

A brief introduction to NP-completeness – following Richard M. Karp

MRain

November 3, 2010

What will we talk about?

- How does it come about? – Combinatorial problem
- What's NP-completeness? – Complexity theory
- How to cope with it? – maybe Randomized algorithm

A breif biography of Karp

- In 1959, got his Ph.D. in applied mathematics if Harvard University
- Started working at IBM's Thomas J. Watson Research Center.
- In 1968, he became Professor of Computer Science, Mathematics, and Operations Research at the UC Berkeley.
- More information in [Wikipedia](#)



Combinatorial problem

Definition: Problems that can be likened to jigsaw puzzles where one has to assemble the parts of a structure in a particular way.

Example:

- n-Queens
- Automatic synthesis of switching circuits
- Traveling Salesman Problem
- Marriage Problem

Marriage Problem

This problem concerns a society consisting of n men and n women.

The problem is to pair up the men and women in a one-to-one fashion at minimum cost. These costs are given by an $n \times n$ matrix, in which each row corresponds to one of the men and each column to one of the women.

$$\begin{pmatrix} 3 & 4 & 2 \\ 8 & 9 & 1 \\ 7 & 9 & 5 \end{pmatrix}$$

Figure: An example

Marriage Problem

How to solve the Marriage Problem?

The number of possible pairings is $n!$, a function that grows so rapidly that brute-force enumeration will be of little avail.

Marriage Problem

How to solve the Marriage Problem?

The number of possible pairings is $n!$, a function that grows so rapidly that brute-force enumeration will be of little avail.

Hungarian Algorithm

It's a beautiful algorithm. The time required for it to solve this problem grows only as the third power of n .

Hungarian Algorithm

The key observation underlying the Hungarian algorithm is that the problem remains unchanged if the same constant is subtracted from all the entries in one particular row of the matrix. Using this freedom to alter the matrix, the algorithm tries to create a matrix in which all the entries are nonnegative, so that every complete pairing has a nonnegative total cost, and in which there exists a complete pairing whose entries are all zero.

$$\begin{pmatrix} 3 & 4 & 2 \\ 8 & 9 & 1 \\ 7 & 9 & 5 \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 \\ 7 & 8 & 0 \\ 2 & 4 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 6 & 6 & 0 \\ 1 & 2 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 5 & 5 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Figure: An instance of the Marriage Problem

Traveling Salesman Problem

Traveling Salesman Problem

Task: Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.

Solutions:

- Brute-force: at most 12 cities.
- Dynamic programming: no more than 16 cities.
- **Integer linear program** maybe good but ...



Figure: An optimal TSP tour through Germany's 15 cities.

A big problem

Whatever algorithms above we use we can not avoid a **combinatorial explosion**.

A **combinatorial explosion** is the vast, furiously growing number of possibilities that have to be searched through.

Now we have a big problem:

A big problem

Whatever algorithms above we use we can not avoid a **combinatorial explosion**.

A **combinatorial explosion** is the vast, furiously growing number of possibilities that have to be searched through.

Now we have a big problem:

Is there a beautiful algorithm which can solve this problem avoiding combinatorial explosion?

How do we define a algorithm is “beautiful”?

How do we define a algorithm is “beautiful”?

During the 1960s, a second major stream of research was gathering force – computational complexity theory.

In 1965, **Juris Hartmanis** and **Richard E. Stearns** published their lecture *On the computational complexity of algorithm*. It introduces framework for computational complexity using abstract machines.

Jack R. Edmonds defines “good” algorithm as one with running time bounded by polynomial function the size of input.

Spring in Berkeley

In 1968, Karp decided to move to the University of California at Berkeley.

There he worked with many outstanding scientists:

In 1968, Karp decided to move to the University of California at Berkeley.

There he worked with many outstanding scientists:

- **Stephen Arthur Cook**, whose work in complexity theory was to influence Karp so greatly.

In 1968, Karp decided to move to the University of California at Berkeley.

There he worked with many outstanding scientists:

- **Stephen Arthur Cook**, whose work in complexity theory was to influence Karp so greatly.
- **Julia Hall Bowman Robinson**, whose work on **Hilbert's tenth problem** was soon to bear fruit.

In 1968, Karp decided to move to the University of California at Berkeley.

There he worked with many outstanding scientists:

- **Stephen Arthur Cook**, whose work in complexity theory was to influence Karp so greatly.
- **Julia Hall Bowman Robinson**, whose work on **Hilbert's tenth problem** was soon to bear fruit.
- **Steve Smale**, whose ground-breaking work on the probabilistic analysis of linear programming algorithm was to influence Karp.

In 1968, Karp decided to move to the University of California at Berkeley.

There he worked with many outstanding scientists:

- **Stephen Arthur Cook**, whose work in complexity theory was to influence Karp so greatly.
- **Julia Hall Bowman Robinson**, whose work on **Hilbert's tenth problem** was soon to bear fruit.
- **Steve Smale**, whose ground-breaking work on the probabilistic analysis of linear programming algorithm was to influence Karp.
- And so on ...

Cook's historic paper

In 1971 Cook, published his historic paper “The Complexity of Theorem Proving Procedures”.

He discussed the classes of problems that we now call P and NP, and introduced the concept that we now refer to NP-completeness.

Cook's historic paper

In 1971 Cook, published his historic paper “The Complexity of Theorem Proving Procedures”.

He discussed the classes of problems that we now call P and NP, and introduced the concept that we now refer to NP-completeness.

Class P: Those problems that can be solved in polynomial time.

Class NP: Those problems whose proposed solution can be checked in polynomial time.

P versus NP Problem

What will happen if P equals to NP?

You can see something maybe exaggerated at [here](#).

P versus NP Problem

What will happen if P equals to NP?

You can see something maybe exaggerated at [here](#).

How to prove that $P=NP$?

P versus NP Problem

What will happen if P equals to NP ?

You can see something maybe exaggerated at [here](#).

How to prove that $P=NP$?

The most important achievement of Cook's paper was to show that $P = NP$ if and only if a particular computational problem called the [satisfiability problem](#) lies in P .

Satisfiability Problem

The **Satisfiability Problem** (SAT for short) comes from mathematical logic and has applications in switching theory, it can be stated as a simple combinatorial puzzle:
Given several sequences of upper- and lowercase letters, is it possible to select a letter from each sequence without selecting both the upper- and lower case versions of any letter?

Based on concept of reducibility, any instance of a problem in NP can be transformed into a corresponding instance of the SAT.

Reducibility

This is just a short explanation.

Problem A is said to be reducible to problem B if, given a subroutine capable of solving problem B, one can construct an algorithm to solve problem A.

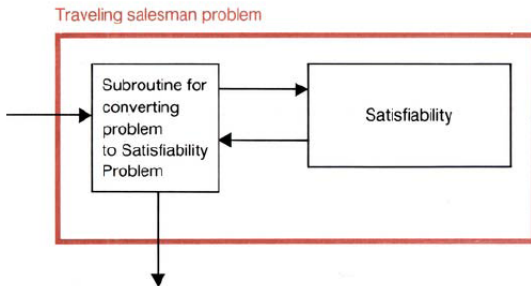


Figure: TSP is reducible to SAT

Why does there always exist some problem which is inherently intractable like TSP or SAT?

NP-completeness

Why does there always exist some problem which is inherently intractable like TSP or SAT?

NP-completeness(NPC for short)

The concept of NP-complete was first introduced in 1971 by Stephen Cook in his historical paper, though the term NP-complete did not appear anywhere in his paper.

NP-completeness

Why does there always exist some problem which is inherently intractable like TSP or SAT?

NP-completeness(NPC for short)

The concept of NP-complete was first introduced in 1971 by Stephen Cook in his historical paper, though the term NP-complete did not appear anywhere in his paper.

Definition: A problem is NP-complete if it lies in the class NP, and every problem in NP is polynomial-time reducible to it.

Influenced by Cook's theory, in 1972, Karp published a **landmark** paper *Reducibility among Combinatorial Problems* in which he proved **21 Problems to be NP-complete**.






His results were quickly refined and extended by other workers, and in next few years, hundreds of different problems, arising in virtually every field where computation is done, were shown to be NP-complete.

Note that if any one of the NPC problems was solved in polynomial time, $P = NP$.

How to cope with NPC problems?

- Polynomial time approximation algorithms
- Use some algorithms worked well in practice like **Simplex Algorithm**
- Randomized algorithms and probabilistic analysis

References

-  Richard M. Karp, *Combinatorics, Complexity, and Randomness*
-  Richard M. Karp, *Reducibility Among Combinatorial Problems*
-  Stephen A. Cook, *An overview of computational complexity*
-  Stephen A. Cook, *The Complexity of Theorem Proving Procedures*
-  Juris Hartmanis & Richard E. Stearns, *On the computational complexity of algorithm*