8 On Difficulty of Graph Problems

How can one compare between easy and hard graph problems?

How can one assess the difficulty of a newly formulated problem?

□

Brief outline of this lecture

- Examples of easily (and efficiently) solvable graph problems.
- Colouring and Hamiltonicity – two traditional hard grah problems.
- How to assess difficulty of a problem – using problem reductions.

8.1 Some Easily Solvable Problems

During the course, we have provided several really simple algorithmic solutions for basic questions on graphs, e.g.:

- testing connectivity and finding the connected components, □
- computing the distance and a shortest path in a graph, □
- computing a minimum spanning tree in a weighted graph, □
- finding a maximum flow and a minimum cut in a network, □
- finding a maximum matching in a bipartite graph, □
- testing 2-colourability of graphs, □
- testing isomorphism of trees. □

Well, that was about algorithm design, and what about the theory side?

Nice characterizations of problems

Such "easily solvable" problems typically come hand in hand with nice theoretical characterizations of the solutions, and of their *existence* in greater generality. □

For example. . .

- *Good characterizations* of problems; either
  - we have got an obvious or easily verifiable solution, or
  - we can find an obvious or easily verifiable obstacle. □

Recall some examples of good characterizations:

- For *connectivity* in a graph; either
  - we can find an x-y walk (or path) in the graph, or
  - a vertex subset $X \subseteq V$ such that $x \in X$, $y \notin X$ and no edge leaves X. □
- For maximum *matching in a bipartite graph*; either
  - we can find a matching with k edges, or
  - we have got a vertex cover with $< k$ vertices. □
- For 2-*colourability* of a graph; either
  - we can find a proper 2-colouring (bipartition), or
  - a cycle of an odd length.

On the dark side

For problems which are "hard to solve", nice theoretical characterizations of their solutions typically do not (and cannot) exist, e.g.; □

- for the 3-*colourability* problem one can easily verify that a given colouring is proper (or not), but that is not sufficient! □
- there is no known good way of showing that a 3-*colouring* does not exist in a given graph, other than trying all possibilities.

Even though we have no such directly provable relation, an absence of a nice characterization of the solutions indicates hardness of a problem... □

Much worse, for some other hard graph problems the correct solution is even not easily verifiable; □

- considering, say, 4-*choosability* of a given graph, how can one verify that for all possible assignments of lists of 4 colours a proper colouring does exist?

More Involved Problems

Obviously, life is not always as easy as in the previous slides. . . □

There exist graph problems for which an efficient algorithmic solution exists, but both the algorithms and related theoretical characterizations of their solutions are rather involved, e.g.; □

- a maximum matching in general graphs, and *maximum weighted matching*, □
- testing planarity and *finding a plane drawing* in linear time (interestingly, the best algorithms are not dir. related to the Kuratowski thm.),
- the *isomorphism of planar graphs* in linear time (needs planarity as a tool). □

Though not so often, there are examples of problems having theoretically fast algorithms, but which are practically unusable, e.g.; □

- the isomorphism of 3-regular graphs, □
- testing graph embeddability in (fixed) higher surfaces,
- possibility to draw a graph in the plane with, say, < 100 edge crossings, □
- a graph minor testing, and consequently all minor-closed decision properties.

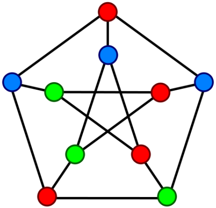
Strange case of the isomorphism

partial obstacles

practically very fast

randomized obstacle

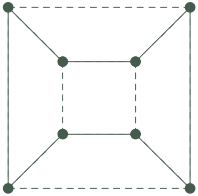
8.2 Colouring and Hamiltonicity

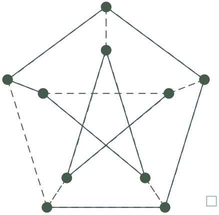


.... See Lecture 6.

Hamiltonian Graphs

Another typical hard graph question is that about an existence of a cycle or a path through all the vertices of the given graph (an older and simplified setting of a "travelling salesman"):





Definition: A cycle C in a graph G is a *Hamiltonian cycle* if C spans all vertices of G.

Analogously, a *Hamiltonian path* P in G is a path in G spanning all the vertices. □

A graph G is *Hamiltonian* if G contains a Hamiltonian cycle.

The same terminology applies in the case of digraphs with dir. cycles and paths. □

The Hamiltonian cycle problem is also related to some attempts to solve the 4 colour problem: precisely, if every 3-regular planar graph was Hamiltonian, then the 4 colour theorem would follow easily. □Well, but this claim later turned out not to be true. . .

Hamiltonian cycle / path = Hamiltonovská kružnice / cesta

Hamiltonian tournaments

Definition: A digraph G is called a *tournament* if for every vertex pair $u, v \in V(G)$, $u \neq v$, exactly one of the arcs (u, v), (v, u) is in G.

In other words, a tournament on n vertices results by arbitrarily orienting each edge of K_n. □

Proposition 8.1. *Every tournament G contains a Hamiltonian (directed) path.*

Proof: We apply a straightforward induction on n, the number of vertices of G. □

- If $n \in \{1, 2\}$, then G itself is a directed path.
- Assume $n \geq 3$, and choose $v_0 \in V(G)$ arbitrarily. Form $G_0 = G \setminus v_0$ by deleting the vertex v_0, and let $P = (v_1, v_2, \dots, v_{n-1})$ be a dir. Hamiltonian path in G_0 (in this order of vertices), which exists by the induction assumption. □
- If $(v_{n-1}, v_0) \in E(G)$, then P with v_0 at the end is a Ham. path in G. Similarly, if $(v_0, v_1) \in E(G)$, then P prefixed with v_0 is a Ham. path in G. □

  Otherwise, let $j \in \{1, \dots, n-1\}$ be the least index such that $(v_0, v_j) \in E(G)$, and hence $(v_{j-1}, v_0) \in E(G)$. Then $(v_1, \dots, v_{j-1}, v_0, v_j, \dots, v_{n-1})$ is a Ham. path in G.

□

tournament = turnaj (spec. druh or. grafu)

Proposition 8.2. *A tournament G contains a Hamiltonian (directed) cycle if, and only if, G is strongly connected.*

Proof:

.... $\square$

tournament = turnaj (spec. druh or. grafu)

Dirac's theorem

Theorem 8.3. *Every graph G on $n \geq 3$ vertices and with minimum degree $\geq n/2$ is Hamiltonian.* □

Proof (a sketch): Let $P \subseteq G$ be a longest possible path in the graph G, such that the vertices on P occur in this order; $(u_0, u_1, \dots, u_k)$. □It is easy to claim:

- every neighbour of u_0 or u_k belongs to P (since otherwise we prolong P),
- since $d_G(u_0), d_G(u_k) \geq \frac{1}{2}n \geq \frac{1}{2}(k+1)$, by the pigeon-hole principle, there is $0 < i < k$ such that $u_0u_{i+1} \in E(G)$ and $u_ku_i \in E(G)$ (draw a picture), □
- consequently, we have got a cycle $C \subseteq G$ on $V(P)$ in the cyclic order of vertices $(u_0, u_{i+1}, u_{i+2}, \dots, u_k, u_i, u_{i-1}, \dots, u_0)$. □

That is all since, if there was a vertex $x \in V(G) \setminus V(C)$ (missed by C), then x would have a neighbour on C and we would get another path longer than original P, a contradiction. □

8.3 Problem Reductions

The term *polynomial reduction* is formally defined in every computational complexity course. However, this course is not on complexity but on graphs, and so we stay with an informal explanation of the concept.

- Imagine that somebody asks us to solve an instance of Problem A, which we do not know how to solve, but we have a nice algorithm / solution for another Problem B. □
- If we are lucky, then we can take an input of Problem A, and *transform it* to an input of Problem B such that a (possible) solution is preserved (meaning that we can always "decode" a solution for A from a solution for such transformed B). □
- Obviously, our tranformation should be "easy and efficient". We say that we have got a

  reduction from Problem A to Problem B.

If we have got a reduction from A to B then, informally, Problem A is not harder (of at most the *same difficulty*) than Problem B.

From a different point of view; if we know that Problem A cannot be solved nicely, then neither can Problem B (a "hardness reduction").

Example 8.4. *Show that the following two problems are of the same difficulty:*

A: to decide whether a given graph G has a Hamiltonian path,

B: to decide whether G has a Hamiltonian path starting in a given vertex.

To provide a proof, we have to show two problem reductions; from A to B and from B to A. We start with the latter one.

- (From B to A): Imagine somebody asks for a Ham. path in G starting with $v \in V(G)$, and we could only solve the general Ham. path problem.

  Then we construct a graph G' from G by adding a new vertex v' adjacent only to v. Any Ham. path in G' has to start in v' and continue with v, and so it gives a Ham. path in G starting with v.

- (From A to B): We could solve general Ham. path by asking for a Ham. path starting in every vertex of G separately, but there is a much nicer alt. solution.

  Construct a graph G'' from given G by adding a new vertex x adjacent to all $V(G)$, and ask for a Ham. path starting in x. Trivially, a Ham. cycle in G exists iff such a cycle can be prolonged till x to make a Ham. cyce in G''.

□

Example 8.5. *Show that the following two problems are of the same difficulty:*

- *to decide whether a given graph G is 3-colourable,*
- *to decide whether a given graph G is 4-colourable.*

To reduce from 3-colourability to 4-colourability is very easy, simply make a graph G' from original G by adding a new vertex x adjacent to all $V(G)$. Then every 3-colouring of G is in a one-to-one correspondece with a 4-colouring of G' using colour 4 at the vertex x.

A converse reduction is much more involved and tricky .... $\Box$

Class $\mathcal{NP}$ and $\mathcal{NP}$-completeness

Decision problems

- *Decision problems* are those, roughly saying, whose answer can be YES/NO. □
- This seems quite restrictive, however;
  - for the *colouring problems*, we usually ask if a graph G is k-colourable, meaning whether $\chi(G) \leq k$, but not about the precise value of $\chi(G)$, □
  - simiarly, in the *clique and independent set* problems, we may ask whether $\omega(G) \leq k$ and $\alpha(G) \leq k$, and □
  - knowing how to solve the decision variant of a problem usually means we can also find a witnessing solution. □

Class $\mathcal{NP}$

- A prominent rank among graph decision problems belongs to those problems in which an answer YES can be verified, with the help of a suitable *advice* (oracle), by an efficient procedure/algorithm. □
  - In computational complexity, the class of such problems is called $\mathcal{NP}$. □
- Actually, the decision version of most common graph problems are of this kind.

  For example, in the colouring problem the advice is a proper colouring of the given graph which can be readily verified. Likewise for the Hamiltonian problem, the advice is a Hamiltonian cycle, etc.

decision problem = rozhodovací problém

The Satisfiability problem

The following one is the "master $\mathcal{NP}$-complete" problem:

Problem 8.6. ***3-SAT (a special version of satisfiability SAT)***
The following problem is $\mathcal{NP}$-complete (reading "hardest in $\mathcal{NP}$"), meaning that it belongs to the class $\mathcal{NP}$ and no other problem in $\mathcal{NP}$ can be more difficult: □

Input: A propositonal logic formula Φ in a conjunctive normal form, such that every clause of Φ contains ≤ 3 literals.

Output: Is there a valuation of the Φ-variables that makes Φ true?

For instance, $\Phi \equiv (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$. □

The true informal meaning of $\mathcal{NP}$-completeness of a problem is as follows;

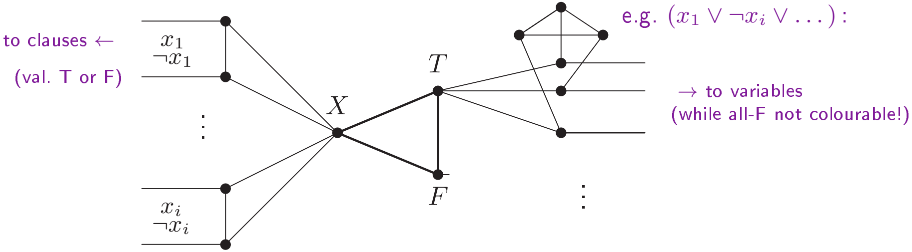
- *"I am the hardest problem in the very natural class $\mathcal{NP}$, and if you could solve me, then you would be able to solve everything in $\mathcal{NP}$."* □
- Consequently, it is very likely that $\mathcal{NP}$-complete problems cannot be solved nicely (again, we refer to computational complexity courses for a precise definition of this).
- As it turns out, many typical problems in graph theory are of the same, "highest", difficulty—they are $\mathcal{NP}$-complete. We will outline this fact, in a series of problem reductions, next.

SAT = problém splnitelnosti logických formulí v konjunktivní normální formě

8.4 Sample Reductions for Hard Problems

Problem 8.7. ***3-COL (3-Colouring of graphs)***
The following problem is as hard as SAT:
Input: A graph G.
Output: Can the vertices of G be properly *coloured* using three colours?

Proof (a sketch): The problem is in $\mathcal{NP}$ and we construct a reduction from 3-SAT. □
For a given formula Φ we construct a graph G_Φ: The basis of the construction of G_Φ is a triangle with vertices denoted by X, T, F. Each variable x_i in Φ is assigned a vertex pair adjacent to X. Each clause of Φ is assigned a subgraph on 6 vertices (three of them adjacent to T), as in the picture. Then the remaining free "halfedges" are joined together in the way corresponding to the literals (x_i or $\neg x_i$) in the clauses.



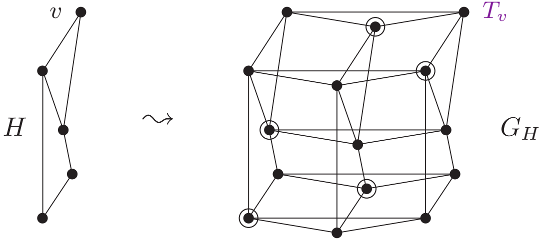
Then one may easily check that G_Φ has a 3-colouring iff Φ is satisfiable. □

Problem 8.8. ***IS (Independent Set)***
The following problem is as hard as SAT and 3-COL:

Input: A graph G, and an integer k.

Output: Is there an *independent set* (i.e., a subset of the vertices with no edges between them) of size at least k in G? □

Proof: The problem is in $\mathcal{NP}$ and we construct a reduction from 3-COL.
Let H be a graph on n vertices which should be 3-coloured. We set $k = n$, and construct a graph G_H made of three disjoint copies of H as shown in the picture:



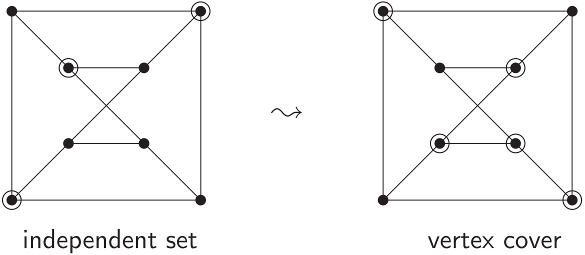
Assume $c : V(H) \to \{1, 2, 3\}$ is a 3-colouring of H. Then one can choose $k = n$ indep. vertices in G_H; for each $v \in V(H)$ choosing the $c(v)$-th copy of v in the graph G_H. □

Convers., if I is an indep. set in G_H of size $k = n$, then every triangle T_v, $v \in V(H)$, intersects I prec. in one vertex. This determines one of three colours for v in H. □

Definition: A *vertex cover* in a graph G is such a set $C \subseteq V(G)$ that every edge of G is incident with a vertex of C, i.e. $G - C$ is independent.

Problem 8.9. ***VC (Vertex Cover)***
The following problem is as hard as SAT and IS:

Input: A graph G, and an integer ℓ.
Output: Is there a vertex cover of size at most ℓ in G? □



Proof: The problem is in $\mathcal{NP}$ and we construct a really trivial reduction from IS.

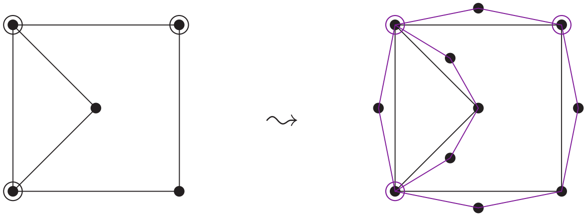
Notice that the complement $C = V(G) \setminus I$ of an arbitrary independent set I is actually a vertex cover, and vice versa. So the reduction works with the same graph G and with $\ell = n - k$ (where k is the desired IS size). □

Definition: A *dominating set* in a graph G is a set $D \subseteq V(G)$ such that every vertex of G not in D has a neighbour in D.

Problem 8.10. ***DOM (Dominating set)***
The following problem is as hard as SAT and VC:

Input: A graph G, and an integer ℓ.
Output: Is there a dominating set of size at most ℓ in G? □

Proof: The problem is in $\mathcal{NP}$ and we construct an easy reduction from VC.
Given any graph H, we construct an input graph G_H for the DOM problem as follows: For every edge $e \in E(H)$, a new vertex v_e is added, forming a triangle with e.



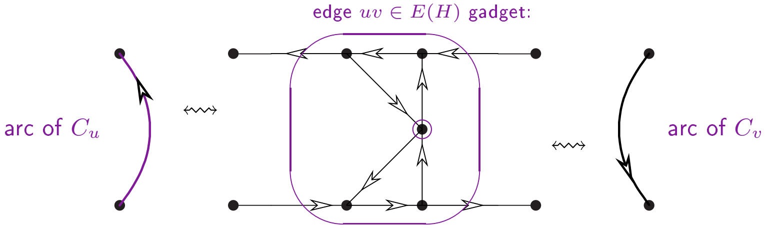
vertex cover $\leadsto$ dominating set

Now a vertex cover of the former graph is the same as a dominating set in the latter graph (any domin. set in the latter can be "pushed away" from the new vertices). □

Problem 8.11. ***HC (Hamiltonian cycle, directed)***
The following problem is $\mathcal{NP}$-complete:

Input: A digraph G.
Output: Is there a directed cycle in G passing through all the vertices?□

Proof (a sketch): The problem is in $\mathcal{NP}$ and we outline a reduction from VC.
Given any graph H and integer ℓ (an instance of VC), we construct a digraph G_H for the HC problem as follows. Every vertex $v \in V(H)$ is transformed into a directed cycle C_v of length $d_H(v)$ and each arc of C_v is then further transformed into one side of the gadget in the following picture, for every edge $uv \in E(H)$:

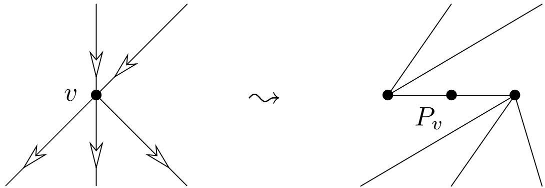


The point of this gadget is that while traversing it, one has to return to the same C_u as started from. □Finally, exactly ℓ new vertices are added to G_H as adjacent to and from every one of transformed C_v's (this models that we can "jump across" altogether ℓ of C_v's while covering all the edge gadgets). □

Problem 8.12. ***HAM (Hamiltonian cycle)***
The following problem is as hard as SAT and HC:

Input: A graph G.
Output: Is there an (undirected) cycle in G passing through all the vertices? □

Proof:



It is an easy reduction from the previous problem HC. Each vertex v of a directed graph H is replaced with three vertices forming a path P_v in the graph G_H. □

Then the directed edges coming into v are joined to the first vertex of P_v, while the edges leaving from v are joined to the last vertex of P_v. Having to travers also the middle vertex of P_v now "enforces" the right direction of passing throuh each P_v as through the original vertex v of H. □

8.5 Appendix: The interesting story of Vertex Cover

Consider the (at first glance) very similar problems of a vertex cover and of a dominating set in a graph—both are among the classical *$\mathcal{NP}$-complete problems*. Yet, we discover a huge difference between them, briefly outlined as follows. □

- If, in the computational complexity analysis, we focus on the value of the input parameter k, then we still cannot solve Dominating Set in a better way than exhaustively checking (almost) all k-tuples of vertices.

  Even when k is fixed small, say $k = 10,\ 20$, this an intractable problem. □

  What does it mean "cannot solve"?

  In this particular case, a solution to Dominating Set significantly faster than checking all k-tuples of vertices, would violate the Exponential Time Hypothesis. □

- On the other hand, a Vertex Cover of size k can be decided by a very simple algorithm running in time $O(2^k \cdot n)$, which is quite usable for small fixed values of k such as $k = 10,\ 20$, giving actually a *linear time algorithm*!

Algorithm 8.14. ***k-VC (Vertex Cover)***

For any fixed parameter k we are solving the following problem.

Input: A graph G.

Output: Is there a vertex cover of size at most k in G? □

We initialize $C = \emptyset$ and $F = E(G)$.

- *If $F = \emptyset$, then C is returned as a vertex cover.*
  If, otherwise, $|C| \geq k$, then the return value is "NO". □
- *We pick an arbitrary edge $f = uv \in F$, and for each of its ends $x = u, v$ we do:*
  - *$C' = C \cup \{x\}$, and the edge set F' results from F by removing all the edges incident with x in G;*
  - *the algorithm is called recursively for G, C' and F'.* □

Finally, how many (self-)recursive calls occurs in Algorithm 8.14 altogether? Every call generates two further recursive calls, but only up to a fixed depth k. Hence the total running time is asymptotically only $O(2^k \cdot n)$. □

Remark: The factor 2^k can be improved by more careful choice of branching. (2006: 1.2738^k)