NP, Karp reduction

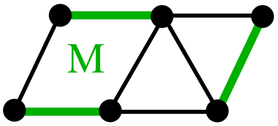
László Papp

BME

18th of April, 2023

Perfect matching

Definition: A matching M is a **perfect matching** of graph G if it covers all the vertices of G.



Problem PERFECT MATCHING
Input: A graph G.
Question: Does G have a perfect matching?

Claim
PERFECT MATCHING is in P.

Proof: We can run the augmenting algorithm to find a maximum matching. If it covers all the vertices then the answer is yes, otherwise it is no. The algorithm runs in polyonimal time.

The NP complexity class

Definition: Let π be a decision problem. We say that π is in class NP if the following criteria hold:

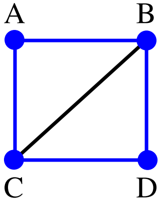
1. For each input I, if the answer for I is YES, then there is a witness W_I.
2. The size of W_I is polynomial in the size of I.
3. We can verify the YES answer for I by a polynomial time algorithm, whose input is I and W_I.

Example: PERFECT MATCHING $\in$ NP.

- The witness W_G is an encoding of a perfect matching.
- Since a matching is a subgraph of G, the size of W_G is not bigger than the size of G. So $|W_G|$ is polynomial in $|G|$.
- The verification algorithm reads the edges of W_G one by one, mark the endpoints of each of them and check whether the two endpoints are adjacent in G. If each vertex of G is marked exactly once, then W_G is a perfect matching, otherwise it is not. This can be done in $O(|V(G)|)$ time, so this is a polynomial algorithm.

HAMILTONIAN is in NP

Problem HAMILTONIAN
Input: A graph G
Question: Does G have a Hamiltonian cycle?



$W_G =$A,B,D,C

The witness W_G is an encoding of the Hamiltonian cycle. It is a list of the vertices in a proper order: Vertices which are adjacent in the list are adjacent in G. The first and the last vertices of the list are also adjacent in G.
The verification algorithm checks these adjacency conditions and whether each vertex of G is contained exactly once in W_G.
The verification can be done in $O(n^2)$ time, where n is the number of vertices of G.

Remark: Most mathematicians and computer scientist believe that HAMILTONIAN is not in P.

Transforming optimization problems into decision problems

If we have an optimization problem, where we want to minimize (or maximize) the objective function f over the set of solutions $\mathbf{A}$, then we can transform it into the following decision problem:

Input: The input of the optimization problem and a number k.
Question: Is there an element $x \in \mathbf{A}$ such that $f(x) \leq k$ (in case of maximization: $f(x) \geq k$)?

Examples:
Maximum matching problem
Input: A graph G.
Task: Find a maximum matching.

Decision version of Maximum matching problem
Input: A graph G and a number k.
Question: Does G contain a matching of size at least k?

Decision version vs the optimization version

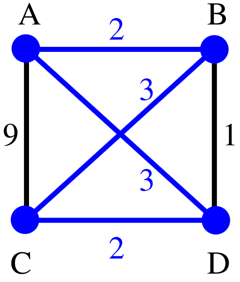
Solving the decision problem can not be harder than solving the optimization problem. For example if we can find a maximum matching in a graph, then we can easily decide whether its size is at least a given k.

optimization version		decision version
easy	$\Longrightarrow$	easy
solvable in polynomial time	$\Longrightarrow$	contained in P
hard	$\Longleftarrow$	hard

An algorithm designed for the decision version usually can be used to solve the optimization version as well by using binary-search.

The decision version of the TSP is in NP

Decision version of the TSP:
Input: A complete graph, a weight function w over its edge set and a number k.
Question: Is there a Hamiltonian cycle whose weight is at most k?



This problem is in NP, because if the answer is YES, then a proper witness is a proper transcription (what we used for HAMILTONIAN) of the Hamiltonian cycle whose weight is not more than k. The verification algorithm checks whether it is a Hamiltonian cycle and it sum the edge weights of the Hamiltonian cycle and compare the result with k.

Example: If the input is the graph above and $k = 11$, then the answer is yes and the witness is A,B,C,D which encodes the blue Hamiltonian cycle.

The decision version of an optimization problem is usually in NP

Remainder: An optimization problem is the following: A is the set of solutions, $f : \mathbf{A} \to \mathbb{R}$ is the objective function. Find an $x \in \mathbf{A}$ which minimizes (or maximizes) $f(x)$!

Decision version:
Input: A transcription of **A**, f and a number k.
Question: Is there a solution $x \in \mathbf{A}$ which satisfies that $f(x) \leq k$ ($f(x) \geq k$ in case of maximization)?

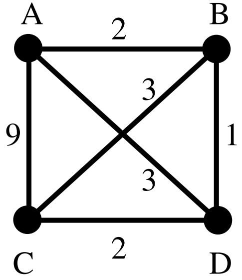
Claim
If the function $f(x)$ can be calculated in polynomial time for any $x \in \mathbf{A}$, the size of x is polynomial in the size of the input and $x \in \mathbf{A}$ can be verified in polynomial time, then the decision version of the optimization problem is in NP.

Proof: The withness is a solution x which satisfies that $f(x) \leq k$ ($f(x) \geq k$ in case of maximization). The verification algorithm checks that $x \in \mathbf{A}$, then it calculates $f(x)$ and compare it with k.

Not all decision problems are in NP

There are many decision problems for which we know (it is proved) that they are not contained in NP. Unfortunately they are complicated. Therefore we consider a problem which is probably not contained in NP, but nobody has proved it yet.

Complement problem of TSP
Input: A complete graph, a weight function w over its edge set and a number k.
Question: Is it true that there is no Hamiltonian cycle in the graph whose weight is at most k?



The conjecture that this problem is not in NP is widely believed by Computer Scientists and Mathematicians.

SAT is in NP

Problem SAT:
Input: A Boolean formula Φ
Question: Can we assign values 0 and 1 to the variables of Φ in such a way that Φ evaluates to 1?

Example:
$\Phi(x_1, x_2, x_3, x_4) = (x_1 \vee x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_4) \wedge (x_1 \vee x_3) \wedge \neg x_3$
$\Phi(1, 1, 0, 1) = 1$. So for this input the answer is YES.

1 and 0 are usually interpreted as TRUE and FALSE, respectively. We say that ϕ is satisfiable if there is an assignment which gives TRUE.

SAT is in NP, the witness is an assignment of the variables which makes Φ be True (1). The verification algorithm evaluates this assignment. It can be done in polynomial time.

Relation between P and NP

Claim

P$\subseteq$NP

Proof: If π is in P, then the witness W_I can be anything. We can verify the YES answer in polynomial time for input I simply by calculating the answer for I by the polynomial time algorithm which solves π. $\square$

There are many problems which are known to be contained in NP but we believe that they are not in P. However no one could prove that $P \neq NP$. The holy grail of the area is the following open question:

Open question:

Does P$\neq$NP?

There is a bounty of one million dollars on this problem. So if you solve it, you will be rich. Furthermore you receive a PhD in computer science immediately.

Using the same tool to handle different problems

In real life, if a tool works well in an area then we usually try to use it somewhere else. Sometimes we found pretty good applications elsewhere. For example Teflon (Polytetrafluoroethylene), the material which coats pans is first used to make gaskets.

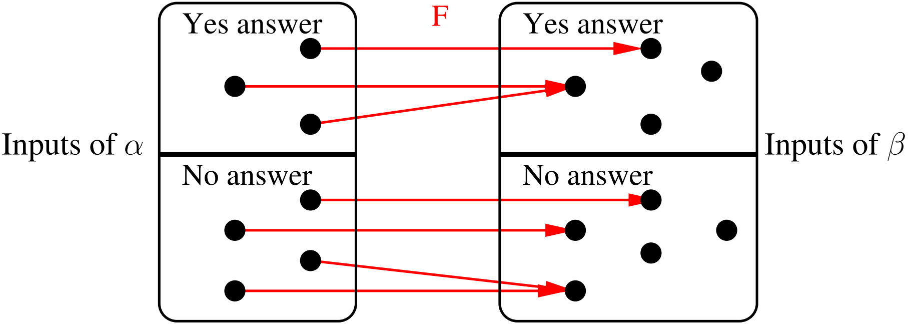
In the area of algorithms and optimization methods, we do the same. If an algorithm works well for a problem α, then we try to use it for problem β.

Polynomial-time reduction

A polynomial-time reduction (Karp reduction) is a method how to transform a (decision) problem to another one.

Definition: We say that the mapping F is a Karp reduction from decision problem α to decision problem β, if:

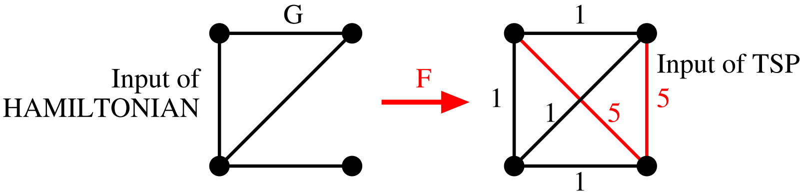
- F maps each input of α to an input of β.
- F can be calculated in polynomial time, so there is a k, such that $F(I)$ can be calculated in $O(|I|^k)$ for any input I.
- The answer for I is YES **if and only if** the answer for $F(I)$ is YES.



A Karp reduction from HAMILTONIAN to TSP

F maps an n vertex input graph G to K_n with a weight function $w : E(K_n) \to \mathbb{R}$ given below and set k to n.

$$w(e) = \begin{cases} 1 & \text{if } e \text{ is an edge of } G \\ n+1 & \text{if } e \text{ is not an edge of } G \end{cases}$$



Correctness of the reduction: If G has a hamiltonian cycle, then its weight in the created K_n is n, so the answer for the TSP is YES. If G does not have a hamiltonian cycle, then every hamiltonian cycle in the created graph contains an edge of weight $n+1$, therefore the answer for TSP is NO.
$F(G)$ can be calculated in $O(n^3)$ time, so in polynomial time.

Karp reductions and class P

Notation: If we have have a Karp reduction from α to β, then we write, that $\alpha \prec \beta$.

Claim

If $\alpha \prec \beta$ and $\beta \in$ P, then $\alpha \in$ P.

Proof: We have an algorithm B which solves β in polynomial time, so there is a number k_B such that any input I_B of β is solved in $O(|I_B|^{k_b})$ time.
Let F be the mapping of the Karp reduction. F can be calculated in polynomial time. Therefore there is a k_F such that for any input I_A of α $F(I_A)$ can be calculated in in $O(|I_A|^{k_F})$ time.
Hence $|F(I_A)| \in O(|I_A|^{k_F})$.
Let A be the following algorithm: It calculates $F(I_A)$,then it runs algorithm B. A solves problem α correctly.
The running time of A is $O(\left(|i_A|^{k_F}\right)^{k_B}) = O(|i_A|^{k_F k_B})$, hence it runs in polynomial time. $\square$