Dependable Real-Time Systems

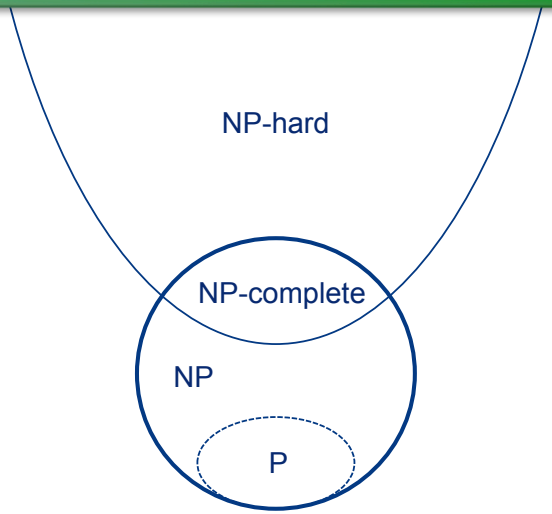
Lecture #4

Professor Jan Jonsson

Department of Computer Science and Engineering
Chalmers University of Technology

NP-hard problems

NP-hard problems:
Problems that are “at least as hard” as the hardest problems in class NP.



Assuming P ≠ NP

NP-hard problems

Turing reducibility:

- A problem Π' is Turing reducible to problem Π if there exists an algorithm A that solves Π' by using a hypothetical subroutine S for solving Π such that, if S were a polynomial time algorithm for Π, then A would be a polynomial time algorithm for Π' as well.

  When Π' is Turing reducible to Π, we write $\Pi' \propto_T \Pi$

A search problem Π is said to be NP-hard if there exists some NP-complete problem Π' that Turing-reduces to Π.

NP-hard problems

Observations:

- All NP-complete problems are NP-hard
- All co-NP-complete problems are NP-hard

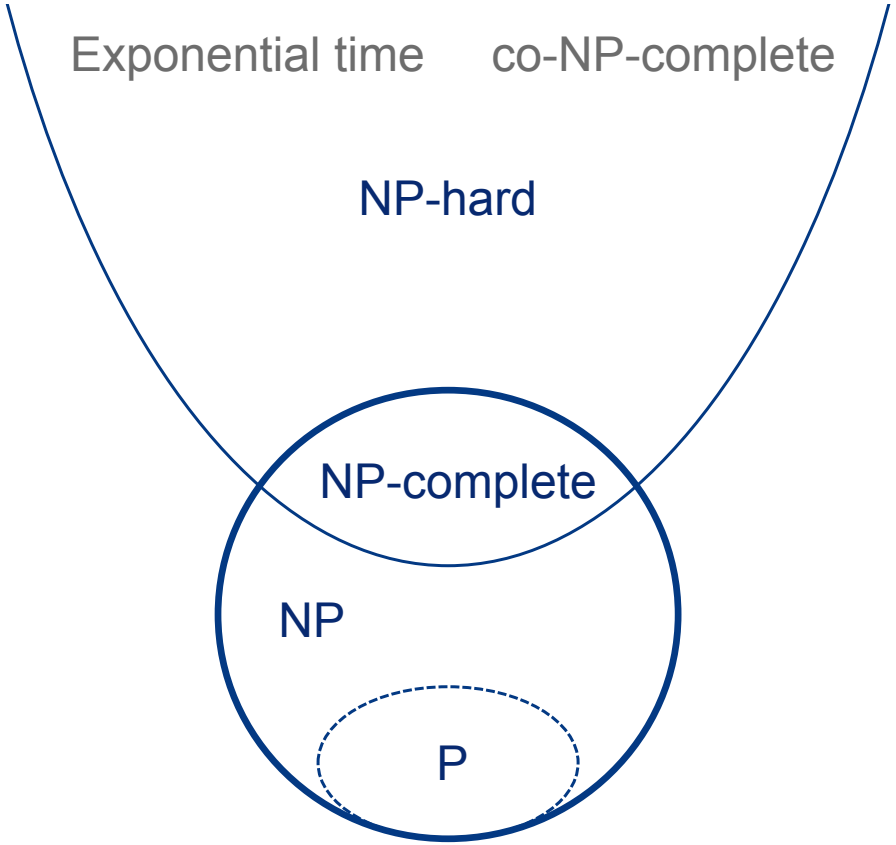
  Turing reduction from Π to Π^C (and vice versa) is trivial.

- Given an NP-complete decision problem, the corresponding optimization problem is NP-hard

  To see this, imagine that the optimization problem (that is, finding the optimal cost) could be solved in polynomial time.

  The corresponding decision problem (i.e., determining whether there exists a solution with a cost no more than B) could then be solved by simply comparing the found optimal cost to the bound B. This comparison is a constant-time operation.

NP-hard problems



Assuming P ≠ NP

NP-hard problems

Observations for the Traveling Salesman Problem:

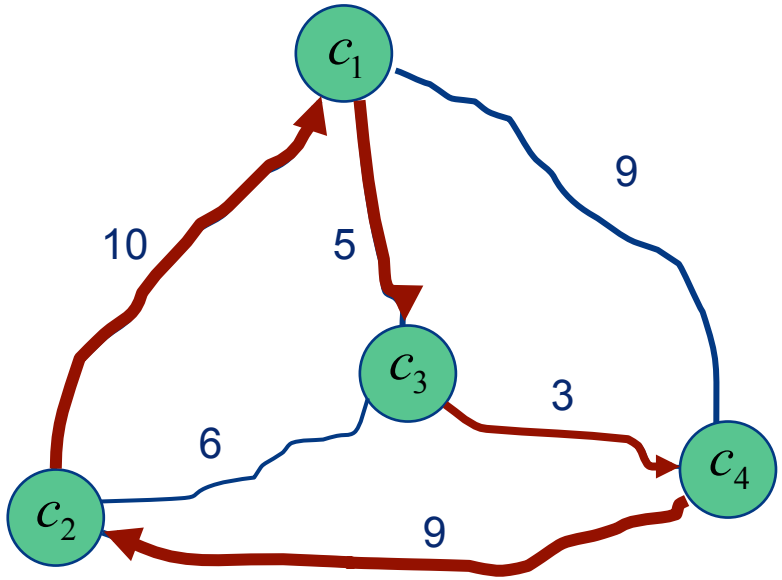
- The original Traveling Salesman Problem is NP-hard
- The complement Traveling Salesman Problem is NP-hard
- The Traveling Salesman Optimization Problem is NP-hard

  To see this, imagine that the shortest tour could be found in polynomial time.

  The original Traveling Salesman problem (i.e., does there exist a tour with total length no more than B) could then be solved by simply comparing the found shortest tour to the bound B. This comparison is a constant-time operation.

NP-hard problems

The Traveling Salesman Optimization Problem:



Minimum “tour” length = 27

Minimize the length of the “tour” that visits each city in sequence, and then returns to the first city.

NP-hard problems

Observations relating to scheduling problems:

- NP-complete scheduling problems are NP-hard
- Co-NP-complete scheduling problems are NP-hard
- Exponential-time decision problems are NP-hard

  Example: hyper-period analysis, which is known to have an exponential time complexity in the general case.

- Metric-optimizing scheduling algorithms are NP-hard

  Example: searching for a schedule that has minimum task lateness (response time minus deadline), which requires an exponential time branch-and-bound algorithm in the general case.

NP-hard problems

Observations relating to scheduling problems:

- NP-complete scheduling problems are NP-hard
- Co-NP-complete scheduling problems are NP-hard
- Exponential-time decision problems are NP-hard

  Example: hyper-period analysis, which is known to have an exponential time complexity in the general case.

- Metric-optimizing scheduling algorithms are NP-hard

  To see this, imagine that a schedule with minimum task lateness could be found in polynomial time.

  Schedulability of a task set could then be determined by simply checking whether or not the found minimum task lateness is less than or equal to 0.

History of NP-completeness

S. Cook: (1971)

"The Complexity of Theorem Proving Procedures"

Every problem in the class NP of decision problems polynomially reduces to the SATISFIABILITY problem:

Given a set U of Boolean variables and a collection C of clauses over U, is there a satisfying truth assignment for C ?

R. Karp: (1972)

"Reducibility among Combinatorial Problems"

Decision problem versions of many well-known combinatorial optimization problems are "just as hard" as SATISFIABILITY.

History of NP-completeness

D. Knuth: (1974)

"A Terminological Proposal"

Initiated a researcher's poll in search of a better term for "at least as hard as the polynomial complete problems".

One suggestion by S. Lin was PET problems:

- "Probably Exponential Time" (if P = NP remain open question)
- "Provably Exponential Time" (if P $\neq$ NP)
- "Previously Exponential Time" (if P = NP)

Some original NP-complete problems

Proving NP-completeness

Proving NP-completeness for a decision problem Π:

1. Show that Π is in NP
2. Select a known NP-complete problem Π'
3. Construct a transformation $\propto$ from Π' to Π
4. Prove that $\propto$ is a (polynomial) transformation

The book "Computers and Intractability – A Guide to the Theory of NP-Completeness" (Garey and Johnson, 1979) contains a categorized list of 300+ NP-complete problems, with problem statements and how each problem was proven NP-complete.

Proving NP-completeness

Transformations for real-time scheduling problems:

In published results regarding the time complexity of known real-time scheduling problems, the following NP-complete problems are predominantly used for the transformations:

- 3-PARTITION
  - NP-complete in the strong sense.
  - Used in the proofs for multiprocessor scheduling, and in the proof for non-preemptive uniprocessor scheduling.
- SIMULTANEOUS CONGRUENCES (SCP)
  - NP-complete in the strong sense.
  - Used in the proofs for preemptive uniprocessor scheduling, by employing a reverse logic (co-NP) strategy.

Proving NP-completeness

3-PARTITION decision problem:

- Set of elements
  - Let $A = \{a_1, \ldots, a_{3m}\}$ be a set of $3m$ elements.
  - Each element $a_i \in A$ has a positive integer "size" $s(a_i)$.
- Element size constraints using a bound
  - Let B be a positive integer.
  - Each $s(a_i)$ satisfies $B/4 < s(a_i) < B/2$ and $\sum_{a_i \in A} s(a_i) = mB$.
- Question:
  - Can A be partitioned into m disjoint sets $S_1, \ldots, S_m$ such that, for each $1 \leq j \leq m$, it applies that $\sum_{a_i \in S_j} s(a_i) = B$?

Note: constraints dictate that each disjoint set must contain exactly 3 elements!

Proving NP-completeness

SIMULTANEOUS CONGRUENCES decision problem:

- Set of ordered pairs
  - Let $A = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ be a set of n ordered pairs.
  - Each pair $(a_i, b_i) \in A$ consists of positive integers.
- Minimum bound
  - Let B be a positive integer, such that $2 \leq B \leq n$.
- Question:
  - Does there exist a subset $A' \in A$ of at least B pairs and a positive integer x such that, for each $(a_i, b_i) \in A'$, it applies that $x \equiv a_i \pmod{b_i}$?

Note: $x \equiv a_i \pmod{b_i}$ means $x = a_i + k_i \cdot b_i$ for some non-negative integer k_i

Complexity in real-time scheduling

General complexity results:

- Any type of scheduling of periodic tasks
  - NP-hard (exponential time)
- Any type of non-preemptive scheduling
  - NP-complete in the strong sense (reduction from 3-PARTITION)
- Any type of preemptive multiprocessor scheduling
  - NP-complete in the strong sense (reduction from 3-PARTITION)
  - Note: applies to both partitioned and global approaches
- Preemptive uniprocessor scheduling of asynchronous tasks
  - Co-NP-complete in the strong sense (reduction from SCP)

Complexity in real-time scheduling

Preemptive uniprocessor scheduling:

- Scheduling of synchronous tasks w/ dynamic task priorities
  - Co-NP-complete in the strong sense (reduction from SCP)
  - Co-NP-complete in the weak sense for $U < 1$
  - Special cases:
    - Pseudo-polynomial time for constrained-deadline tasks for $U < 1$
    - Polynomial time for implicit-deadline tasks
- Scheduling of synchronous tasks w/ static task priorities
  - NP-hard for arbitrary-deadline tasks (exponential time)
  - NP-complete in the weak sense for constrained-deadline tasks
  - Special cases:
    - Pseudo-polynomial time for constrained-deadline tasks
    - Polynomial time for implicit-deadline tasks for $U \leq \ln 2$

SATISFIABILITY

The original NP-complete decision problem:

- Variables and literals
  - Let U be a set of Boolean variables.
  - If u is a variable in U then u and u' are literals over U.
- Conjunctive normal form
  - A formula is in conjunctive normal form (CNF) if it is a conjunction of one or more clauses, where a clause is a disjunction of literals.
- SATISFIABILITY question:
  - Given a formula in CNF does there exist a truth assignment for the variables in U that yields a True statement?

SATISFIABILITY

Some variations of the conjunctive viewpoint:

- FALSIFIABILITY decision problem
  - Given a formula in CNF does there exist a truth assignment for the variables in U that yields a False statement?
- CONTRADICTION decision problem
  - Given a formula in CNF does every possible truth assignment for the variables in U yield a False statement?
- TAUTOLOGY decision problem
  - Given a formula in CNF does every possible truth assignment for the variables in U yield a True statement?

What are the time complexities of these variations?

SATISFIABILITY

SATISFIABILITY from a disjunctive viewpoint:

- Variables and literals
  - Let U be a set of Boolean variables.
  - If u is a variable in U then u and u' are literals over U.
- Disjunctive normal form
  - A formula is in disjunctive normal form (DNF) if it is a disjunction of one or more clauses, where a clause is a conjunction of literals.
- (DNF) SATISFIABILITY question:
  - Given a formula in DNF does there exist a truth assignment for the variables in U that yields a True statement?

SATISFIABILITY

Some variations of the disjunctive viewpoint:

- (DNF) FALSIFIABILITY decision problem
  - Given a formula in DNF does there exist a truth assignment for the variables in U that yields a False statement?
- (DNF) CONTRADICTION decision problem
  - Given a formula in DNF does every possible truth assignment for the variables in U yield a False statement?
- (DNF) TAUTOLOGY decision problem
  - Given a formula in DNF does every possible truth assignment for the variables in U yield a True statement?

What are the time complexities of the DNF variations?