Finding Maximum Independent Sets in Sparse and General Graphs

Richard Beigel*
University of Illinois at Chicago

Closely related to the maximum clique problem, the maximum independent set problem (MIS) was among the first problems proved NP-complete [5]. It is now known that the size of a maximum independent set cannot be approximated even within a factor of $n^{1-o(1)}$ in polynomial time [2, 1, 6, 3]. Therefore it is important to find faster exact algorithms for MIS.

Tarjan and Trojanowski [9] were the first to break through the 2^n barrier: they gave a $2^{0.333n}$-time, polynomial-space algorithm for MIS. Tarjan and Trojanowski [9] observed that if G has a node of degree 0 or 1, then it suffices to solve MIS on an induced subgraph of G. If G has a higher-degree vertex, then they solve MIS on several induced subgraphs of G.

The result of Tarjan and Trojanowski [9] was improved by Jian [4] and independently by Robson [7]. Jian gave a $2^{0.304n}$-time, polynomial-space algorithm. Robson gave a $2^{0.292n}$-time, polynomial-space algorithm, as well as a $2^{0.276n}$-time, exponential-space algorithm. All of those algorithms involve tedious case enumeration. Shindo and Tomita [8] gave a $2^{0.349n}$-time, polynomial-space algorithm that is easy to describe but nontrivial to analyze.

Our algorithm produces subproblems that need not be induced subgraphs of G. One immediate advantage is that if G has a node of degree 2 we solve only one subproblem, whereas previous algorithms solved two. On the other hand, the resulting recursion may produce sub*problems containing nodes of arbitrarily high degree, even though the degree of G is bounded. By counting edges instead of vertices as our progress measure, we are nonetheless able to obtain good results for bounded-degree graphs.

We obtain the fastest MIS algorithms for sparse graphs and for graphs whose degree is bounded by 3 or by 4. We also obtain the fastest MIS algorithm for general graphs that runs in polynomial space, and the fastest MIS algorithm for general graphs that fits on one journal page.

Our first simple algorithm finds maximum independent sets in time

- $2^{0.114e}$ for graphs with e edges
- $2^{0.171n}$ for n-node graphs with maximum degree 3
- $2^{0.228n}$ for n-node graphs with maximum degree 4

Our second simple algorithm finds maximum independent sets in time $2^{0.303n}$ for general graphs. Our third algorithm, which uses a complicated case analysis, finds maximum independent sets in time $2^{0.290n}$ for general graphs. All of our algorithms run in polynomial space.

Space allows us to present only our first two algorithms.

References

[1] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Inform. and Comput.*, 96:77–94, 1992.

[2] U. Feige, S. Goldwasser, L. Lovàsz, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32th Ann. IEEE Symp. Found. Computer Sci.*, pages 2–12, 1991.

[3] J. Hastad. Clique is hard to approximate within $n^{1-o(1)}$. In *FOCS96*, 1996.

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

[5] R. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations.* Plenum Press, 1972.

[6] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. Technical memorandum, AT&T Bell Laboratories, 1992.

[7] J. Robson. Algorithms for maximum independent sets. *J. Algorithms*, 7:425–440, 1986.

[8] M. Shindo and E. Tomita. A simple algorithm for finding a maximum clique and its worst-case time complexity. *Systems and Comput. in Japan*, 21:1–13, 1990.

[9] R. Tarjan and A. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6:537–546, 1977.

*Research performed in part at Yale, U. Maryland, and Lehigh University. Address: Dept. of Electrical Engineering and Computer Science, University of Illinois at Chicago, Rm. 1116, SEO Building, M/C 154, 851 S. Morgan Street, Chicago, IL 60607, USA. Supported in part by the National Science Foundation under grants CCR-9700417 and CCR-9796317. Email: `beigel@eecs.uic.edu`. Web: `http://www.eecs.uic.edu/~beigel/`

```
function SparseFindMIS(G)
begin
if G contains an edge {u, v} such that N(u) ⊂ N(v) ∪ {v} then
      return SparseFindMIS(G − {v})
for each vertex v in G (in increasing order of degree) do
      if N(v) contains no anti-triangle then
            for each anti-edge {u, w} in N(v) do
                  form a new vertex uw with N(uw) = N(u) ∪ N(w)
            place edges between each pair of new vertices
            G' := G − {v} − N(v) ∪ {uw : {u, w} is an anti-edge in N(v)}
            if G' contains no more edges than G then
                  /* G' must contain fewer vertices than G */
                  I := SparseFindMIS(G')
                  if I contains a vertex of the form uw then
                        return I − {uw} ∪ {u} ∪ {w}
                  else
                        return I ∪ {v}
let v be a maximum-degree vertex in G
      (with a degree-3 neighbor if possible, otherwise minimizing the number of edges in N(v))
return max(SparseFindMIS(G − {v}), {v} ∪ SparseFindMIS(G − {v} − N(v)))
end
```

Figure 1: An algorithm that runs in time $2^{0.114e}$ on e-edge graphs

```
function FindMIS(G)
begin
if G contains at most 5n/2 edges then
      return SparseFindMIS(G)
if G contains an edge {u, v} such that N(u) ⊂ N(v) ∪ {v} then
      return FindMIS(G − {v})
let d be the maximum degree of any vertex in G (* so d ≥ 6 *)
if d ≥ 8 then
      let v be a degree-d vertex
      return max(FindMIS(G − {v}), {v} ∪ FindMIS(G − {v} − N(v)))
choose v such that (∀w ∈ N(v))[deg(v) ≤ deg(w)] and (deg(v) ≥ 6 or (∃w ∈ N(v))[deg(v) < deg(w)])
let C be a maximum clique in N(v)
      (choose the lowest degree neighbors if there is more than one choice)
let D be a maximum independent set in N(v) − C
      (choose the highest degree neighbors if there is more than one choice)
I := ∅
for all but one vertex w ∈ N(v) − C (starting with elements of D and proceeding in decreasing order of degree)
do
      I := max(I, {w} ∪ FindMIS(G − {w} − N(w)))
      G := G − {w}
let z be the unique remaining element of N(v) − C
add all edges in C × N(z) to G
J := FindMIS(G − {v} − {z})
if N(v) ∩ J = ∅ then J := J ∪ {v}
else J := J ∪ {z}
return max(I, J)
end
```

Figure 2: A simple algorithm that runs in time $2^{0.303n}$ on n-node graphs