Lecture 23: Intractability III

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

November 30, 2023
Overview

- Intractability
  - Scheduling Problems
  - Algebraic Problems
  - Mathematical Programming Problems
  - Taxonomy of Hard Problems

- Further Explorations
  - Computational view of the world
  - Courses
  - Research
  - AMA

- Acknowledgements
A job will be a tuple \((r, d, t)\) where
- \(r\) is the release time
- \(d\) is the deadline by which the job must be completed
- \(t\) is the duration it takes to complete the job, once you started it
Scheduling with Release Times and Deadlines

- A job will be a tuple \((r, d, t)\) where
  - \(r\) is the release time
  - \(d\) is the deadline by which the job must be completed
  - \(t\) is the duration it takes to complete the job, once you started it

- Allocate jobs to one machine. Our machine
  - can only run one job at a time,
  - once started on job, must finish it before taking on another job.
A job will be a tuple \((r, d, t)\) where
- \(r\) is the release time
- \(d\) is the deadline by which the job must be completed
- \(t\) is the duration it takes to complete the job, once you started it

Allocate jobs to one machine. Our machine
- can only run one job at a time,
- once started on job, must finish it before taking on another job.

**Input:** Set of jobs \(\{(r_i, d_i, t_i)\}_{i=1}^{n} \subset \mathbb{N}^3\)

**Output:** can we schedule all jobs so that each is completed by its deadline?
Scheduling with Release Times and Deadlines

- A job will be a tuple \((r, d, t)\) where
  - \(r\) is the release time
  - \(d\) is the deadline by which the job must be completed
  - \(t\) is the duration it takes to complete the job, once you started it

- Allocate jobs to one machine. Our machine
  - can only run one job at a time,
  - once started on job, must finish it before taking on another job.

- **Input:** Set of jobs \(\{ (r_i, d_i, t_i) \}_{i=1}^{n} \subset \mathbb{N}^3\)

- **Output:** can we schedule all jobs so that each is completed by its deadline?

- We will show this problem is \(NP\)-complete
A job will be a tuple \((r, d, t)\) where
- \(r\) is the release time
- \(d\) is the deadline by which the job must be completed
- \(t\) is the duration it takes to complete the job, once you started it

Allocate jobs to one machine. Our machine
- can only run one job at a time,
- once started on job, must finish it before taking on another job.

**Input:** Set of jobs \(\{(r_i, d_i, t_i)\}_{i=1}^n \subset \mathbb{N}^3\)

**Output:** can we schedule all jobs so that each is completed by its deadline?

We will show this problem is **NP-complete**

**Membership in NP:**
- **Proof/witness:** the proof/witness is a scheduling (linear size)
- **verification algorithm:** check that the scheduling satisfies the resease time and deadline (linear time)
Proof of Hardness

- Polynomial transformation from SUBSET-SUM to our problem
Proof of Hardness

- Polynomial transformation from SUBSET-SUM to our problem
- Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{N}$ and $T \in \mathbb{N}$ be an instance of the SUBSET-SUM problem.
Proof of Hardness

- Polynomial transformation from SUBSET-SUM to our problem
- Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{N}$ and $T \in \mathbb{N}$ be an instance of the SUBSET-SUM problem.
- Let $S := \sum_{i=1}^{n} x_i$, and consider the following jobs:
  - $(0, S + 1, w_i)$, for $i \in [n]$
  - $(T, T + 1, 1)$
Proof of Hardness

- Polynomial transformation from SUBSET-SUM to our problem
- Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{N}$ and $T \in \mathbb{N}$ be an instance of the SUBSET-SUM problem.
- Let $S := \sum_{i=1}^{n} x_i$, and consider the following jobs:
  - $(0, S + 1, w_i)$, for $i \in [n]$
  - $(T, T + 1, 1)$
- Note that there is a good scheduling iff the job $(T, T + 1, 1)$ gets scheduled at time $T$, which can only happen if there is a subset of the other jobs that can be scheduled exactly between $[0, T]$. 
Solving System of Equations

- **0-1 QUADEQ** (quadratic equations problem)
  - **Input:** System of quadratic equations
    \[ \{ Q_i(x_1, \ldots, x_n) = 0 \}_{i \in [m]} \cup \{ x_i^2 - x_i = 0 \}_{i=1}^n \]
  - **Output:** YES \( \iff \) there is a solution to the system above.
Solving System of Equations

- **0-1 QUAD Eq** (quadratic equations problem)
  - **Input:** System of quadratic equations
    \[ \{Q_i(x_1, \ldots, x_n) = 0\}_{i \in [m]} \cup \{x_i^2 - x_i = 0\}_{i=1}^n \]
  - **Output:** YES if there is a solution to the system above.

- QUAD Eq is NP-complete
0-1 QUADEQ (quadratic equations problem)

- **Input:** System of quadratic equations
  \[ \{Q_i(x_1, \ldots, x_n) = 0\}_{i \in [m]} \cup \{x_i^2 - x_i = 0\}_{i=1}^n \]
- **Output:** YES \(\iff\) there is a solution to the system above.

QUADEQ is NP-complete

Membership in NP: proof/witness is a solution to the equations.
Solving System of Equations

0-1 QUADEQ (quadratic equations problem)

- **Input:** System of quadratic equations
  \[ \{ Q_i(x_1, \ldots, x_n) = 0 \}_{i \in [m]} \cup \{ x_i^2 - x_i = 0 \}_{i=1}^n \]
- **Output:** YES ⇔ there is a solution to the system above.

QUADEQ is NP-complete

- Membership in NP: proof/witness is a solution to the equations.
- Completeness for NP: reduction from 3SAT
  Encode each clause as a quadratic equation.
Integer Programming

- **IPROG**
  - **Input:** System of linear inequalities \( \{ \sum_{j=1}^{n} a_{ij}x_j \geq b_i \}_{i \in [m]} \), where \( x_i \in \mathbb{Z} \)
  - **Output:** YES \( \iff \) there is a solution to the system above.

- **IPROG** is NP-complete
- **Membership in NP:** proof/witness is a solution to the inequalities.
- **Completeness for NP:** reduction from 3SAT
  - Encode each clause as a linear inequality.
  - Enforce boolean constraint by adding linear inequalities.
Integer Programming

- **IPROG**
  - **Input:** System of linear inequalities \( \{ \sum_{j=1}^{n} a_{ij} x_j \geq b_i \}_{i \in [m]} \), where \( x_i \in \mathbb{Z} \)
  - **Output:** YES \( \iff \) there is a solution to the system above.

- IPROG is NP-complete
Integer Programming

- **IPROG**
  - **Input:** System of linear inequalities \( \{ \sum_{j=1}^{n} a_{ij} x_j \geq b_i \}_{i \in [m]} \), where \( x_i \in \mathbb{Z} \)
  - **Output:** YES \(\iff\) there is a solution to the system above.

- IPROG is NP-complete

- Membership in NP: proof/witness is a solution to the inequalities.
Integer Programming

- **IPROG**
  - **Input:** System of linear inequalities \( \{ \sum_{j=1}^{n} a_{ij} x_j \geq b_i \}_{i \in [m]} \), where \( x_i \in \mathbb{Z} \)
  - **Output:** YES \( \Leftrightarrow \) there is a solution to the system above.

- IPROG is NP-complete
- Membership in NP: proof/witness is a solution to the inequalities.
- Completeness for NP: reduction from 3SAT
  
  Encode each clause as a linear inequality.
  
  Enforce boolean constraint by adding linear inequalities.
Integer Programming

- IPROG
  - **Input:** System of linear inequalities \( \{ \sum_{j=1}^{n} a_{ij} x_j \geq b_i \}_{i \in [m]} \), where \( x_i \in \mathbb{Z} \)
  - **Output:** YES \( \iff \) there is a solution to the system above.
- IPROG is NP-complete
- Membership in NP: proof/witness is a solution to the inequalities.
- Completeness for NP: reduction from 3SAT
  
  Encode each clause as a linear inequality.
  
  Enforce boolean constraint by adding linear inequalities.
  
  - \( x_1 \lor \overline{x}_2 \lor x_3 \iff x_1 + (1 - x_2) + x_3 \geq 1 \)
  - \( 0 \leq x_i \leq 1 \)
Intractability

- Scheduling Problems
- Algebraic Problems
- Mathematical Programming Problems
- Taxonomy of Hard Problems

Further Explorations

- Computational view of the world
- Courses
- Research
- AMA

Acknowledgements
Packing Problems

**Packing problems:** given a collection of objects (with certain conflicts between them), want to choose at least $k$ of them.

- **NP-complete packing problems:**
  1. Clique
  2. Independent Set
  3. Set packing

  **Input:** collection of subsets $S_1, S_2, \ldots, S_m$ of $[n]$, number $k \in \mathbb{N}$

  **Output:** YES $\iff$ there is collection of $k$ sets with empty pairwise intersection
Covering Problems

Covering Problems: given collection of objects and a particular goal, want to choose a subset of objects of size \( \text{at most} \ k \) that achieve this goal

- NP-complete covering problems:
  1. Vertex Cover
  2. Set Cover

- **Input:** subsets \( S_1, \ldots, S_m \) of \([n]\), \( k \in \mathbb{N} \)
- **Output:** YES \( \iff \) there are at most \( k \) subsets \( S_i \) whose union is all of \([n]\)
Partitioning Problems: dividing collection of objects into subsets such that each object appears in exactly one of these subsets

- NP-complete partitioning problems
  - Graph Coloring
  - 3-dimensional matching

**Input:** given disjoint sets $X$, $Y$, $Z$ each of size $n$, and subset $T \subset X \times Y \times Z$

**Output:** YES $\iff$ there are $n$ triple such that every element of $X \cup Y \cup Z$ is contained in exactly one of the triples
Sequencing Problems

- NP-complete sequencing problems
  - directed Hamiltonian cycle
  - directed Hamiltonian path
  - TSP
Numerical & Mathematical Programming Problems

- NP-complete numerical & mathematical programming problems
  - Subset-Sum
  - Integer Programming
  - 0-1 Quadratic Programming
Constraint Satisfaction Problems

- NP-complete constraint satisfaction problems
  - SAT
  - 3SAT
  - Circuit SAT
Intractability
- Scheduling Problems
- Algebraic Problems
- Mathematical Programming Problems
- Taxonomy of Hard Problems

Further Explorations
- Computational view of the world
- Courses
- Research
- AMA

Acknowledgements
What have we learned

- Decision problems are not very restrictive - thus good to build theory upon
- Reductions between problems
  - allows us to put partial order on hardness of problems
  - classify problems according to their difficulty
- Three important classes of decision problems: P, NP and coNP
- Completeness for NP
- Problems that are NP-hard but not in NP
What else is there?

- this is just the tip of the iceberg
  - parallel computation
  - non-uniform computation
  
    What if we could give a different algorithm for each input size?
  - randomized computation
  - What about space requirements?
  - What about problems with more quantifiers ($\exists, \forall$)?
    - distributed
    - streaming (low memory, few passes through data)
    - online algorithms
    - algebraic algorithms
    - approximation algorithms
    - numerical methods
    - parallel algorithms
Algorithmic Side

- Courses being offered in Winter 2024
  - Prof Assadi’s CS 860: modern topics in graph algorithms
  - Prof Khanna’s CS 860: algorithmic gems
Courses being offered in Winter 2024

- Prof Blais CS 365: undergraduate complexity
- Prof Blais CS 764: graduate complexity
Research Opportunities at UW!

Consider doing a URA, URF or USRA with a U Waterloo faculty!
See research openings at:

- **Undergraduate Research Assistanship (URA):**
  
  [https://cs.uwaterloo.ca/computer-science/current-undergraduate-students/research-opportunities/undergraduate-research-assistantship-ura-program](https://cs.uwaterloo.ca/computer-science/current-undergraduate-students/research-opportunities/undergraduate-research-assistantship-ura-program)

- **Undergraduate Research Fellowship (URF):**
  
  [https://cs.uwaterloo.ca/current-undergraduate-students/research-opportunities/undergraduate-research-fellowship-urf](https://cs.uwaterloo.ca/current-undergraduate-students/research-opportunities/undergraduate-research-fellowship-urf)

- **Mathematics Undergraduate Research Assistanship (MURA):**
  
  [https://uwaterloo.ca/math/undergraduate-research-assistantships-faculty-mathematics](https://uwaterloo.ca/math/undergraduate-research-assistantships-faculty-mathematics)

- **For Canadians, please check out NSERC’s USRA:**
  
  [https://cs.uwaterloo.ca/usra](https://cs.uwaterloo.ca/usra)
Ask me anything!
Acknowledgement

Based on

- [Kleinberg Tardos 2006, Chapter 8]
References I

Cormen, Thomas and Leiserson, Charles and Rivest, Ronald and Stein, Clifford (2009)
MIT Press

Dasgupta, Sanjay and Papadimitriou, Christos and Vazirani, Umesh (2006)
Algorithms

Erickson, Jeff (2019)
Algorithms
https://jeffe.cs.illinois.edu/teaching/algorithms/

Kleinberg, Jon and Tardos, Eva (2006)
Algorithm Design.
Addison Wesley