Algorithms and Computation

(grad course)

Lecture 7: Algorithms for NP-Complete problems

Instructor: Hossein Jowhari

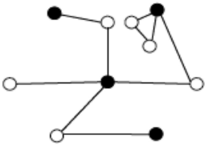
jowhari@kntu.ac.ir

Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

Fall 2021

Dominating Set Problem

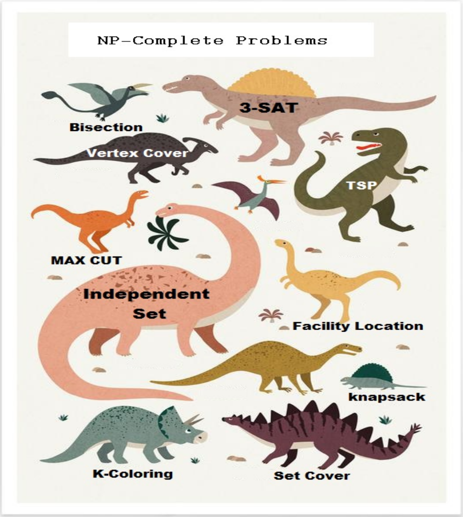
Definition: Given the unidrected graph $G = (V, E)$, the set $D \subseteq V$ is a dominating set for G if every vertex in G is either contained in D or has a neighbor in D.



The Dominating Set problem asks if the given graph G has a dominating set of size at most k?

Show that Dominating Set problem is NP-Complete.

Inside NP-Completeness



NP-Complete problems are equivalent with respect to polynomial-time reducibility but each one is a different beast.

3-SAT problem

Is there an assignment that satisfy the Boolean formula ϕ (in $3-CNF$ format) defined on n variables?

$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge \ldots \wedge (\ldots)$$

Brute-force algorithm tries all 2^n different assignment. Running time is $O(2^n m)$ when m is the number of clauses.

Is there a faster algorithm for 3-SAT?

Yes there is! but not much faster.

A general framework for solving 3-SAT, DPLL

Davis-Putnam-Logemann-Loveland

ALG(ϕ: 3CNF formula, P: Partial Assignment)

Pick a variable x (IN A CLEVER WAY):

- try ALG(ϕ, $P \cup \{x = T\}$)
- try ALG(ϕ, $P \cup \{x = F\}$)

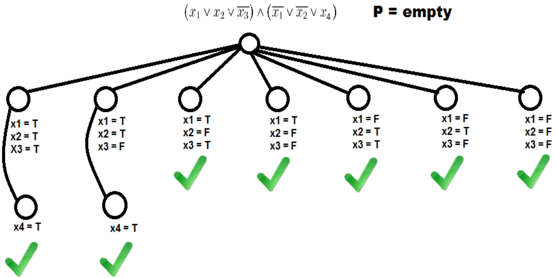
Some ways to pick the next variable x:

- Pick x that appears in many clauses. Try $x = T$ first.
- Pick x that appears a lot but $\overline{x}$ appears a little.

The 7 algorithm

ALG(ϕ: 3CNF formula, P: Partial Assignment)

- If ϕ is in $2-CNF$ use 2SAT algorithm to solve it.
- Find $C = (x \vee y \vee z)$ a clause not touched.
- For all 7 ways to set (x, y, z) so that $C = TRUE$. In each case, let P' be the extension of P to the new assignment. Try ALG(ϕ, P')



Algorithm analysis

$$T(n) \to \text{ running time of the algorithm}$$

$$T(0) = 1$$

$$T(n) = 7T(n-3)$$

$$T(n) = 7^2 T(n - 3 \times 2)$$

$$T(n) = 7^i T(n - 3i)$$

Plug $i = n/3$.

$$T(n) = 7^{n/3} O(1) = O((7^{1/3})^n) = O(1.913^n)$$

- There is an $O(1.439^n)$ deterministic algorithm for 3-SAT.

  (Konstantin Kutzkov, Dominik Scheder, 2010)

- There is a $O(1.321^n)$ randomized algorithm for 3-SAT.

  (Timon Hertli, Robin A. Moser, Dominik Scheder, 2011)

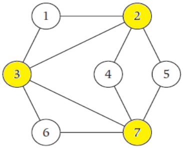
- The **ETH** (*Exponential Time Hypothesis*) asserts that no algorithm can solve 3-SAT in $2^{o(n)}$ time.

- Note that all problems in NP have $2^{\mathrm{poly}(n)}$ time algorithms.

Vertex Cover

Vertex cover with parameter k. Given the undirected graph G, does G have a vertex cover of size less than or equal to k?

Recall: A vertex cover for graph $G = (V, E)$ is a subset $S \subseteq V$ where each edge $(u, v) \in E$ has at least one endpoint in S.

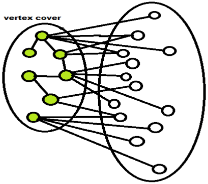


Brute-force algorithm examines every subset S of size k and checks if S covers the edge set E. There $\binom{n}{k} \approx n^k$ subsets. The algorithm runs in $O(n^k kn) = O(n^{k+1}k)$ time.

A better algorithm for Vertex Cover

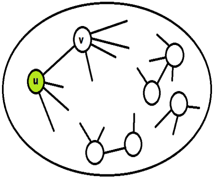
There is an algorithm for the Vertex Cover problem with parameter k that runs in $O(2^k kn)$ time. In particular when $k = O(\log n)$ the algorithm runs in polynomial time.

Observation 1: If G has a vertex cover of size $\leq k$, then G can have at most $k(n-1)$ number of edges.



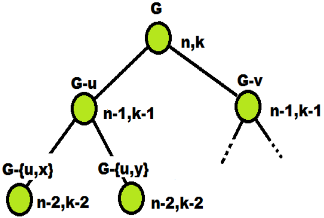
First Step of the algorithm: If $|E| > k(n-1)$ then we reject the input and claim the size of the min vertex cover is greater than k

Observation 2: Let (u, v) be any edge in the graph. G has a vertex cover of size at most k if and only if $G - \{u\}$ or $G - \{v\}$ has a vertex cover of size at most $k - 1$.



Recursive Algorithm:

```
To search for a k-node vertex cover in G:
  If G contains no edges, then the empty set is a vertex cover
  If G contains> k |V| edges, then it has no k-node vertex cover
  Else let e = (u, v) be an edge of G
    Recursively check if either of G−{u} or G−{v}
                has a vertex cover of size k − 1
    If neither of them does, then G has no k-node vertex cover
    Else, one of them (say, G−{u}) has a (k − 1)-node vertex cover T
      In this case, T ∪ {u} is a k-node vertex cover of G
    Endif
  Endif
```



Analyzing the algorithm

$T(n, k) \rightarrow$ running time of the algorithm

There exists constant c where

$T(n, 1) \leq cn$

$T(n, k) \leq 2T(n-1, k-1) + ckn$

We can show (by induction) $T(n, k) \leq 2^k cnk$.

This shows that $T(n, k) = O(2^k nk)$

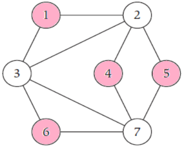
There is a better algorithm for Vertex Cover that runs in $O(1.2832^k k + nk)$ time.

An algorithm with a running time $f(k).\mathrm{poly}(n, k)$ for a NP-complete problem is called a fixed parameter algorithm.

MAX Independent Set

MAX Independent Set with parameter k. Given the undirected graph G, does G have an independent set of size at least k?

Recall: An independent set in graph $G = (V, E)$ is a subset $S \subseteq V$ where there is no edge between the vertices in S.



Brute-force algorithm examines every subset S of size k and checks if S is an independent set. There $\binom{n}{k} \approx n^k$ subsets. The algorithm runs in $O(n^k kn) = O(n^{k+1}k)$ time.

- There is no known fixed parameter algorithm for the Independent Set problem.

- In other words, we do not know of any $O(f(k)\mathrm{poly}(n,k))$ algorithm for the Independent Set problem (or for the k-Clique problem)

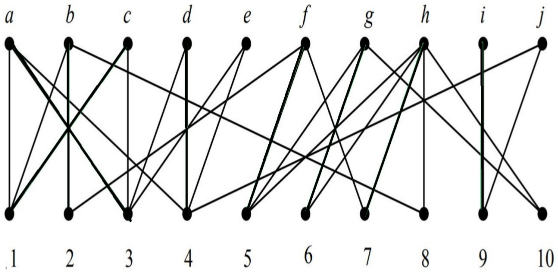
- **Fact**: If there is a $O(f(k)\mathrm{poly}(n,k))$ algorithm for k-Clique then it would lead to a $2^{o(n)}$ algorithm for the n-variable 3-SAT (i.e the ETH fails)

- Some NP-Complete problems (like the Vertex Cover problem) have fixed parameter algorithm. For example the k-PATH problem and the k-Disjoint Triangle problem.

NP-Complete problem on Special Inputs

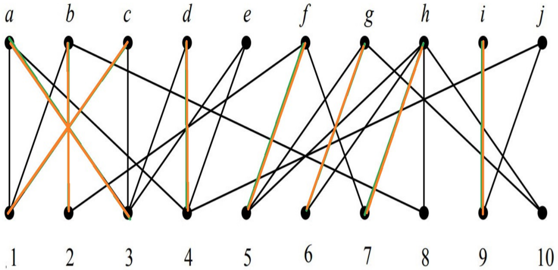
- When the input (of the algorithm) is restricted to special instances the problem might become tractable.
- For example in the SAT problem when all the clauses are of length 2 or 1 (2SAT) we have polynomial time algorithm for that.
- Independent Set problem is easy to solve when the input graph is a tree.
- Independent Set and Vertex Cover problems remain NP-Complete even on planar graphs of degree at most 3.
- Hamiltonian Cycle remains NP-Complete on Bipartite graphs.

Vertex Cover for Bipartite Graphs



Kőnig's theorem: In any bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.

Conclusion: There is a polynomial time algorithm for the Vertex Cover in bipartite graphs.



$$m(G) = 8$$

Find a vertex cover of size 8.

Short Proof of Kőnig's theorem (non-constructive)

Lemma 1: In every graph G, we have $v(G) \geq m(G)$.

Proof: No vertex can cover two edges of a matching.

Lemma 2: In every bipartite graph G, we have $v(G) = m(G)$.

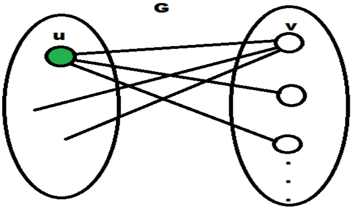
Proof: If G is is a path or cycle then it is true (check it!)

Now for the sake of contradiction, suppose the statement is wrong. In other words, there is a bipartite graph G where $v(G) > m(G)$.

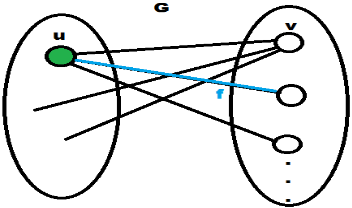
Let $G = (V, E)$ be the minimal counter-example. It means:

- For any edge $e \in E$, $v(G - e) = m(G - e)$
- For any vertex $v \in E$, $v(G - v) = m(G - v)$

Since G is not a path or a cycle, it must have a vertex with degree 3.



Suppose $m(G - v) < m(G)$. (It means a maximum matching of G must use v) Then by minimality $G - v$ has a vertex cover W' of size less than $m(G)$. Hence $W' \cup \{v\}$ is a vertex cover of G of size $m(G)$. Contradiction!



Now suppose there is maximum matching of G that does not use v. So u must be part the maximum matching. Let f be an edge incident on u and not part of the maximum matching.

Let W' be a cover of $G - f$. We have $|W'| = m(G)$ (by minimality). W' should cover every edge of the maximum matching in $G - f$. Since v is not part of the matching W' cannot contain v. Therefore W' must contain u and there it should be a cover for G. A contradiction!