Algorithmic Modeling and Complexity **Fall 2002**

Lecture 7: 23 September

Lecturer: J. van Leeuwen *Scribes: Dirk Smit and Peter Scheer*

7.1 Overview

In this lecture we will see two approaches to solving NP-complete and other hard problems: *heuristics*, and *exhaustive search*. Heuristics are (usually) simple algorithms that give answers that hopefully are close to the optimum solution. Exhaustive search methods are designed to solve problems exactly. To illustrate the methods, we use the Maximum Independent Set problem as an example.

7.2 Maximum Independent Set

The Maximum Independent Set problem often arises in the following form: *given n tasks which compete for the same resources, compute the largest set of tasks which can be performed at the same time without sharing resources.* More generally, in case the tasks have weights (priorities), one may ask for the set of tasks of largest total weight which can be performed at the same time. This is called the *Maximum Weighted Independent Set* problem.

The network model for this problem is given by a graph $G = (V, E)$ where V is the set of tasks and an edge $e \in E$ is drawn between two tasks when they need resources that cannot be used simultaneously, i.e. when the two tasks are dependent. An *independent set* in G is any subset $I \subseteq V$ such that no two vertices in I are joined by an edge in E.

Definition 7.1 (Maximum Independent Set) *Given a graph $G = (V, E)$, determine an independent $I \subseteq V$ of largest possible size.*

One may also consider the complementary graph $\overline{G}$ of G in which an edge is drawn between two tasks when there is *no* edge between them in G, i.e. when the two tasks *can* be scheduled simultaneously. A *clique* in G is any subset $C \subseteq V$ such that every two vertices in C are joined by an edge in E. The Maximum Independent Set problem in G is equivalent to the Maximum Clique problem in $\overline{G}$.

Definition 7.2 (Maximum Clique) *Given a graph $G = (V, E)$, determine a clique $C \subseteq V$ of largest possible size.*

In this lecture, we will often make use of maximAL independent sets instead of only maximUM independent sets. An independent set $I \subseteq V$ is called *maximal* if any enlargement of the set

makes it dependent, i.e. if $I \cup \{v\}$ is no longer independent for any $v \notin I$. So the maximUM independent set is the largest maximAL independent set. It is important to note that the size of a maximal independent set can be rather far off the size of a maximum independent set. For example, consider the graph G consisting of one node v and $n-1$ nodes $w_1, \ldots, w_{n-1}$, with v connected by an edge to each of the w_i and no further edges. Clearly $\{v\}$ is a maximal independent set in G, but $\{w_1, \ldots, w_{n-1}\}$ is the maximum independent set and it has size $n-1$.

The decision version of the Maximum Independent Set problem is the following problem, denoted by IS: *given a graph G and a positive integer k, decide whether G contains a maximum independent set of size $\geq k$.* Of course it is equivalent to ask whether G contains any independent set of size $\geq k$.

Lemma 7.3 *The decision version of the Maximum Independent Set problem is NP-complete.*

Proof: (i) Clearly $IS \in NP$. The only thing left to show is that $SAT \preccurlyeq_p IS$, because we know that SAT is NP-complete.

Suppose we have an instance $C = C_1, \ldots, C_k$ of SAT with k clauses. Make a graph G as follows. Construct for each clause C_i a component, which is a complete graph: each literal in the clause is a vertex in that component and all literals in it are connected by an edge. Now connect two nodes in different components iff the corresponding literals are each other's negation. (The two nodes can be seen as competing for the value 'True'.) The construction can be done in polynomial time. We show that this is a p-reduction.

Indeed $C_1, \ldots, C_k$ is satisfiable if and only if G has an independent set of size $\geq k$. For, consider any satisfying truth-value assignment for C and how it translates to G. If one gives a literal x value 'True' in one component, the literals $\overline{x}$ in the other components connected from x will not be given value 'True'. But in every component at least one literal must become true. Let I be any set consisting of one vertex from each component that corresponds to a literal that gets value 'True'. I is an independent set and $|I| = k$. Conversely, let I be any independent set of size $\geq k$ in G. I must contain precisely one node from each component (and thus has size precisely k). Assigning the corresponding literals the value 'True' is a valid truth-value assignment (no two of the chosen literals conflict) that satisfies each clause of C. ■

Interestingly, there is a direct connection between the construction question for the Maximum Independent Set problem and the P-versus-NP problem.

Lemma 7.4 *There is a polynomial time algorithm for the Maximum Independent Set problem if and only if P = NP.*

Proof:

(Not presented in class.) Suppose A is a polynomial time algorithm for the Maximum Independent Set problem. A can be used to solve the decision version of the problem, in polynomial time. Because we have just shown the decision problem to be NP-complete, it follows that P = NP.

Conversely, suppose P = NP. This means that there is a polynomial time algorithm A for the decision version of Maximum Independent Set. Given any graph G, one can use A in a binary search procedure that looks for the largest k for which there exists an independent set of size $\geq k$ in G: this k is the size of the maximum independent set. Because the maximum independent set has size $\leq n$, the binary search needs only $O(\log n)$ calls of A and the whole procedure takes polynomial time. Let the resulting algorithm that computes the size of the maximum independent set in a graph be called B.

So far we only solved the 'evaluation problem'. To solve the *construction problem* for Maximum Independent Set we proceed as follows. Let G be any given graph. Run B to determine the size k of the maximum independent set in G. We now build an independent set I from a set of candidate nodes J in stages as follows, while maintaining as an invariant that

(a) the nodes in J are not connected to any nodes in I, and
(b) I can be extended to an independent set of size k by means of nodes from J.

Begin stage 1 with $I = \emptyset$ and $J = V$. Suppose we have progressed to stage i, with sets I and J that satisfy the invariant. If $J = \emptyset$, then we can stop: I must be a maximum independent set. If $J \neq \emptyset$, pick any $v \in J$. Let G' be the graph G with all nodes of $I \cup \{v\}$ *and their neighbours* (and all incident edges) *deleted*. Run B on G' and let the answer be t.

If $|I| + 1 + t < k$, then the extension of I by v cannot lead to a maximum independent set. In this case, leave I unchanged, delete v from J and proceed to the next stage. If $|I| + 1 + t = k$, then set $I \leftarrow I \cup \{v\}$, delete v and all neighbours it has in J from J, and proceed to the next stage again. It is clear that the invariant is maintained and that the size of J shrinks in every stage. Thus the construction must end within at most n rounds, necessarily with a maximum independent set and in polynomial time. ■

Without proof we mention the following general fact:

Theorem 7.5 *For every NP-optimization problem $\mathcal{M}$ with an NP-complete decision problem, there is a polynomial time algorithm for solving (the construction problem for) $\mathcal{M}$ if and only if P = NP.*

7.3 Heuristics

In solving NP-hard problems there often is a choice: design a 'heuristic' that is fast but does not necessarily give an optimal solution, or design and tune an algorithm that solves the problem exactly at the risk that it is infeasible to run it on instances of more than moderate size.

In this part, we will see two heuristics to approach the Maximum Independent Set problem. A heuristic is an algorithm that is based on a clear strategy of searching in the space of all feasible solutions, but that does not guarantee an optimal solution. In all instances we consider, we assume that the isolated nodes have been removed: they are trivially part of any maximal independent set and need not be considered further.

7.3.1 Local Improvement

A first approach is to take some (maximal) independent set I and to try to improve it further by swapping some nodes in and out. We consider two different ways to improve I:

1 - improve: add one node that is independent to the set, and

2 - improve: delete one node and add two other nodes which do not disturb the independency.

Repeat the improve-steps in any order until none can be applied anymore. The improvement process is finite, because $|I|$ increases by 1 in every step. Observe that the resulting set I must be a *maximal* independent set.

A characteristic of many heuristics is that they just 'work'. In fact, the given improvement heuristic is fast and simple enough that it can be run 1000 times on an initially empty set I, in each run trying the improvement steps in random order. The largest maximal independent set I from these runs is likely to be a pretty good approximation to a maximUM independent set. Only further analysis or experimentation can tell.

Let M be any maximum independent set in a given graph G.

Theorem 7.6 (Khanna *et al.*, 1995) *Let I be any maximal independent set of G that results after a run of the improvement heuristic. Then $\mid I \mid \geq \frac{2}{\Delta+1} \mid M \mid$, where Δ is the maximum degree of a node in G.*

Proof: The bound holds when $\mid I \mid = \mid M \mid$ because $\Delta \geq 1$ in all instances we consider.

Assume $\mid I \mid < \mid M \mid$. Let $X = M \cap I$. It is clear that $M - X$ $(= M - I)$ and $V - I$ are not empty. We can conclude the following:

1. every node in $V - I$ has at least one edge to I (otherwise we could use 1-improve).
2. at most $\mid I \mid - \mid X \mid$ nodes from $M - X$ can have exactly one edge to a node in I (thus in $I - X$). Otherwise at least one node in $I - X$ would have an incoming edge from 2 nodes in $M - X$ and we could use 2-improve to swap them; note that the two vertices in $M - X$ cannot have an edge between them (because they are in M) or to any node in I.
3. at least $(\mid M \mid - \mid X \mid) - (\mid I \mid - \mid X \mid) = \mid M \mid - \mid I \mid$ nodes from $M - X$ have at least two edges to a node in I (thus in $I - X$).

Thus, counting the number of edges from $M - X$ to a node in $I - X$ we get (writing set names for their sizes now):

$$
\begin{aligned}
(I - X)\Delta &\geq (I - X) + 2(M - I) && (1)\\
I\Delta - X\Delta &\geq 2M - I - X && (2)\\
I(\Delta + 1) &\geq 2M + X(\Delta - 1) && (3)
\end{aligned}
$$

$$I(\Delta+1) \geq 2M \tag{4}$$
$$\frac{I}{M} \geq \frac{2}{\Delta+1} \tag{5}$$

and hence $\mid I \mid \geq \frac{2}{\Delta+1} \mid M \mid$. ■

If we take a closer look at the incoming edges of I, we can make some further observations in the analysis. There are $V - I$ nodes that have at least one edge to I, and there are at least $M - I$ nodes with at least a second edge to I, *viz.* to $I - X$. Thus, counting the number of edges to I again and using $\mid V \mid = n$ we obtain:

$$I\Delta \geq (V - I) + (M - I) \tag{6}$$
$$I\Delta \geq n + M - 2I \tag{7}$$
$$I(\Delta+2) \geq n + M \tag{8}$$
$$I \geq \frac{(1+\frac{M}{n})}{\Delta+2} n \tag{9}$$

Using that $\mid M \mid \geq \mid I \mid$, it follows from line (7) that the simple improvement heuristic *always* delivers a maximal independent set of size $\mid I \mid \geq \frac{1}{\Delta+1} n$. Line (9) shows an even better estimate, the larger M is.

The simple heuristic seems to do pretty well, especially because it can be run many times for a best result. Also it can be added on for 'final improvement' to any other algorithm that already gives good maximal independent sets.

7.3.2 Greedy algorithm

Another approach is to be less random and follow a very targeted heuristic. In the so-called *greedy* approach one tries to choose/schedule vertices that are 'the best' first, iteratively. For the Maximum Independent Set problem on a graph G, the greedy algorithm builds an independent set I by repeating the following step, starting with $G' = G$:

- select the node v with lowest degree in G', breaking ties arbitrarily,
- add v to I, and
- remove v and all its neighbours from G'.

The algorithm runs in polynomial time, and results in an independent set I that is maximal. The greedy algorithm has some points of choice, thus different runs of the algorithm may give different answers. Again one can take the best answer over a series of runs of the algorithm.

A lower bound on the quality of the resulting maximal independent set can be given. Let M be an arbitrary, not necessarily maximum or maximal, independent set in G and let the *average degree* of the nodes in M be α, with $\alpha \geq 1$. (Recall that α is not < 1, because isolated nodes are not considered.)

Lemma 7.7 *The greedy algorithm gives a maximal independent set I of size $\geq \frac{M}{(\alpha+1)}$.*

Proof: Considering the algorithm, we distinguish the following two kinds of steps depending on the choice of the node v in each step:

a. the algorithm takes a vertex v outside M, thus in $V - M$.

b. the algorithm takes a vertex v in M.

If the algorithm only takes steps of type b, the resulting set I must be equal to M and the theorem follows. Thus assume that the algorithm takes at least one step of type a.

First, we take a look at the steps of type a. Suppose in the i^{th} step, the algorithms takes a vertex from $V - M$ with exactly k_i neighbors in the set M in the remaining graph. Since the algorithm takes the vertex with lowest degree, these k_i neighbors have at least k_i edges incident on them. In the step we thus loose k_i nodes of M and (at least) k_i^2 edges incident to M. Assume that there are p steps of type a in total and let $\sum_{i=1}^{p} k_i = x$. Then $\sum_{i=1}^{p} k_i^2 \leq \alpha \mid M \mid$ and, by the Cauchy-Schwartz inequality, $\frac{x^2}{p} \leq \sum_{i=1}^{p} k_i^2$. (Note that $p > 0$.)

Next we take a look at the steps of type b. When the algorithm selects a vertex from the independent set M, we do not delete any other vertices from M in carrying out the step, because M is independent. Assume that there are q steps of type b in total. Then the greedy algorithm gives us an independent set with $p + q$ vertices, with $q \geq r =\mid M \mid -x$.

Now we can make the following estimates (writing set names for their sizes):

$$\begin{aligned}
\alpha M &\geq \frac{x^2}{p} && (10)\\
(\alpha+1)M &\geq \frac{x^2}{p} + x && (11)\\
(\alpha+1)M - x &\geq \frac{x^2}{p} && (12)\\
\frac{(\alpha+1)M - x}{x^2} &\geq \frac{1}{p} && (13)\\
p &\geq \frac{x^2}{(\alpha+1)M - x} && (14)\\
&\geq \frac{x^2}{(\alpha+1)M} && (15)\\
&\geq \frac{M}{(\alpha+1)} + \frac{r^2}{(\alpha+1)M} - \frac{2r}{(\alpha+1)} && (16)
\end{aligned}$$

Thus

$$p + q \geq \frac{M}{(\alpha+1)} + \frac{q^2}{(\alpha+1)M} - \frac{2r}{(\alpha+1)} + r.$$

As $\frac{2r}{\alpha+1} \leq r$, it follows that $\mid I \mid = p + q \geq p + r \geq \frac{|M|}{\alpha+1}$.

The second step, the substitution (11) needs some explanation. Because $M \geq x$, we may add M to the left part and x to the right part of the inequality. ■

Let's take a closer look at the analysis again. Let $v = v_i$ be the node chosen in the i-the step of the greedy algorithm, and let its degree in the remaining graph be d_i. Then $d_i + 1$ nodes

of degree at least d_i and thus at least $\frac{1}{2}d_i(d_i+1)$ edges are eliminated from the graph in this step. As the algorithm takes $|I|$ steps, it follows that:

$$\sum_{i=1}^{|I|}(d_i+1) = n$$

and

$$\sum_{i=1}^{|I|}\frac{1}{2}d_i(d_i+1) \leq m,$$

where $m = |E|$. Combining the two facts, one gets:

$$\sum_{i=1}^{|I|}(d_i+1)^2 \leq (n+2m).$$

and we note that by the Cauchy-Schwartz inequality:

$$\sum_{i=1}^{|I|}(d_i+1)^2 \geq (\sum_{i=1}^{|I|}(d_i+1))^2/(\sum_{i=1}^{|I|}1) = \frac{1}{|I|}n^2.$$

Thus $\frac{1}{|I|}n^2 \leq (n+2m)$, hence

$$\mid I \mid \geq \frac{1}{2\frac{m}{n}+1}n = \frac{1}{\Delta_0+1}n,$$

where $\Delta_0 = 2\frac{m}{n}$ is the average degree in G. Once again we see that we appear to get a pretty sizable maximal independent set, depending on the average degree of G.

Khanna *et al.* argued that one might as well run the greedy algorithm and the improvement algorithm separately and take the best result.

Theorem 7.8 (Khanna *et al.*, 1995) *The best of the greedy algorithm and the 'improver' gives a maximal independent set I of size $\mid I \mid \geq \frac{2.414}{\Delta} \mid M \mid$ where $\mid M \mid$ is the size of a maximum independent set.*

Theorem 7.9 (Halldórsson and Radhakrishnan, 1995) *There is an algorithm that, given any graph G, computes maximal independent sets I in G of size $\mid I \mid \geq \frac{6+o(1)}{\Delta} \mid M \mid$ where $\mid M \mid$ is the size of a maximum independent set.*

To finish the discussion of heuristic approaches to the Maximum Independent Set problem, you can wonder if there is a polynomial time algorithm that can give independent sets of size $\geq (1-\epsilon)M$ for this problem, for any specified ϵ. The answer is NO, unless $P = NP$. However, for the case of planar graphs, there is such a approximation algorithm.

7.4 Exhaustive Search

Exhaustive search algorithms are designed to search through the entire space of feasible solutions of a problem and find the exact solution or optimum. We will indicate a few basic approaches for the Maximum Independent Set problem.

7.4.1 Enumeration

In an enumerative approach one might try to enumerate all feasible solutions and look for the best one among them. One might even enumerate a *superset* of the feasible solutions, if this is easier and faster (despite the fact that one has to sift out infeasible solutions as well). For the case of the Maximum Independent Set problem for a given graph $G = (V, E)$ with n nodes, one may consider the following enumerative techniques.

- *Enumerate all subsets of* V. This is a straightforward but not very enticing technique: with $|V| = n$, the number of possible subsets equals 2^n and the independency test costs up to $O(m)$ computation steps for each subset. For sizes $n \geq O(100)$, this may be prohibitive.

Exercise. Show how to enumerate all subsets of a finite set without repetitions with an average of $O(1)$ operations per next subset generated.

- *Enumerate all* k*-subsets of* V *and use binary search on* k. One can try to save by enumerating only the subsets of a size 'close' to the suspected size of a maximum independent set. Also, using an algorithm $A = A(k)$ that enumerates only the $\binom{n}{k}$ subsets of given size k, one could search for a maximum independent set by binary search: use $A(k)$ to determine the largest k for which there still exist subsets that are an independent set. If $|M|$ is the size of a maximum independent set, at most $O(\log |M|)$ calls to $A(k)$ suffice. For graph G with small maximum independent sets, this might be a possible approach.

Exercise. Show how to enumerate all k-subsets of a finite set without repetitions, with an average of $O(k)$ operations per next subset generated.

The question arises how to limit the collection of subsets that is enumerated to a more targeted class.

- *Enumerate all possible maximal independent sets of* G. This is not necessarily a great improvement, because of the following fact:

Theorem 7.10 (Moon and Moser, 1965) *An n-node graph can have up to $3^{n/3} \approx 1.445^n$ maximal independent sets and this bound is tight.*

Despite this, it is an interesting problem to actually design a fast algorithm that enumerates all maximal independent sets without repetitions and without spending much extra time on the independence tests. One of the most practical algorithms for it is based on backtracking and due to Bron and Kerbosch (1973), who actually presented it as an algorithm to find all maximal cliques. A striking result is the '4-Japanese algorithm'.

Theorem 7.11 (Tsukiyama *et al.*, 1977) *The maximal independent sets of an n-node graph can be enumerated with a delay of $O(mn)$ time between successive sets and no more than linear space.*

The result has been improved for special cases e.g. for planar graphs. Johnson *et al.*[2] showed that even a *lexicographic* enumeration of all maximal independent sets can be achieved with only a polynomial delay between successive sets (at the expense of a greater amount of space).

7.4.2 Faster Search

We have seen that with an enumerative approach to the Maximum Independent Set problem, one may have to investigate exponentially many possible sets. It seems very expensive to enumerate all maximal independent sets, if all we need is one maximum independent set. We will see now that with an intelligent way of backtracking, one can avoid useless parts of the search space and limit the search complexity to $O(\alpha^n)$ steps, for some $\alpha < 3^{1/3} \approx 1.445$. The essential idea of the faster algorithm is to target the search for candidate sets.

Lemma 7.12 *Let v be any node of G and let $N(v)$ be the set of neighbors of v in G. Then a maximum independent set of G is obtained by taking the largest of the set $I \cup v$ with I a maximum independent set of $G - (\{v\} \cup N(v))$ and a maximum independent set of $G - \{v\}$.*

Exercise. Prove the lemma.

Consider the following algorithm based on the lemma above, which solves the the maximum independent set problem:

$A(G)$
a. if G has only one node, then output this node and return.

b. choose a 'good' v in G.

c. output the largest set of:

c1 $\{v\} \cup A(G(V - (\{v\} \cup N(v))\})$, and
c2 $A(G(V - \{v\}))$

and return.

The question to ask first is how we can choose a good v. Best is to take a node with large degree, because it gives a 'large' reduction in the recursive step c2. To make sure that we can always choose a node of large degree in step c, we must handle some cases where G only has low degree nodes directly. In particular, replace step a by:

a. if all nodes of G have degree ≤ 2, determine and output a maximum independent set in G directly and return.

Observe that if all nodes have degree ≤ 2, then G is a cycle of nodes or a chain of nodes or a disjoint combination thereof. If G has this form, then a maximum independent set can easily be found in it.

Now we can also modify step b of the algorithm. The modification in step a guarantees that a node v of degree ≥ 3 always exists (and it can easily be found) if one gets to step b.

b. choose a node v of some degree ≥ 3 in G.

Lemma 7.13 *Algorithm A solves the Maximum Independent Set problem in* $O(1.38^n)$ *steps.*

Proof: Assume the algorithm needs $c(n)$ computation cycles to solve the problem for a graph of size n. Then we have for $n \geq 4$:

$$c(n) \leq c(n-1) + c(n-4) + \gamma n^2$$

Estimating $c(n)$ by λa^n steps for some a and λ, the bound is satisfied when we choose a such that

$$a^n \approx a^{n-1} + a^{n-4} + o(1)$$

and λ such that λa^n bounds $c(n)$ for e.g. n up to 10. Now

$$a^4 \geq a^3 + 1 + \epsilon$$

gives $a \approx 1.38$.

■

Algorithm A obviously looks better than the algorithms that enumerate all maximal independent sets. It is interesting that by elementary means we have come done from an initial 2^n bound to a $O(1.38^n)$ bound for the exact solution of the Maximum Independent Set problem. By more elaborate reductions and case analyses, the upperbound on the complexity of exact solvers for the Maximum Independent Set problem has been steadily lowered. The table below summarizes some of the known results.

year	author(s)	upperbound
1977	Tarjan and Trojanowski	$2^{0.333n}$
1986	Jian	$2^{0.304n}$
1986	Robson	$2^{0.276n} \approx 1.211^n$
1999	Chen, Kanj and Jia	1.18842^n

References

[1] J. Chen, I.I. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. In: P. Widmayer, G. Neyer, S. Eidenbenz (Eds.), *Graph Theoretic Concepts in Computer Science*, Proc. 25th Int. Workshop (WG'99), Lecture Notes in Computer Science Vol. 1665, Springer-Verlag, Berlin, 1999, 313-324.

[2] D.S. Johnson, M. Yannakakis, and C.H. Papadimitriou. On generating all maximal independent sets, *Information Processing Letters* 27 (1988) 119-123.

[3] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *Electronic Colloquium on Computational Complexity*, Report TR95-023, ECCC.

[4] J.M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms* 7 (1986) 425-440.

[5] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets, *SIAM Journal on Computing* 6 (1977) 505-517.