible is less than the number of recursive calls generated) to about 0.0618n thereby reducing the constant c to just under 0.25.

For graphs of degree greater than 9, we simply choose any vertex A of degree greater than 9 and call stab(G)=max(stab(G-A),1+stab(G-[`N](A))). Since this gives recursive calls on graphs of sizes n-1 and £ n-11 and $a^{¢-1}+a^{¢-11} < 1$, this gives run time of $O(a^{¢n})$ for all graphs.

References

[1]
Chen, J. Kanj, I. A. and Jia, W., Vertex cover: further observation and further improvement, *25th International workshop on graph-theoretic concepts in computer science*, Springer LNCS.

[2]
Jian, T., An $O(2^{0.304n})$ algorithm for solving maximum independent set problem, *IEEE Trans. Comput.* **C-35** (1986), 847-851.

[3]
Robson, J. M., Algorithms for maximum independent sets, *J. Algorithms* **7** (1986), 425-440.

[4]
Tarjan, R. E. and Trojanowski, A. E., Finding a maximum independent set, *SIAM J. Comput.* **6** (1977), 537-546.
