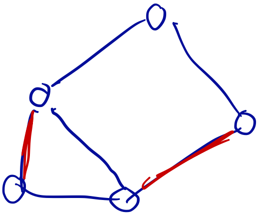
Application: Kőnig's Theorem

Definition 7.10.

Consider an undirected graph $G = (V, E)$.

- A matching in G is a subset of edges $M \subseteq E$ with $e \cap e' = \emptyset$ for all $e, e' \in M$ with $e \neq e'$.



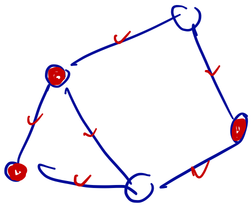
Application: Kőnig's Theorem

Definition 7.10.

Consider an undirected graph $G = (V, E)$.

i A matching in G is a subset of edges $M \subseteq E$ with $e \cap e' = \emptyset$ for all $e, e' \in M$ with $e \neq e'$.

ii A vertex cover is a subset of nodes $C \subseteq V$ with $e \cap C \neq \emptyset$ for all $e \in E$.



Application: Kőnig's Theorem

Definition 7.10.

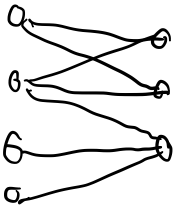
Consider an undirected graph $G = (V, E)$.

i. A matching in G is a subset of edges $M \subseteq E$ with $e \cap e' = \emptyset$ for all $e, e' \in M$ with $e \neq e'$.
ii. A vertex cover is a subset of nodes $C \subseteq V$ with $e \cap C \neq \emptyset$ for all $e \in E$.

Theorem 7.11.

In bipartite graphs, the maximum cardinality of a matching equals the minimum cardinality of a vertex cover.

ILP formulation of Max-cardinality matching



$$\max \sum_{e \in E} x_e$$

$$\text{s.t.} \sum_{w \in \delta(v)} x_{vw} \leq 1 \qquad \forall v \in V$$

$$x_e \in \{0,1\} \qquad \forall e \in E$$

Application: Kőnig's Theorem

Definition 7.10.

Consider an undirected graph $G = (V, E)$.

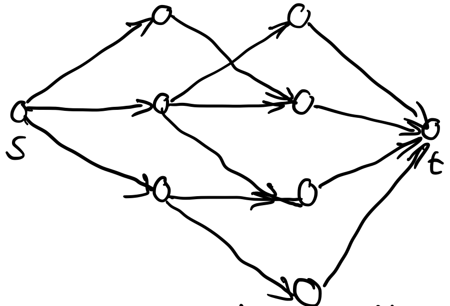
1. A matching in G is a subset of edges $M \subseteq E$ with $e \cap e' = \emptyset$ for all $e, e' \in M$ with $e \neq e'$.
2. A vertex cover is a subset of nodes $C \subseteq V$ with $e \cap C \neq \emptyset$ for all $e \in E$.

Theorem 7.11.

In bipartite graphs, the maximum cardinality of a matching equals the minimum cardinality of a vertex cover.

Observation: In a bipartite graph $G = (P \dot\cup Q, E)$, a maximum cardinality matching can be found by a maximum flow computation.

Computing a max-cardinality matching can be done by max-flow methods (for bipartite graphs)



$u_a = 1 \quad \forall a \in A$

↳ after running Ford-Fulkerson:

- flow carrying arc define a matching
- max flow is integer, so all values are 0/1 per arc
- conversely, every matching defines an s-t-flow with values 0/1

COMP331/557

Chapter 8:
Complexity Theory

(Cook, Cunningham, Pulleyblank & Schrijver, Chapter 9;
Korte & Vygen, Chapter 15
Garey & Johnson)

Efficient Algorithms: Historical Remark

Edmonds (1965):

I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

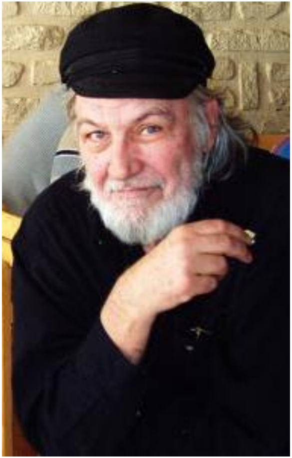
There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

The mathematical significance of this paper rests

Edmonds (1967):

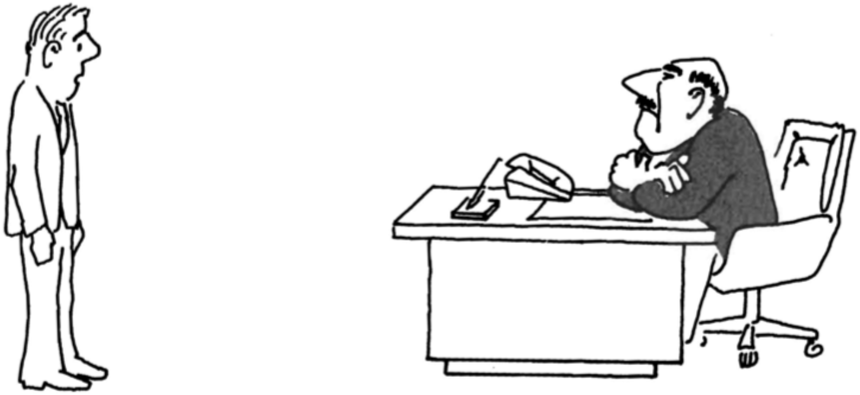
We say an algorithm is *good* if there is a polynomial function $f(n)$ which, for every positive-integer valued n, is an upper bound on the "amount of work" the algorithm does for any input of "size" n. The concept

traveling saleman problem [cf. 4]. I conjecture that there is no good algorithm for the traveling saleman problem. My reasons are the same as for any mathematical conjecture: (1) It is a legitimate mathematical possibility, and (2) I do not know.



Jack Edmonds (1934–)

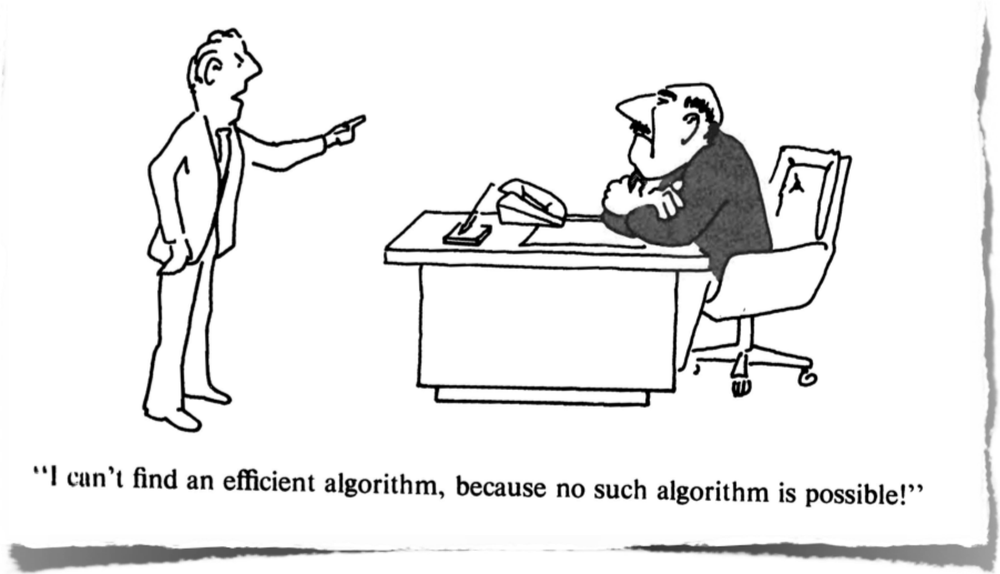
Is There a Good Algorithm for the TSP?



"I can't find an efficient algorithm, I guess I'm just too dumb."

Source: Garey & Johnson, Computers and Intractability, 1979.

Is There a Good Algorithm for the TSP?



Source: Garey & Johnson, Computers and Intractability, 1979.

Is There a Good Algorithm for the TSP?

"I can't find an efficient algorithm, but neither can all these famous people."

Source: Garey & Johnson, Computers and Intractability, 1979.

Decision Problems

Most of complexity theory is based on decision problems such as, e. g.:

(Undirected) Hamiltonian Circuit Problem

Given: undirected graph $G = (V, E)$.

Task: decide whether G contains a Hamiltonian circuit.

↳ circuit visiting each node exactly once

Decision Problems

Most of complexity theory is based on decision problems such as, e. g.:

(Undirected) Hamiltonian Circuit Problem

Given: undirected graph $G = (V, E)$.

Task: decide whether G contains a Hamiltonian circuit.

Definition 8.1.

i. A decision problem is a pair $\mathcal{P} = (X, Y)$. The elements of X are called instances of $\mathcal{P}$, the elements of $Y \subseteq X$ are the yes-instances, those of $X \setminus Y$ are no-instances.
ii. An algorithm for a decision problem (X, Y) decides for a given $x \in X$ whether $x \in Y$.

Decision Problems

Most of complexity theory is based on decision problems such as, e. g.:

(Undirected) Hamiltonian Circuit Problem

Given: undirected graph $G = (V, E)$.

Task: decide whether G contains a Hamiltonian circuit.

Definition 8.1.

i. A decision problem is a pair $\mathcal{P} = (X, Y)$. The elements of X are called instances of $\mathcal{P}$, the elements of $Y \subseteq X$ are the yes-instances, those of $X \setminus Y$ are no-instances.
ii. An algorithm for a decision problem (X, Y) decides for a given $x \in X$ whether $x \in Y$.

Example. For Hamiltonian Circuit, X is the set of all (undirected) graphs and $Y \subset X$ is the subset of graphs containing a Hamiltonian circuit.

Further Examples of Decision Problems

(Integer) Linear Programming Problem (decision version)

Given: matrix $A \in \mathbb{Z}^{m \times n}$, vector $b \in \mathbb{Z}^m$.

Task: decide whether there is $x \in \mathbb{R}^n$ ($x \in \mathbb{Z}^n$) with $A \cdot x \geq b$.

Further Examples of Decision Problems

(Integer) Linear Programming Problem (decision version)

Given: matrix $A \in \mathbb{Z}^{m \times n}$, vector $b \in \mathbb{Z}^m$.

Task: decide whether there is $x \in \mathbb{R}^n$ ($x \in \mathbb{Z}^n$) with $A \cdot x \geq b$.

Spanning Tree Problem

Given: graph $G = (V, E)$, edge weights $w : E \to \mathbb{Z}$, positive integer k.

Task: decide whether there is a spanning subtree of weight at most k.

Further Examples of Decision Problems

(Integer) Linear Programming Problem (decision version)

Given: matrix $A \in \mathbb{Z}^{m \times n}$, vector $b \in \mathbb{Z}^m$.

Task: decide whether there is $x \in \mathbb{R}^n$ ($x \in \mathbb{Z}^n$) with $A \cdot x \geq b$.

Spanning Tree Problem

Given: graph $G = (V, E)$, edge weights $w : E \to \mathbb{Z}$, positive integer k.

Task: decide whether there is a spanning subtree of weight at most k.

Steiner Tree Problem

Given: graph $G = (V, E)$, terminals $T \subseteq V$, edge weights $w : E \to \mathbb{Z}$, positive integer k.

Task: decide whether there is a subtree of G of weight at most k that contains all terminals in T.

Complexity Classes P and NP

P

The class of all decision problems for which there is a deterministic polynomial time algorithm is denoted by P.

Complexity Classes P and NP

P

The class of all decision problems for which there is a deterministic polynomial time algorithm is denoted by P.

Example: The Spanning Tree Problem is in P.

Complexity Classes P and NP

P

The class of all decision problems for which there is a deterministic polynomial time algorithm is denoted by P.

Example: The Spanning Tree Problem is in P.

NP

A decision problem belongs to the complexity class NP if solutions can be verified in polynomial time.

Complexity Classes P and NP

P

The class of all decision problems for which there is a deterministic polynomial time algorithm is denoted by P.

Example: The Spanning Tree Problem is in P.

NP

A decision problem belongs to the complexity class NP if solutions can be verified in polynomial time.

Examples: The Hamiltonian Circuit Problem and all problems listed on the previous slides belong to NP.

Nondeterministic Turing Machines

Remarks.

- The complexity class P consists of all decision problems that can be solved by a deterministic Turing machine in polynomial time.