ECE 641
Advanced Topics in Supervisory Control for Discrete Event Systems

Lecture 5

Associate Prof. Dr. Klaus Schmidt

Department of Mechatronics Engineering – Çankaya University

PhD Course in Electronic and Communication Engineering
Credits (3/0/3)
Course webpage: http://ece641.cankaya.edu.tr/

Computational Complexity: Properties

Properties

- Big Theta is equivalent to Big O and Big Omega at the same time: $f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$
- Transitivity: $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$
  $\Rightarrow$ Same holds for O and Ω
- Additivity: $f(n) = \Theta(h(n))$ and $g(n) = \Theta(h(n)) \Rightarrow$ $f(n) + g(n) = \Theta(h(n))$
  $\Rightarrow$ Same holds for O and Ω
- Reflexivity: $f(n) = \Theta(f(n))$
  $\Rightarrow$ Same holds for O and Ω
- Symmetry: $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$
- Transpose Symmetry: $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$

Computational Complexity: Properties

Computations

Gap 1

Decidability: Solution of an Algorithm

Guaranteed Solution Approach

- Brute force: For many non-trivial problems, there is a natural brute force search algorithm that checks every possible solution.
  $\Rightarrow$ Enumerate all candidate solutions and check if the solution properties are fulfilled
- Typically the complexity of such approach is $O(2^n)$ or worse for inputs of size n
  $\Rightarrow$ Unacceptable in practice

Desirable Scaling Property

- There exists constants $K > 0$ and $d > 0$ such that run-time time is bounded by $K \cdot n^d$ steps for every input of size n
  $\Rightarrow$ Polynomial time $O(n^d)$

Decidability: Algorithmic and Problem Complexity

Algorithmic Complexity

- Measure of how difficult it is to perform the algorithmic computation
  $\Rightarrow$ Algorithmic complexity is specific to an algorithm

Problem Complexity

- Complexity of the algorithm with the lowest order of growth of complexity for solving a given problem
  $\Rightarrow$ Specific to the problem

Decision Problem

- Formulation of a problems as question with Yes/No answer

$\Rightarrow$ We want to classify decision problems according to their difficulty

Decidability: Decision Problems

Diagnosability

Gap 2

Boolean Satisfiability

- Literal: boolean variable or its negation: x_i or $\overline{x}_i$
- Clause: Disjunction of literals: $C_j = x_1 \vee \overline{x}_2 \vee x_3$
- Conjunctive normal form (CNF): Boolean formula that is conjunction of clauses: $B = C_1 \wedge C_2 \wedge \cdots \wedge C_n$
- Decision Problem CNF-SAT: Given B in conjunctive normal form, is there an assignment of the variables x_i such that $B = true$?

Decidability: Decision Problems

3-CNF SAT

- CNF-SAT, where each clause has 3 distinct literals.

Gap 3

Directed Hamilton Cycle

- Given a directed graph $G = (V, E)$, does there exist a simple directed cycle C that contains every vertex in V

Gap 4

Klaus Schmidt
Department of Electronic and Communication Engineering – Çankaya University

Decidability: Class P

Definition

- Class P consists of (decision) problems that are solvable in polynomial time
  $\Rightarrow$ There is a polynomial-time algorithm with complexity $O(n^d)$ for some $d \geq 0$ to solve the problem

Remarks

- Problems in P are also called tractable
- Problems not in P are called intractable or unsolvable
  $\Rightarrow$ Such problems can be solved in reasonable time only for small inputs or cannot be solved at all

Klaus Schmidt
Department of Electronic and Communication Engineering – Çankaya University

Decidability: Class P

Examples

Gap 5

Decidability: Class NP

Nondeterministic algorithm

- Two stage procedure
- 1. Nondeterministic (guessing) stage: Randomly generate a "candidate" solution
- 2. Deterministic (verification) stage: Take the candidate to the problem and returns YES if the candidate represents a solution

NP algorithms (Nondeterministic polynomial)

- Nondeterministic algorithm with a polynomial-time verification stage

Class NP

- Class of problems that could be solved by NP algorithms
  $\Rightarrow$ If we are given a candidate, we could verify that the candidate is correct in polynomial time

Warning: NP DOES NOT mean non-polynomial

Decidability: Comparison

$P \subseteq NP$

Gap 6

Remarks

- Big open question in Computer Science: P = NP? (1000000$ prize)
- Most computer scientists believe that this is false but there is no proof

Klaus Schmidt
Department of Electronic and Communication Engineering – Çankaya University

Decidability: Polynomial Reduction

Definition

- Given two problems A, B, we say that A is polynomially reducible to B ($A \leq_P B$) if:
  - There exists a function f that converts the input of A to inputs of B in polynomial time
  - A(s) = YES if and only if B(f(s)) = YES for any canditate s

Illustration

Gap 7

Klaus Schmidt
Department of Electronic and Communication Engineering – Çankaya University

Decidability: Polynomial Reduction

Example

Gap 8

Klaus Schmidt
Department of Electronic and Communication Engineering – Çankaya University

Decidability: NP-complete Problems

Definition

- A problem B is NP-complete if:
  1. $B \in NP$
  2. $A \leq_P B$ for all $A \in NP$
- If B satisfies only property (2) we say that B is **NP-hard**

Remarks

- NP-complete problems are defined as the most difficult problems in NP
- Most practical problems turn out to be either P or NP-complete.
- No polynomial time algorithm has been discovered for an NP-Complete problem
- No one has ever proven that no polynomial time algorithm can exist for any NP-Complete problem

Klaus Schmidt
Department of Electronic and Communication Engineering – Çankaya University

Decidability: Relation between Classes

Illustration

Gap 9

Decidability: NP-complete Problems

Standard NP-complete Problems

- Boolean Satisfiability
- Hamilton Path Problem
- Vertex Cover
- etc.

Examples for NP-hard Problems

- Modular supervisory control problem (SUPMM)
- m plant automata G_i
- n specification automata H_j
  $\Rightarrow$ Show that this problem is NP-hard

Gohari, P. and Wonham, W. M.: On the Complexity of Supervisory Control Design in the RW Framework, IEEE Trans. on Syst., Man, and Cyb.–Part B, 30(5), 2000.

Decidability: NP-hard Problems

Examples

Gap 10

Computational Complexity Decidability

Decidability: NP-hard Problems

Examples

Gap 11

Klaus Schmidt
Department of Electronic and Communication Engineering – Çankaya University

Decidability: NP-hard Problems

Examples

Gap 12

Decidability: Further Remarks

Remarks

- If a problem is proved to be NP-Complete, a good evidence for its intractability (hardness)
  ⇒ Not waste time on trying to find efficient algorithm for it
- Instead, focus on design approximate algorithm or a solution for a special case of the problem

Discussion

- NP-complete: means problems that are 'complete' in NP, i.e. the most difficult to solve in NP
- NP-hard - stands for 'at least' as hard as NP (but not necessarily in NP);
- NP-easy - stands for 'at most' as hard as NP (but not necessarily in NP);