Approximation Algorithms

Syamantak Das

Nicole Megow

University of Bremen

FB 3: Mathematics/Computer Science

Summer Term 2017

Administrative Trivia

- Lectures : Wed 12:15-13:45 MZH 1090
  Thu 10:15- 11:45 MZH 5210

- 6 ECTS

- Lecture Notes : Mainly Blackboard and Rough Notes

- Course webpage : TBA

- Recommended Books :

  The Design of Approximation Algorithms by D. Williamson and D. Shmoys

  Free online version available at :

  http://www.designofapproxalgs.com/

  Approximation Algorithms by Vijay Vazirani

Exercises and Exams

- Exercises : 8-10 Sheets to be uploaded on the webpage
  To be handed in 1 week from the upload date

- Exams :

  Option 1: 50% in the Exercises to qualify + Fachgespräch
  (Pass with 4.0 if not appearing for Fachgespräch)

  Option 2: Full Oral Exam

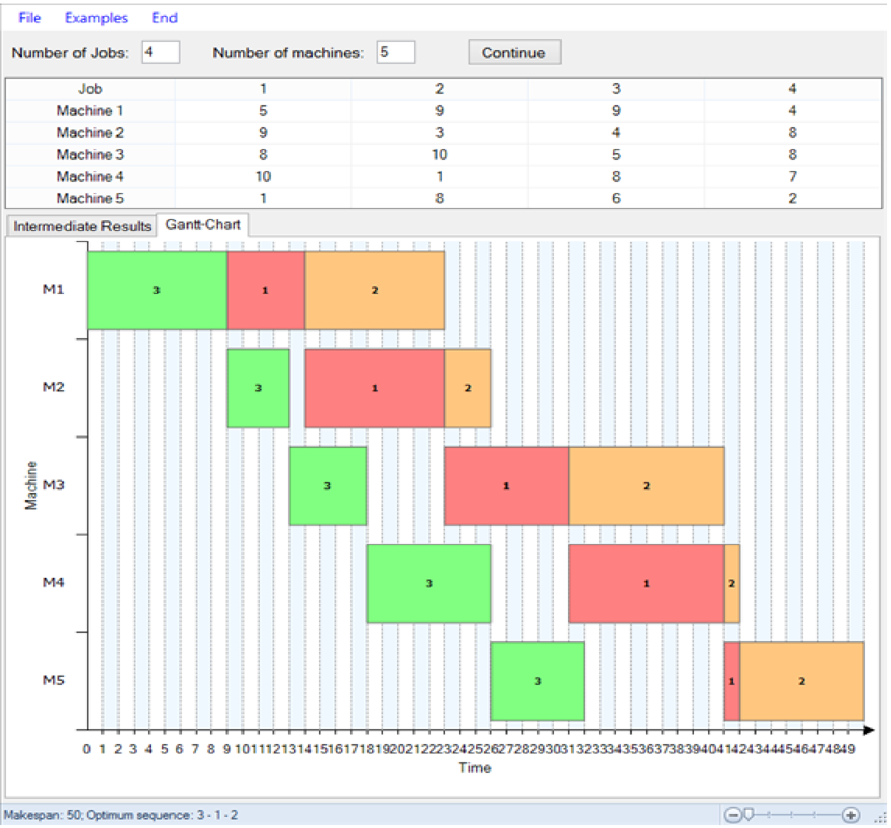
Approximation Algorithms

Example 1: The Travelling Salesman

- The salesman has to visit all the cities
- There is a cost associated to go from city A to city B

Goal : Find the tour of cheapest total cost

Example 2: Scheduling Jobs on Processors

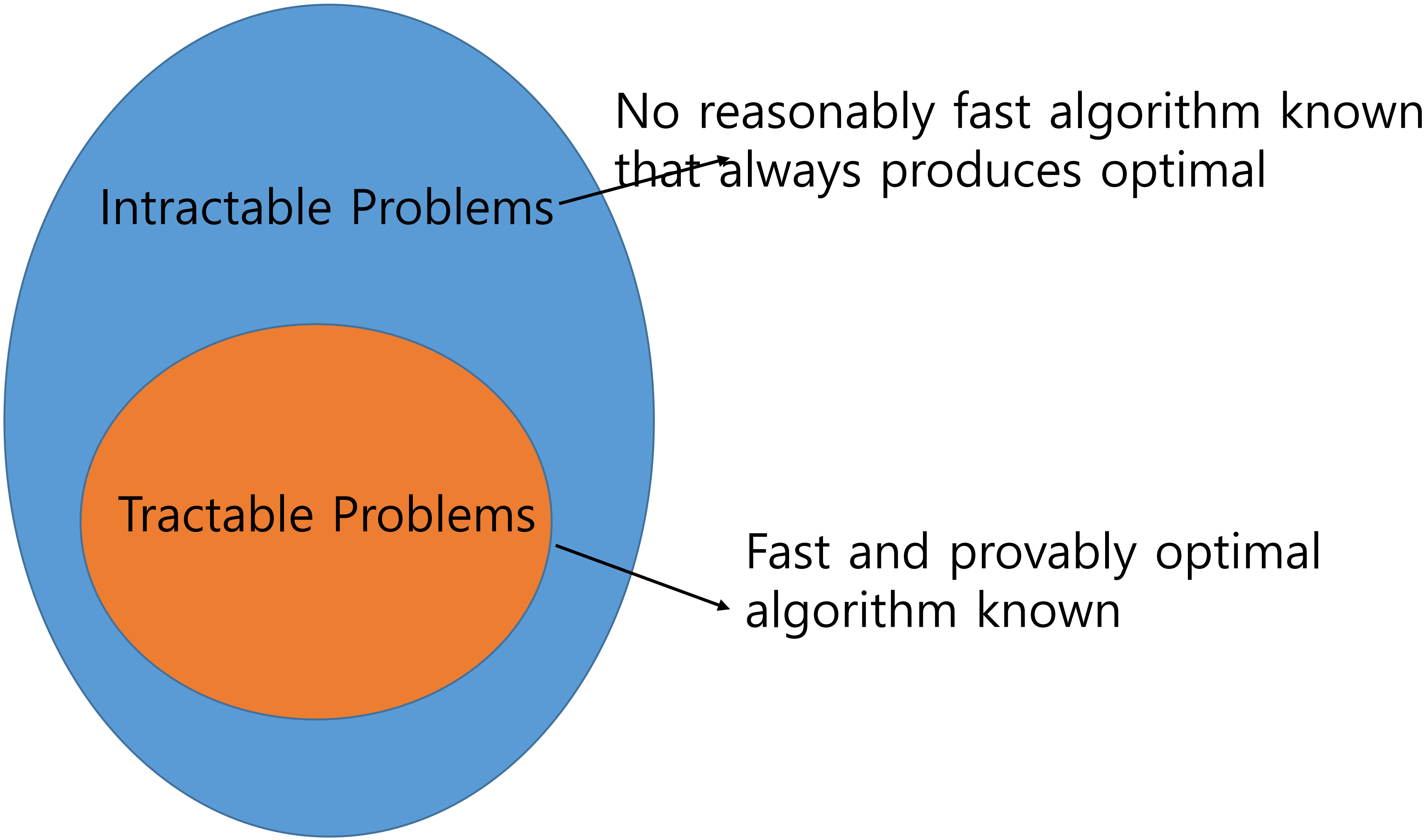


Job	1	2	3	4
Machine 1	5	9	9	4
Machine 2	9	3	4	8
Machine 3	8	10	5	8
Machine 4	10	1	8	7
Machine 5	1	8	6	2

- Identical parallel processors
- Schedule jobs on the processors

Goal : Find a schedule that balances load

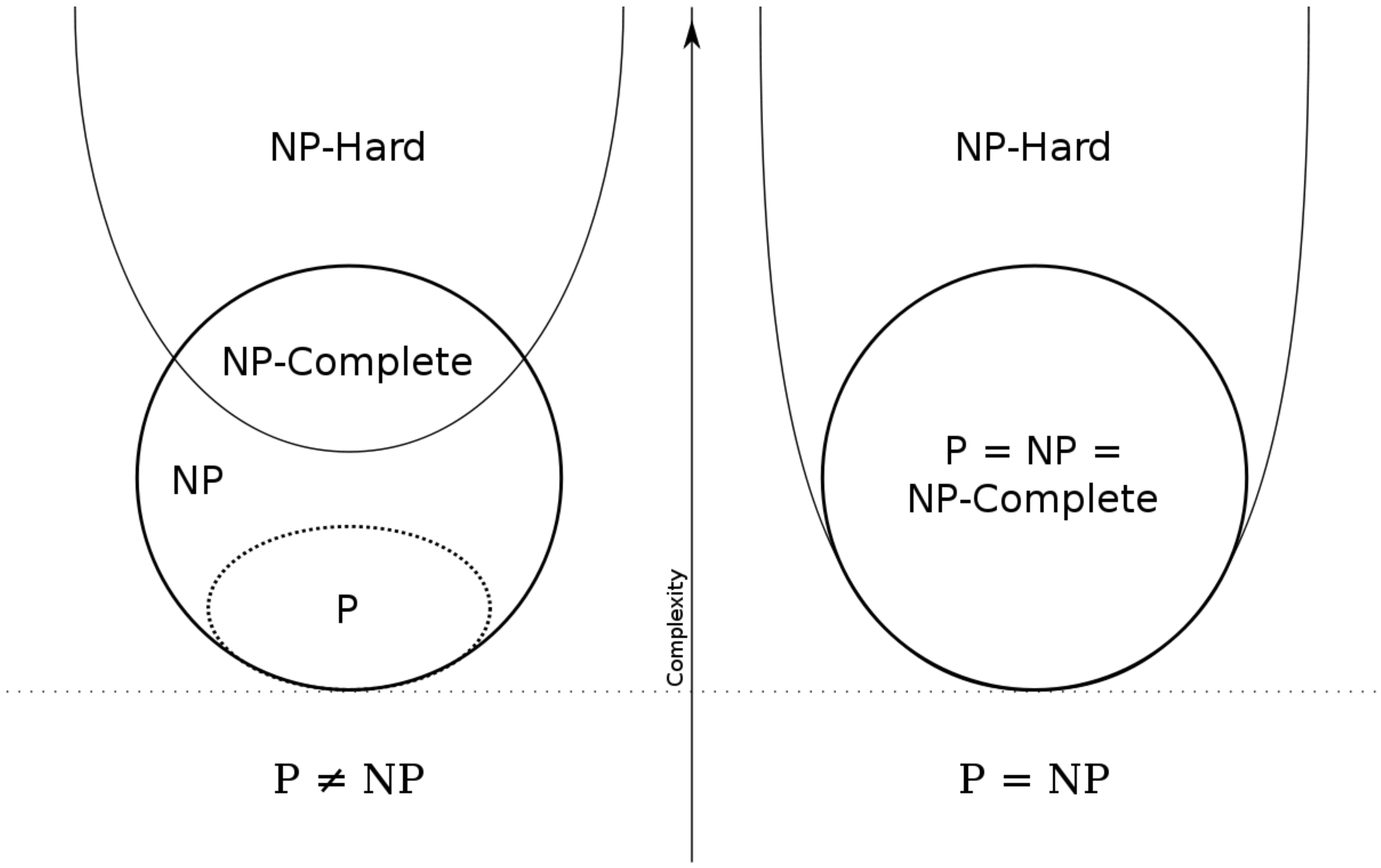
How efficiently can a computer solve them?



Polynomial time solvability

- Problems that are known to have algorithms which run in time polynomial in *size of input*
- Size of input : Number of bits required to describe the problem
- Such problems are said to be in class P
- Problems that can be verified in polynomial time are said to be in NP – not known if they can be solved in polynomial time

Complexity classes



What do we do for Intractable Problems ?

- Design heuristics :

  Backtrack search, tabu search, simulated annealing etc.

- Might give an optimal but run in exponential time or run faster but do not guarantee optimal solution

- Approximation algorithms : Design fast algorithms that can *provably* give a *near optimal* solution in the worst case

Approximation Factor

- I : an instance of our problem

- Objective that we want to minimize or maximize, e.g. shortest TSP tour

- $OPT(I)$: optimal value of this objective
  $ALG(I)$: objective for the algorithm

- An algorithm is an α −approximation if for every instance I

$$\alpha = \frac{ALG(I)}{OPT(I)}$$

Remarks

- For maximization problems, $\alpha \leq 1$
- Sometimes, α might depend on size of the instance and technically should be $\alpha(I)$
- Why multiplicative guarantee and not additive?

Course Objectives

- Develop a toolkit of approximation algorithms that can solve variety of problems in practice
- Understand the mathematical nature of fundamental problems (especially intractable problems) in computation by studying their approximability
- Have Fun ! : Beautiful Connections between algorithm design and Maths or other areas of Computer Science

Caveats

- Focus on the worst case and hence might ignore possible algorithms that work great in average cases
- Unlike some techniques, for example, mathematical techniques, there is often no continuous tradeoff between runtime and quality of solution
- Framework not (directly) applicable to decision problems