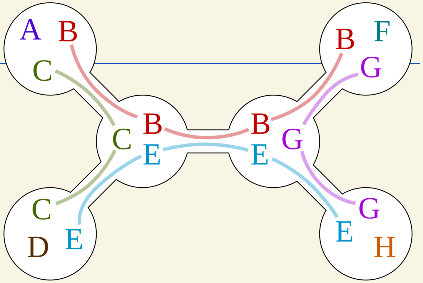
Introduction and Overview



DM898: Parameterized Algorithms
Lars Rohwedder

Today's lecture

- What is this course about?
- Organization
- Introductory examples

What is this course about?

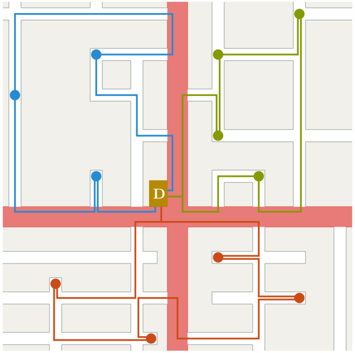
Operations Research

Operations Research is a field at the intersection of computer science, mathematics, managemental sciences, and military. It concerns optimization problems that are important in real world.

Examples

- Train scheduling
- Production planning
- Packet delivery
- Warehouse order picking
- Kidney exchange programs
- Telecommunications network design

  . . .

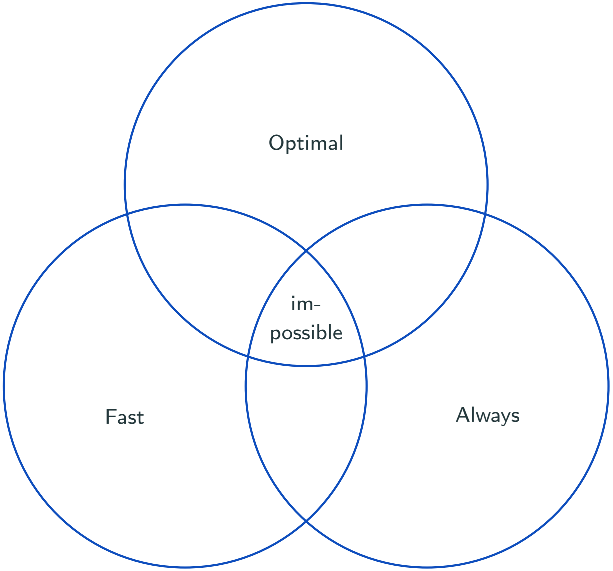


https://commons.wikimedia.org/wiki/File:Illustration_of_the_Vehicle_Routing_Problem.svg

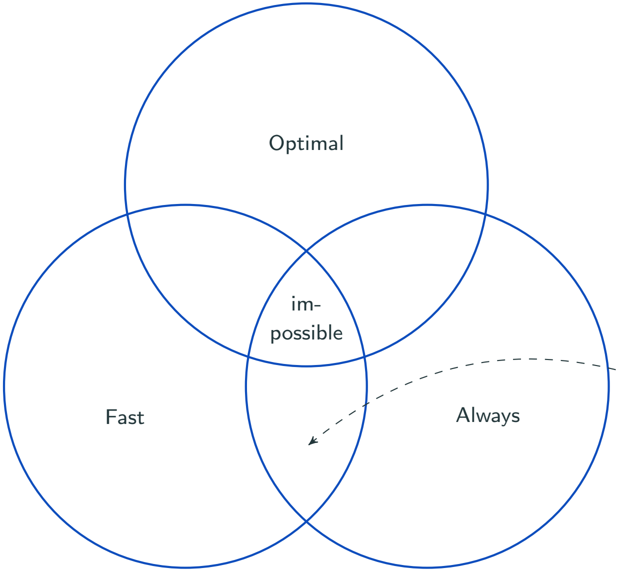
These are often **inherently difficult**, for details see complexity theory recap later.

. . . what can we do?

Algorithms for difficult problems

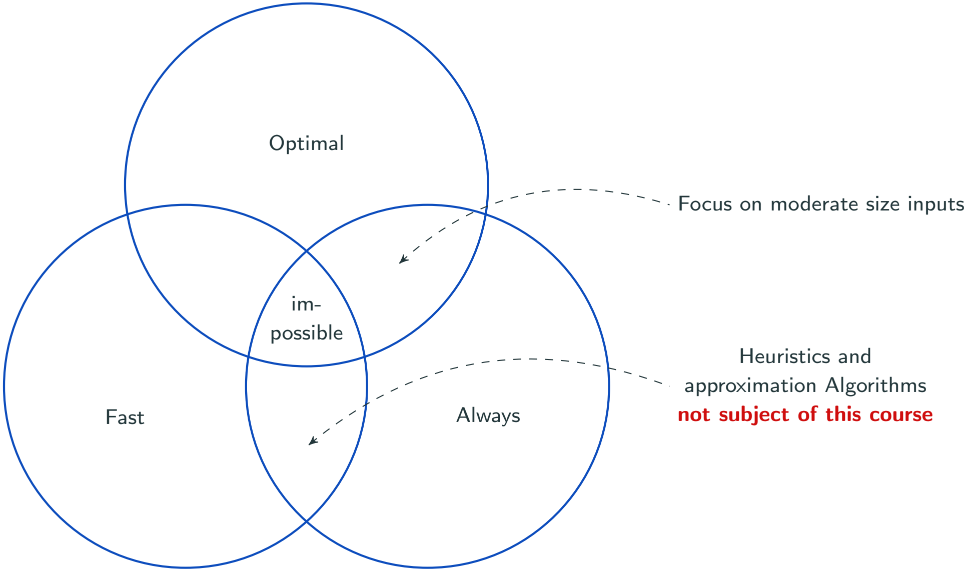


Algorithms for difficult problems



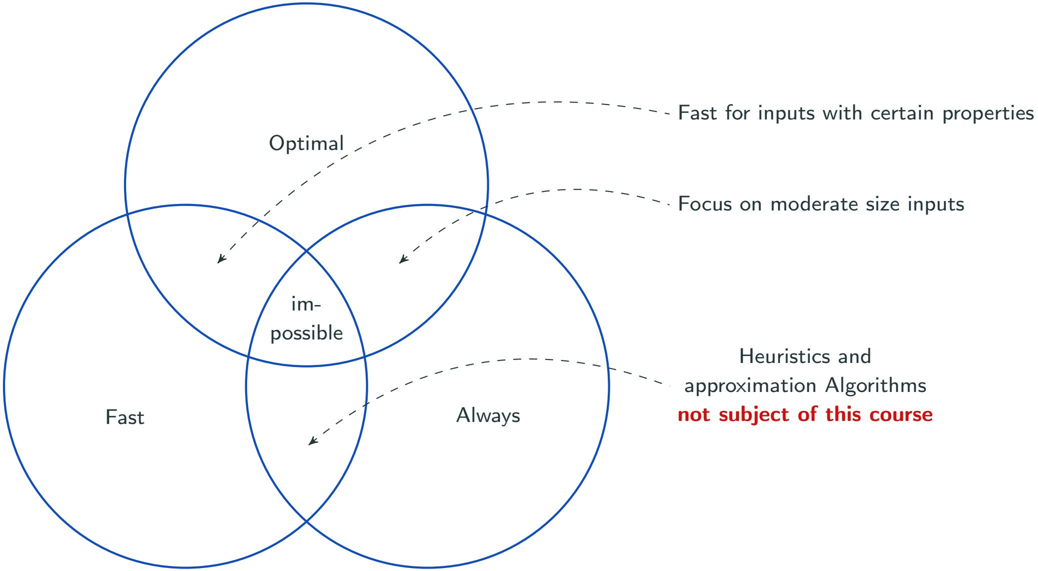
Heuristics and approximation Algorithms
not subject of this course

Algorithms for difficult problems



not subject of this course

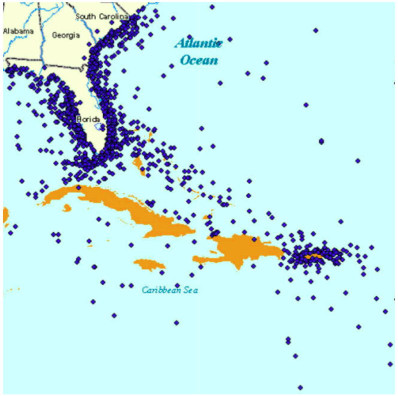
Algorithms for difficult problems


Optimal
im-
possible
Fast
Always
Fast for inputs with certain properties
Focus on moderate size inputs
Heuristics and
approximation Algorithms
not subject of this course

Operations Researcher's perspective

Operations Researchers have been using and developing exact algorithms for NP-hard problems since the early days of computers (at least 1960s).

As an example we look at a research paper from the field[1].



- Given emergency distress calls (from historical data), determine the best positions to place US Coast Guard air stations
- Exactly p stations should be placed
- Minimize weighted average distance from emergencies to closest stations and costs to establish facilities

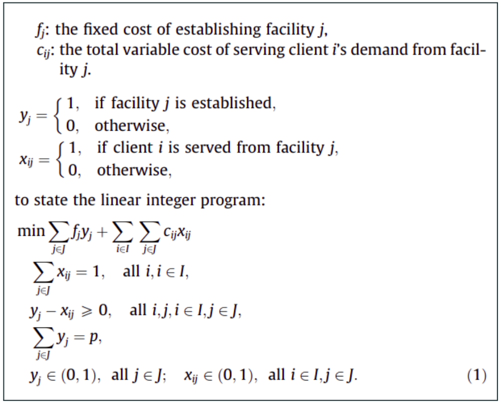
[1] "US Coast Guard air station location with respect to distress calls: A spatial statistics and optimization based methodology". Afshartous, Guan, and Mehrotra. European Journal of Operational Research. 2009.

Operations Researcher's perspective (cont)

Approach

- **Complete enumeration** of all possible solutions would take too long
- Formulate problem as **Integer Linear Program (ILP)**
- Use commercial solver (e.g. IBM CPLEX or Gurobi)
- Solver heavily uses **preprocessing** and **branching** to solve ILP
- Hope that solver finishes in reasonable time. Otherwise, fine-tune formulation or reconfigure solver

ILP Formulation



f_j: the fixed cost of establishing facility j,
c_{ij}: the total variable cost of serving client i's demand from facility j.

$$y_j = \begin{cases} 1, & \text{if facility } j \text{ is established,} \\ 0, & \text{otherwise,} \end{cases}$$
$$x_{ij} = \begin{cases} 1, & \text{if client } i \text{ is served from facility } j, \\ 0, & \text{otherwise,} \end{cases}$$

to state the linear integer program:

$$\begin{aligned} &\min \sum_{j \in J} f_j y_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\ &\sum_{j \in J} x_{ij} = 1, \quad \text{all } i, i \in I, \\ &y_j - x_{ij} \geqslant 0, \quad \text{all } i, j, i \in I, j \in J, \\ &\sum_{j \in J} y_j = p, \\ &y_j \in (0,1), \ \text{all } j \in J; \quad x_{ij} \in (0,1), \ \text{all } i \in I, j \in J. \end{aligned} \tag{1}$$

Advantages: generic approach, capable of handling very complex problems, relatively low development costs

Theoretitian's perspective

Theory of algorithms (for example, asymptotic running time analysis) is a simplified model of practice.

$\rightsquigarrow$ enables rigorous proofs contributing to **reliable algorithms, explainations for empirical behavior, transfers of insights to practice**

Theoretitian's perspective

Theory of algorithms (for example, asymptotic running time analysis) is a simplified model of practice.

$\rightsquigarrow$ enables rigorous proofs contributing to **reliable algorithms, explainations for empirical behavior, transfers of insights to practice**

Problem: Classical computational complexity only distinguishes between easy problems (P – polynomial time solvable) and hard problems (NP-hard). Insufficient as a theory for Operations Resarch, e.g.:

- Cannot explain success of polynomial time preprocessing for NP-hard problems
- Cannot explain success of branching and ILP techniques

Theoretitian's perspective

Theory of algorithms (for example, asymptotic running time analysis) is a simplified model of practice.

$\rightsquigarrow$ enables rigorous proofs contributing to **reliable algorithms, explainations for empirical behavior, transfers of insights to practice**

Problem: Classical computational complexity only distinguishes between easy problems (P – polynomial time solvable) and hard problems (NP-hard). Insufficient as a theory for Operations Resarch, e.g.:

- Cannot explain success of polynomial time preprocessing for NP-hard problems
- Cannot explain success of branching and ILP techniques

Parameterized Complexity

- **Parameterization**: a quantity k derived from input, usually something that is small in typical instances.
- Assumption: $k \ll n$
- Efficient: **fixed-parameter tractable (FPT)** running time of the form $f(k) \cdot n^{O(1)}$ for a function f, e.g. $O(2^k n^3)$, $O(k!n)$, $O(2^{2^k} n^2)$
- Less desirable: **XP** running time of $n^{f(k)}$

- Parameterized complexity offers some explanations for empirical behavior of algorithms used in operations research
- Detailed understanding of when instance is hard to solve
- Interesting theory also motivated from mathematical curiosity

Goals of the course

You will learn the fundamental results for parameterized algorithms **through the lens of Operations Research**:

- Preparation for theoretical research in the field of parameterized algorithms
- Approaches for attacking NP-hard problems in practice
- Connections between the above

Relationship to other courses

- Theory of parameterized algorithms naturally builds up or relates to courses **Discrete Math**, **Advanced Algorithms**, **Complexity and Computability**
- The course **Linear and Integer Programming** focuses on commercial ILP solvers and complements this course by giving more details on these methods, but no theoretical perspective

Organization

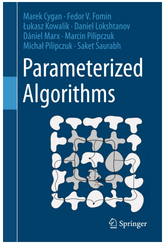
Your teacher

Lars Rohwedder

Short Bio

- Focus on systems during early career: Research assistant at Oracle, 2013-2014, and VMWare, 2015 (both San Francisco Bay Area)
- 2019: PhD from University of Kiel on theoretical algorithms
- 2019-2021: Post-Doc at EPFL
- 2022-2024: Assistant prof. at Maastricht University
- Since Oct. 2024: Associate prof. at IMADA, SDU.

Resources



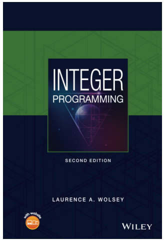
Marek Cygan · Fedor V. Fomin
Łukasz Kowalik · Daniel Lokshtanov
Dániel Marx · Marcin Pilipczuk
Michał Pilipczuk · Saket Saurabh
Parameterized Algorithms
Springer

Main textbook

- Parameterized Algorithms by Cygan et al.
- Purely theoretical textbook
- Lectures and exercises based on various chapters from book, but with added references to practice and practical content
- PDF available for free: `https://www.mimuw.edu.pl/~malcin/book/parameterized-algorithms.pdf`

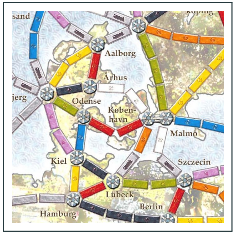
Additional resources

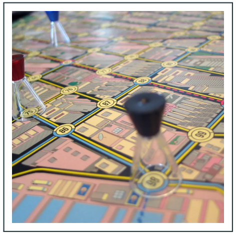
- `https://larsrohwedder.com/teaching/dm898-25` for everything you need (link also on itslearning)
- Textbook **Integer Programming (2nd edition)** by Wolsey for practical aspects, in particular, Branch-and-Bound (not strictly required for course)



Project

- In teams of two you get an intricate boardgame-inspired problem, **which have not been researched yet!**
- Throughout the course you tackle your problem in project consisting of **4 elements**, which resemble actual research:
    1. formalizing your problem including possible parameterizations, modelling it as ILP, and noting relations it to other problems and interesting special cases
    2. Developing algorithms based on techniques from course (preprocessing rules, branching, etc.)
    3. Determining complexity regarding different parameters
    4. Either implementing an algorithm or analyzing it theoretically (choice)





Course organization

During class:

- mix of lecture, exercise solving, and discussion of homework or projects

Projects:

- Given out 3 weeks into semester
- Deadlines for drafts of each of the 4 project elements throughout semester (TBD), discussion in class
- Final report with all revised elements at the end of the semester

Assessment

- 80% of grade comes from **oral exam** during exam period, questions about topics from course or on project report
- 20% comes from **project** evaluation

Introductory examples