Algorithms – CMSC-272

Branch-and-bound: improved exponential time bounds
The "Maximum Independent Set" problem 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 (all NP problems are Cook-reducible to it), so it cannot be solved in polynomial time unless P = NP. As a consequence, we don't expect it to be solvable in polynomial time. Our goal is to improve over the "brute force" search 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. The cost of an exhaustive search would be $T(n) = \Theta(2^n)$. We wish to significantly reduce this cost.

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, and $\widetilde{N}(v) = N(v) \cup \{v\}$. So $|\widetilde{N}(v)| = \deg(v) + 1$. Let $G \setminus v$ denote the graph G with v deleted; and $G \setminus \widetilde{N}(v)$ the graph G with v and its neighbors deleted.

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

Observation. If $n \geq 1$ then for any $v \in V$,

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

(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 $O(n^2)$. Denoting by $T(G)$ the cost of computing $\alpha(G)$, we obtain the recurrence

$$T(G) \leq T(G \setminus v) + T(G \setminus \widetilde{N}(v)) + O(n^2). \tag{2}$$

It is plausible then, that we should always choose v to have maximum degree (so the number of vertices in the graph $G \setminus \widetilde{N}(v)$ on the right-hand side will be minimized). (DO: Modify your pseudocode to reflect this choice!)

Let now $T(n)$ denote the maximum cost of computing $\alpha(G)$ for graphs with n vertices and let $T(n, \Delta)$ denote the maximum cost of computing $\alpha(G)$ for graphs with n vertices and maximum degree $\deg_{\max}(G) \geq \Delta$. From inequality (2) we now infer

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

Note that $T(n) = T(n, 0)$. But if we set $\Delta = 0$ in Eq. (3) then we get the recurrence $T(n) \leq 2 \cdot T(n-1)$, which resolves to $T(n) = O(2^n)$. So and we did not gain over brute force enumeration.

But $\Delta = 0$ is only required when $\deg_{\max}(G) = 0$, so the graph has no edges. So let us modify the algorithm to detect if $\deg_{\max}(G) = 0$, and if so, then instead of using the recurrence (2), simply observe that $\alpha(G) = n$ and exit. (DO: Modify the pseudocode to reflect this change!)

Notice that at this point, we have eliminated entire branches of the tree (we "bounded the search.") Let us see what this straightforward modification buys us. We notice that inequality (3) now implies

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

since in the case $\deg_{\max}(G) = 0$, we are done in $O(n)$ steps by detecting that $\deg_{\max}(G) = 0$. In all other cases, $T(n - \deg_{\max}(G) - 1) \leq T(n-2)$. We refer to Eq. (4) as the **"perturbed Fibonacci recurrence"** (the term $O(n^2)$ representing the perturbation).

DO: Use the method of reverse inequalities (see the "Evaluation of recurrent inequalities" handout) to show that the solution to the perturbed Fibonacci recurrence is $T(n) = O(\phi^n)$ where $\phi = (1+\sqrt{5})/2 \approx 1.618$ is the golden ratio. (Look for a solution to the reverse inequality $g(n) \geq g(n-1)+g(n-2)+Cn^2$ in the form $g(n) = A\phi^n - Bn^2$.)

While this trick already significantly reduced the complexity, a simple additional observation will help further reduce the 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 modified algorithm will be based on the following observation.

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

DO: Given a graph G, decide whether $\deg_{\max}(G) \leq 2$ and if so, find $\alpha(G)$. Your algorithm should run in $O(n)$ time.

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

Procedure `find-alpha`
Input: a graph $G = (V, E)$
Output: $\alpha(G)$

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

This algorithm leads to the recurrence

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

because lines 5–6 become active only when $\deg_{\max} \geq 3$ (so we can set $\Delta = 3$ in Eq. (3)) and lines 1–2 only require $O(n)$ steps.

The analysis of inequality (5) is analogous to the analysis of the perturbed Fibonacci recurrence (4). First we ignore the $O(n^2)$ term and look for a solution to the reverse inequality in the form of $g(n) = \psi^n$. This means $\psi^n \geq \psi^{n-1} + \psi^{n-4}$; dividing each side by ψ^{n-4} we obtain $\psi^4 \geq \psi^3 + 1$. 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

$$g(n) \geq g(n-1) + g(n-4) + Cn^2 \tag{6}$$

in the form $g(n) = A\psi^n - Bn^2$, with appropriate constants A, B. (DO: work out the details!)