Algorithms – CS-27200/CS-37000
Branch-and-bound: improved exponential time bounds
Maximum independent sets in graphs

Instructor: László Babai

In this note, *graph* means *undirected graph* without loops. An *independent set* in a graph $G = (V, E)$ is a set $S \subseteq V$ such that no pair of vertices in S is adjacent (there are no edges within S). In other words, in the complement of G, the subset S is a clique.

Let $\alpha(G)$ denote the maximum size of independent sets in G. The problem is to determine $\alpha(G)$. This problem is "NP-hard," so we don't expect it to be solvable in polynomial time. Our goal is to improve over the brute force method which would perform an exhaustive search of the search space.

The search space consists of all subsets of V. If $|V| = n$ then the size of the search space is 2^n.

We can search the search space by organizing it as a binary tree; at each node, we make a decision whether or not to include a particular vertex into S. This tree has depth n and it has 2^n leaves.

While tracing this tree, we can cut off entire branches when we recognize that nothing in that branch can lead to optimum; or we recognize that within that branch, we have an easier way to find the optimum.

Our goal is to show that simple "branch-and-bound" ideas can be analysed and lead to considerably better bounds than 2^n (although the bounds will still be exponential).

For a vertex $v \in V$, let $N(v)$ denote the set of neighbors of v plus v itself. Let $G \setminus v$ denote the graph G with v deleted; and $G \setminus N(v)$ the graph G with the entire neighborhood of v (including v itself) deleted. Note that $|N(v)| = \deg(v) + 1$.

The key observation is the following, dynamic-programming-style equation.

Observation. If $n \geq 2$ then for any $v \in V$,

$$\alpha(G) = \max\{\alpha(G \setminus v), 1 + \alpha(G \setminus N(v))\}. \tag{1}$$

(The first value corresponds to the decision $v \notin S$, the second to $v \in S$.)

This equation corresponds to an evident recursive algorithm. (DO: Write the algorithm in pseudocode. It should be a few lines only.)

The algorithm reduces an instance with n vertices to two instances, one with $n - 1$, and the other with $n - \deg(v) - 1$ vertices. The cost of the reduction is $O(n^2)$. This gives us the recurrence

$$T(n) \leq T(n-1) + T(n - \deg(v) - 1) + O(n^2). \tag{2}$$

It is plausible then, that we should always choose v to have maximum degree (so the right-hand side of inequality (2) will be minimized). (Modify your pseudocode to reflect this choice!) Now inequality (2) will read

$$T(n) \leq T(n-1) + T(n - \deg_{\max} -1)) + O(n^2), \tag{3}$$

where $\deg_{\max}$ is the maximum degree.

Now, if $\deg_{\max} = 0$ then the graph has no edges, and we declare $\alpha(G) = n$ without any further recursive calls. (DO: Modify the pseudocode to reflect

this change!) Notice that at this point, we eliminated an entire branch of the tree (we "bounded the search.") Let us see what this evident step buys us. We notice that inequality (3) now implies

$$T(n) \le T(n-1) + T(n-2) + O(n^2), \tag{4}$$

since in the case $\deg_{\max} = 0$, no work is needed (except to establish this fact, which takes only $O(n)$ steps); in all other cases, $T(n - \deg_{\max}) \le T(n-2)$.

We have shown (see the "Evaluation of recurrent inequalities" handout) that the solution to this recurrence is $T(n) = O(\phi^n)$ where $\phi = (1+\sqrt{5})/2 \approx 1.618$ is the golden ratio.

With a simple additional trick we can further improve on this bound.

Theorem. $\alpha(G)$ can be computed in $O(\psi^n)$ steps, where $\psi > 1$ satisfies the equation $\psi^4 = \psi^3 + 1$ (so $\psi \approx 1.381$).

The trick is based on the following observation:

Observation. If $\deg_{\max} \le 2$ then each connected component of G is a path or a cycle. (Paths of length zero are permitted, they are isolated vertices.) (Prove!)

It is straightforward to find $\alpha(G)$ for such a graph in $O(n)$ time. (How?)

We modify the algorithm so that we use the recurrence (1) only when $\deg_{\max} \ge 3$.

```
1    if  max_deg ≤ 2 then
2            find α(G) in time O(|V|)
3        else
4            find v ∈ V such that deg(v) = deg_max
5            recursively compute α(G \ v) and α(G \ N(v))
6            α(G) := max{α(G \ v), 1 + α(G \ N(v))}
7    end(if)
8    return  α(G)
```

This algorithms leads to the recurrence

$$T(n) \le T(n-1) + T(n-4) + O(n^2), \tag{5}$$

because lines 5–6 become active only when $\deg_{\max} \ge 3$.

The analysis of inequality (5) is analogous to the analysis of the Fibonacci recurrence (4) discussed in the "Evaluation of recurrent inequalities" handout. Ignoring first the $O(n^2)$ term, we look for a solution to the reverse inequality in the form of $g(n) = \psi^n$. This means $\psi^n \ge \psi^{n-1} + \psi^{n-4}$; dividing each side by ψ^{n-4} we obtain $\psi^4 \ge \psi^3 + 1$. We want ψ to be as small as possible subject to this condition; this means $\psi^4 = \psi^3 + 1$. Only one positive number ψ satisfies this equation; the solution is $\psi \approx 1.341$. As usual, we take the $O(n^2)$ term into account by looking for a solution of the reverse inequality $g(n) \ge g(n-1) + g(n-4) + Cn^2$ in the form $g(n) = A\psi^n - Bn^2$, with appropriate constants A, B. (DO: work out the details!)