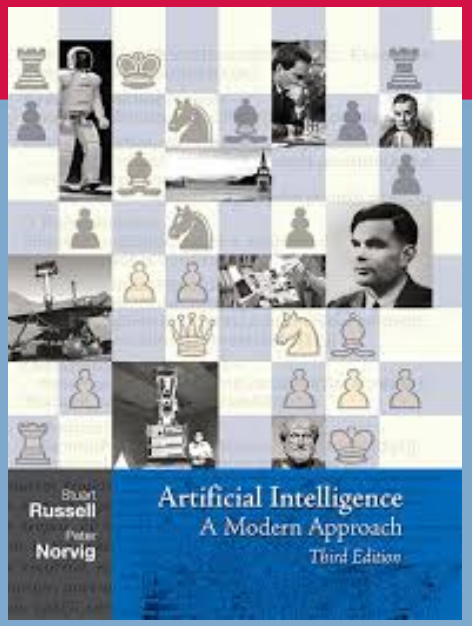
CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

Metaheuristic Optimization

Introduction

Dr. Diarmuid Grimes



Acknowledgments

I am grateful to Dr. Alejandro Arbelaez who shared with me the following material.

Personal and Research Background

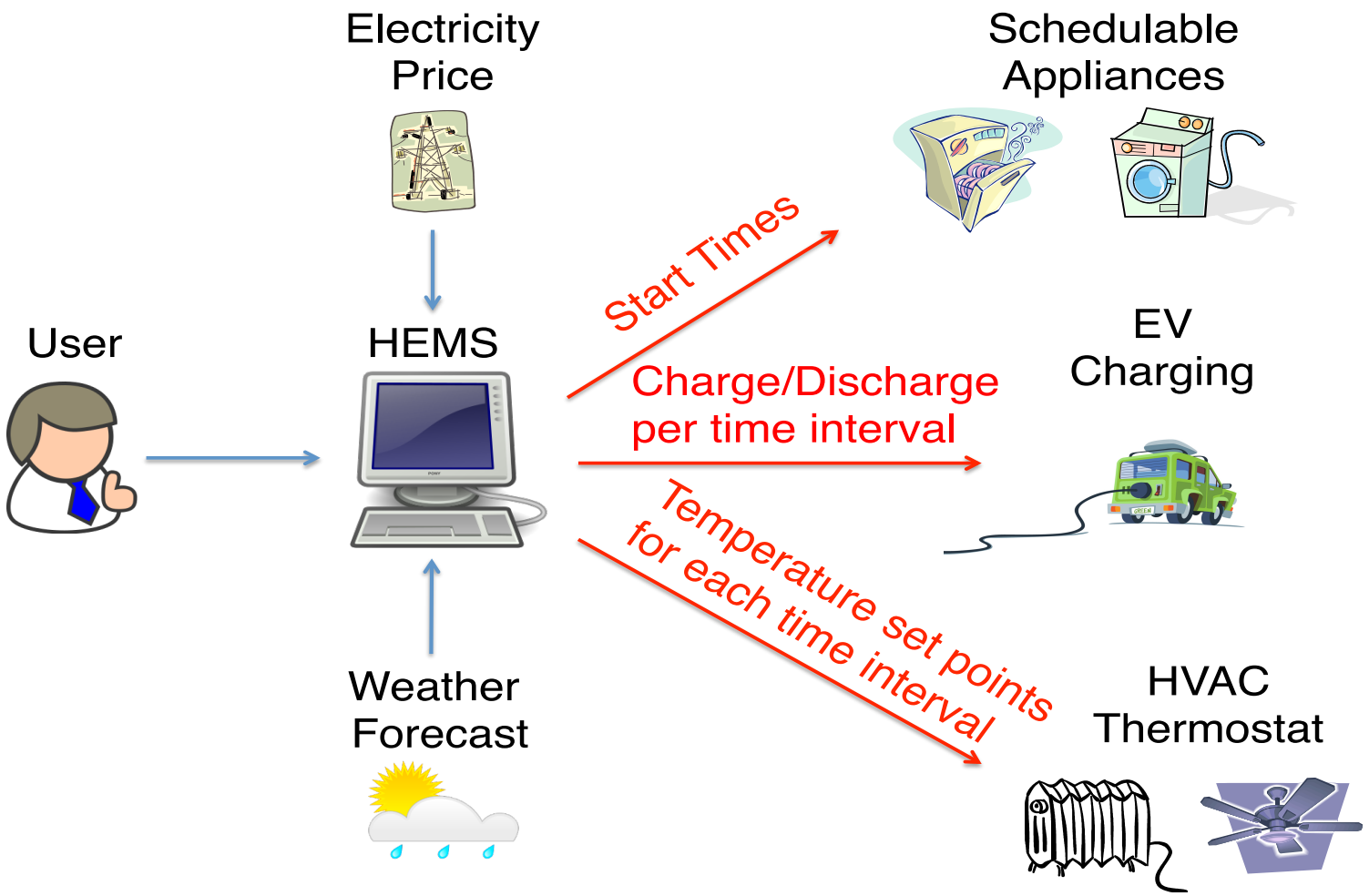
Diarmuid.Grimes@cit.ie

B223L

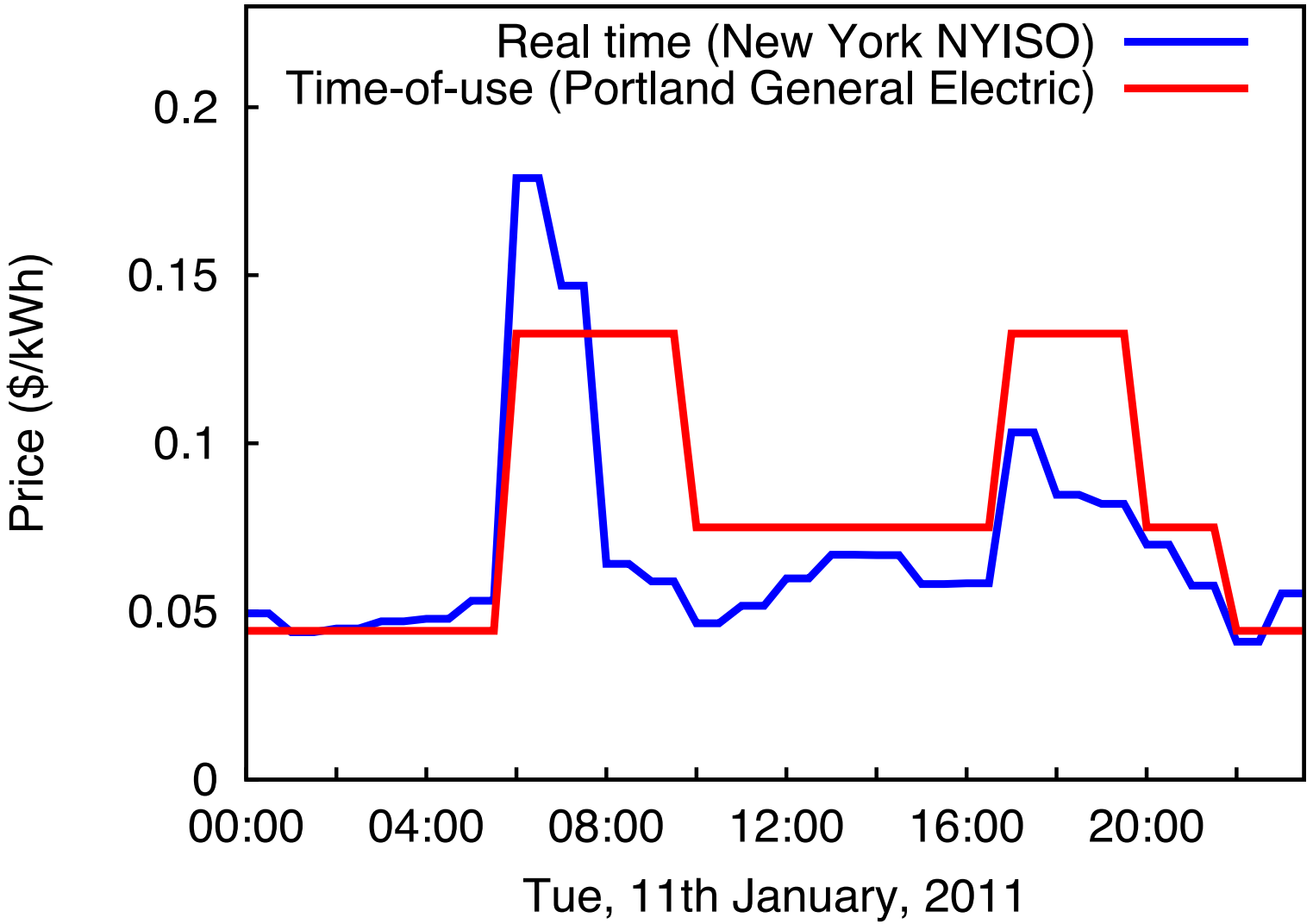
Ext: 5506

- Combinatorial Optimisation
  - Generic approaches for CP, LNS, Genetic Algorithms
  - Dedicated methods for different application areas:
    - Energy (minimize consumption/costs in residential / industrial sectors)
    - Transport (schedule maintenance for rail fleets to minimise cost function, schedule timetable of urban electric rail fleet to minimise peak usage)
- Machine Learning
  - Application areas:
    - Energy (real time electricity price prediction using historical data and forecasts for demand and renewable supply)
    - Transport (predicted health of rail-fleet components such as axle bearings using condition monitoring)
  - One of key findings: ”best” predictor according to ML metrics not best predictor for optimisation

Home Energy Management System



Time-Variable Pricing Tariffs



Home Energy Management Problem

- Schedulable appliances: Start time
- For each time interval:
  - EV: Charge/discharge power → battery state
  - HVAC: Heating/cooling power → temperature

“Best” ML predictor for Optimization

- Standard price forecasting metrics did not result in expected results, i.e. best forecast according to standard metrics such as MSE/MAE did not result in best performance when used for energy aware scheduling!

Why?!!!! – example with 3 tasks of 1kw, 10kw, and 100kw, each lasting 30 mins.

- Price forecast 1 for the next 3 time periods is €1, €2, €3
- Price forecast 2 for the next 3 time periods is €1000, €100, €10
- Actual price for the next 3 time periods is €3, €2, €1

Which is best forecast, F1 or F2?

- According to MAE/MSE F1 is **much better**

But …

- If F1 used: €3x100+€2x10+€1x1=€321
- If F2 used: €3x1+€2x10+€1x100=€123

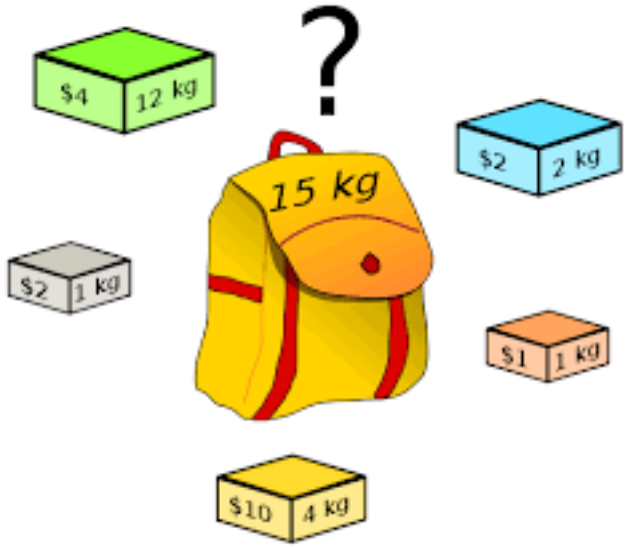
Real life deployment needed for finetuning!

- Deployed in a home using a time-of-use price tariff
- User thought there was an issue as the home wasn't as warm on Sunday mornings
  - Issue wasn't in the algorithm/code but that price tariff for Sunday was flat so did not preheat home in cheaper night time as on other days
- More importantly, assumption had been made!
  - During night time temperature bounds were relaxed, but still constraint was added to ensure it didn't get too cold
  - Did not consider that it could get too hot!
  - User was waking in the middle of the night in sauna-like conditions
  - Again issue wasn't in the algorithm/code, but in the settings.
    - Price increased by so much at 5am that optimal according to the settings was to really heat the house at 2am and allow heat loss to reduce temperature to required setting at 7am!

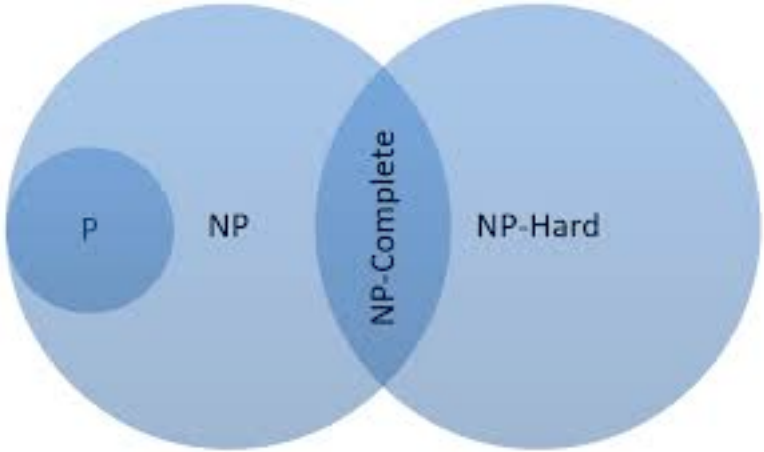
Learning Outcomes

- Module Descriptor

https://courses.cit.ie/index.cfm/page/module/moduleId/13443



Combinatorial Optimization



Complexity

Learning Outcomes

- Module Descriptor

https://courses.cit.ie/index.cfm/page/module/moduleId/13443

- Categorize a **real-life problem** with respect to its **computational complexity**
- Assess the **benefits** and **limitations** of **meta-heuristics** to solve NP-hard problems
- Solve an NP-hard problem with **meta-heuristics** to find a satisfactory lower-bound solution
- Analyze the average **performance** of a randomised algorithm to solve an NP-hard problem
- Apply **nature-inspired** and **local search** meta-heuristics to solve real-life problems

Main Components

- Complexity
- Combinatorial Problems
- Local Search algorithms
- Nature inspired algorithms

Module workload

2 hours lecture every week

2 hours lab

3 hours Independent studies

Assessment Breakdown

- Week 6 (50%)
- Week 12 (50%)

Programming Language

- This module is designed for students with prior programming experience (and basic probability & statistics knowledge)

- Programming Language → Python

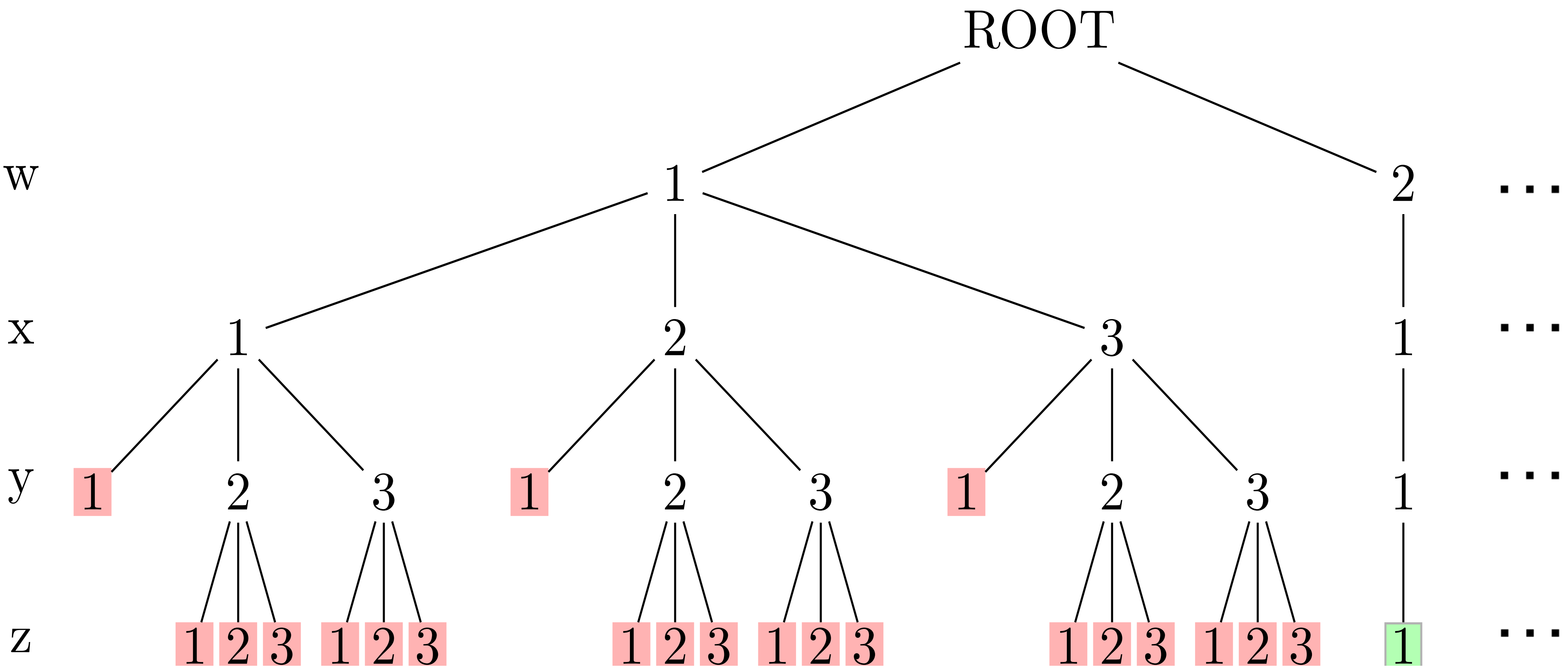
Content

- Introduction
- NP-completeness
- Population-based Meta-heuristics
  - Genetic Algorithms
  - Particle Swarm optimization
  - Ant-Colony Optimization
- Single-solution based Meta-heuristics
  - Local Search
  - Simulated Annealing
  - Etc..

Combinatorial Problems

- Set of *n* variables ***V***.
  - E.g. Sudoku, variables are the empty squares that we need to fill
- Set of finite domains ***D***, possible values each variable can take.
  - For sudoku each variable has the domain 1-9
- Set of constraints ***C***, each constraint involves at least one variable:
  - For sudoku, each variable is involved in 3 *AllDifferent* constraints across row, column and subgrid.
  - Alternative constraints:
    - Unary constraint: $x_i > 10$
    - Precedence constraint: $x_j + p_j < x_k$
- Size of search space d^n: typically too large for brute force search.

Search Space/Tree



Search Space/Tree

	1	2		3	4	5	6	7
	3	4	5		6	1	8	2
		1		5	8	2		6
		8	6					1
	2				7		5	
		3	7		5		2	8
	8			6		7		
2		7		8	3	6	1	5

- Variable $x_{i,j}$ for every empty cell
- Domains {1…9}
- Constraint enforcing all different in values in each row, in each column, and each subgrid.

Combinatorial problem example: Timetabling

- Assign every lecture/lab a classroom and a timeslot.
- Combinatorics are huge, particularly if we don't consider constraints.
- 2 variables for each lecture/lab to be assigned
- Domains?
  - Set of possible classrooms (>200) / timeslots (45)
- Sample constraints?
  - lecturer can't teach 2 classes at the same time,
  - room can't hold 2 classes at same time,
  - classroom has capacity

CO.MSC-A! - KARIN_9_1 - MSc. in Artificial Intelligence

NOTE THAT THESE TIMETABLES ARE PROVISIONAL. PLEASE CONFIRM WITH HEAD OF DEPARTMENT.

Weeks 2 (9 Sep 2019-15 Sep 2019)

	Monday		Tuesday		Wednesday	Thursday	Friday
9:00				CO.MSC-AI-A Knowledge Repr. C127	CO.MSC-AI-A, CO.MSC-AI-B Natural Lang. Proc. B260	CO.MSC-AI-A, CO.MSC-AI-B Natural Lang. Proc. B214	CO.MSC-AI-B, CO.MSC-AI-A Knowledge Repr. B269
9:15							
9:30							
9:45							
10:00	CO.MSC-AI-B, CO.MSC-AI-A Metaheuristic Optim. C214				CO.MSC-AI-A, CO.MSC-AI-B Rsrch. Practice & Ethics B260	CO.MSC-AI-A, CO.MSC-AI-B Metaheuristic Optim. B225	CO.MSC-AI-B Knowledge Repr. C127
10:15							
10:30							
10:45							
11:00		CO.MSC-AI-A, CO.MSC-AI-B Big Data Processing D237		CO.MSC-AI-A, CO.MSC-AI-B Knowledge Repr. B260	CO.MSC-AI-A Prac. Mach. Learning C127	CO.MSC-AI-A, CO.MSC-AI-B Prac. Mach. Learning B149	
11:15							
11:30							
11:45							
12:00	CO.MSC-AI-A, CO.MSC-AI-B Rsrch. Practice & Ethics B149			CO.MSC-AI-A, CO.MSC-AI-B Prac. Mach. Learning B149		CO.MSC-AI-A, CO.MSC-AI-B Big Data Processing B149	
12:15							
12:30							
12:45							
13:00							
13:15							
13:30							
13:45							
14:00	CO.MSC-AI-B Metaheuristic Optim. C127			CO.MSC-AI-A Big Data Processing C127	CO.MSC-AI-A Metaheuristic Optim. C127	CO.MSC-AI-B Prac. Mach. Learning C127	
14:15							
14:30							
14:45							
15:00		CO.MSC-AI-A Rsrch.	CO.MSC-AI-B Rsrch.				
15:15							

Combinatorial problem example: Timetabling

- CS @ CIT (Rough numbers):
  - > 2000 campus students
  - > 300 Modules
  - 50 full time staff
  - > 50 rooms
- Example from https://www.unitime.org/uct_description.php
  - *Purdue is a large (39,000 students) public university with a broad spectrum of programs at the undergraduate and graduate levels. In a typical term there are 9,000 classes offered using 570 teaching spaces. Approximately 259,000 individual student class requests must be satisfied*
  - Problem decomposed into
    - a centrally timetabled large lecture room problem (about 800 classes being timetabled into 55 rooms with sizes up to 474 seats),
    - individually timetabled departmental problems (about 70 problems with 10 to 500 classes),
    - and a centrally timetabled computer laboratory problem (about 450 classes timetabled into 36 rooms with 20 to 45 seats).

Combinatorial problem example: Timetabling

(insertion) heuristic approach

- Repeat following step until all assigned:
  - Choose module+classgroup, choose time slot and choose room

- But how to choose?
  - Random?
  - Dedicated heuristic,
    - e.g. choose room with smallest non-negative value for (*capacity* – *classize*)
    - Start with variables that will be hard to satisfy, e.g. large classes that can only fit into a few rooms (compared to small classes that can fit in most rooms)

Combinatorial Problems

- Combinatorial Problems arise in many areas of computer science and applications domains:
  - Finding shortest / cheapest round trips (TSP)
  - Finding models of propositional formulae (SAT)
  - Planning, scheduling, time-tabling
  - Vehicle routing
  - Location and clustering
  - Internet data packet routing
  - Protein structure prediction

Optimization

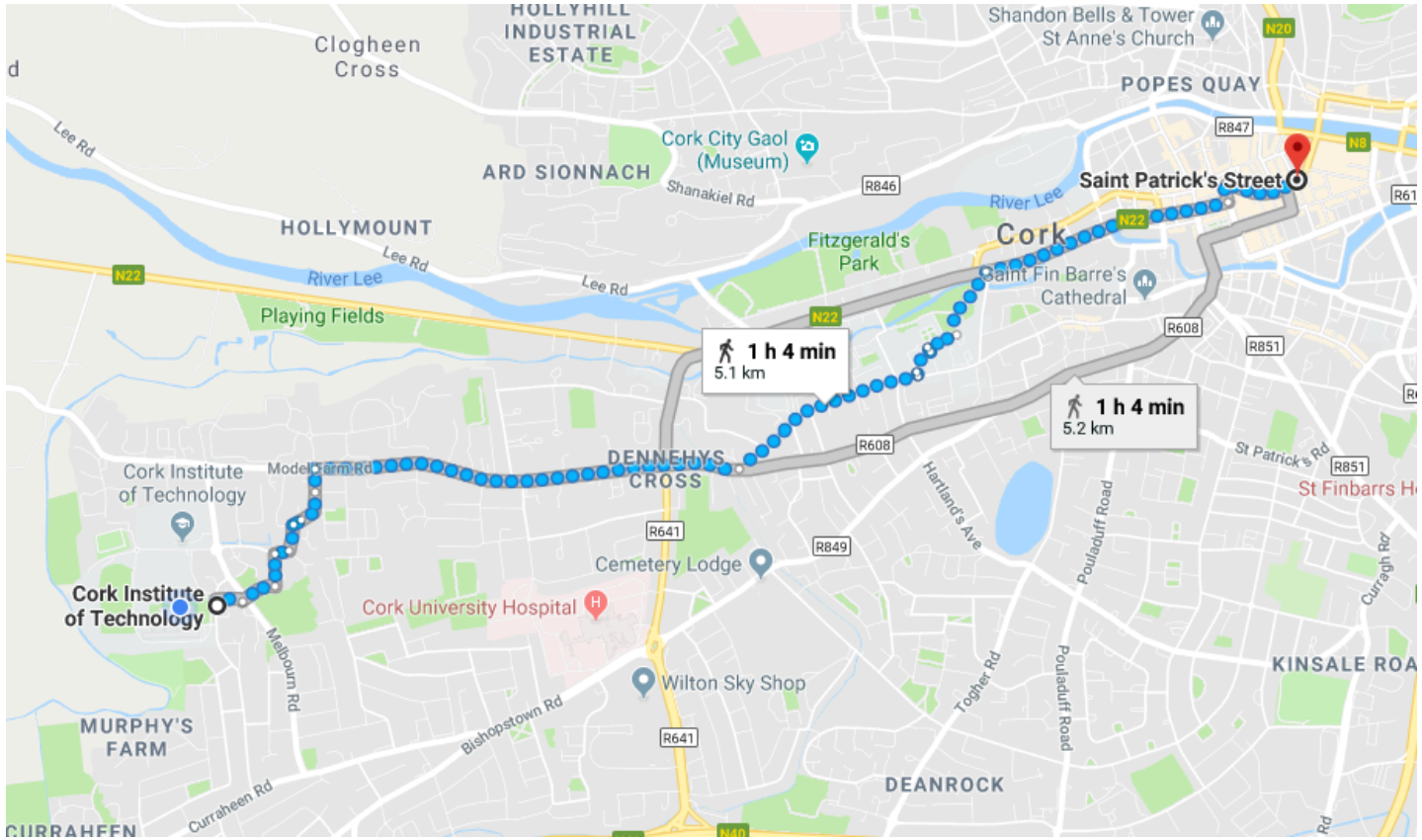
- Sudoku is a *Satisfaction* problem. Satisfaction problems are where we just want to find a solution, i.e. an assignment such that no constraint is violated.
- Optimization refers to choosing the ***best*** solution from some set of available alternatives, so we have a criteria that distinguishes amongst quality of solutions.
- Optimization problems:
  - Arise in wide variety of applications
  - Arise in many different forms, e.g., continuous, combinatorial, multi-objective, stochastic, etc.
  - Here we focus mainly on combinatorial problems
    - E.g. A cost function in timetabling problem penalizing classes with more than 2 hours in a row, gaps between classes, etc.
- Range from quite easy to hard ones
  - Here we focus on the hard ones

Easy example

Find the best (most valuable) element from the set of alternatives

A more difficult (but still “easy”) one

Find best (shortest) route from A to B in an edge-weighted graph

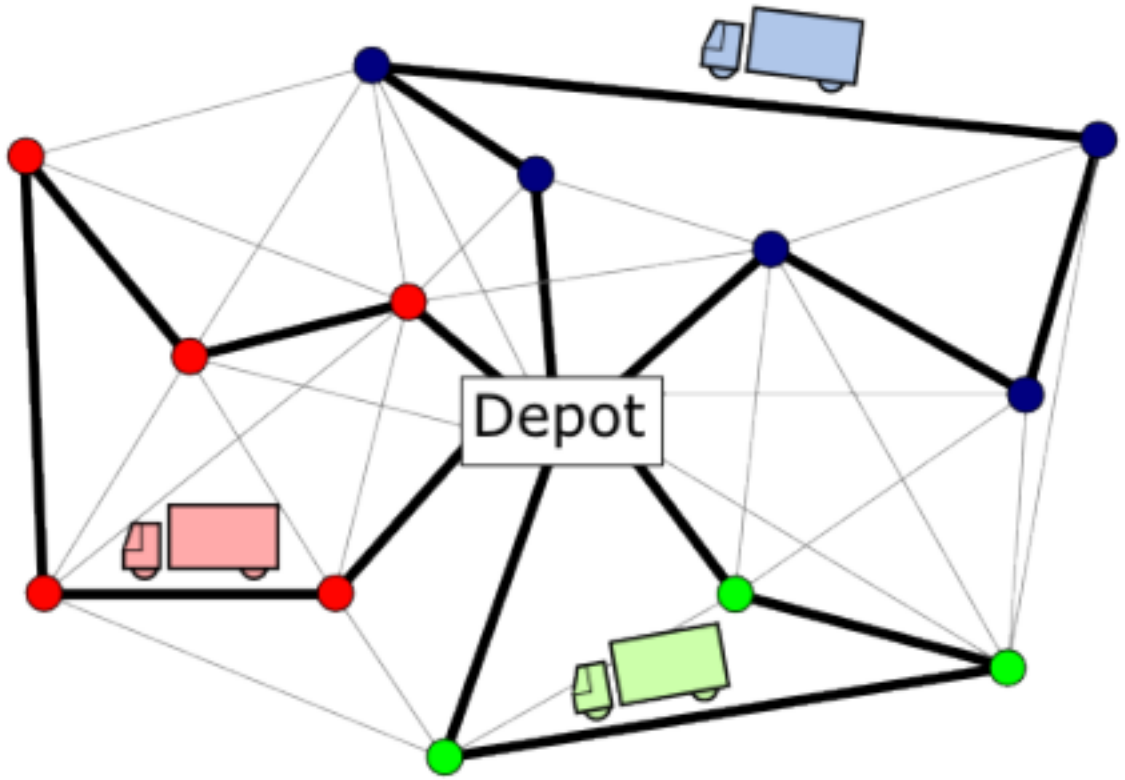


A harder one

Find best (shortest) round trip through some cities, aka Traveling Salesman Problem (TSP)

A real-life like problem

TSP arises as sub-problem, e.g., in vehicle routing problems (VRPs)



Combinatorial Problems

- Realistic problems can involve many complicating details
- Examples in VRP case are:
  - Time windows, access restrictions, priorities, split delivery, …
  - Capacity constraints, different cost of vehicles, …
  - Working time constraints, breaks, …
  - Stochastic travel times or demands, incoming new request, …
- We will focus on simplified models of (real-life) problems
  - Useful for illustrating algorithm principles
  - They are “hard” to capture essence of more complex problems
  - Are treated in research to yield more general insights

Simple Scheduling Example

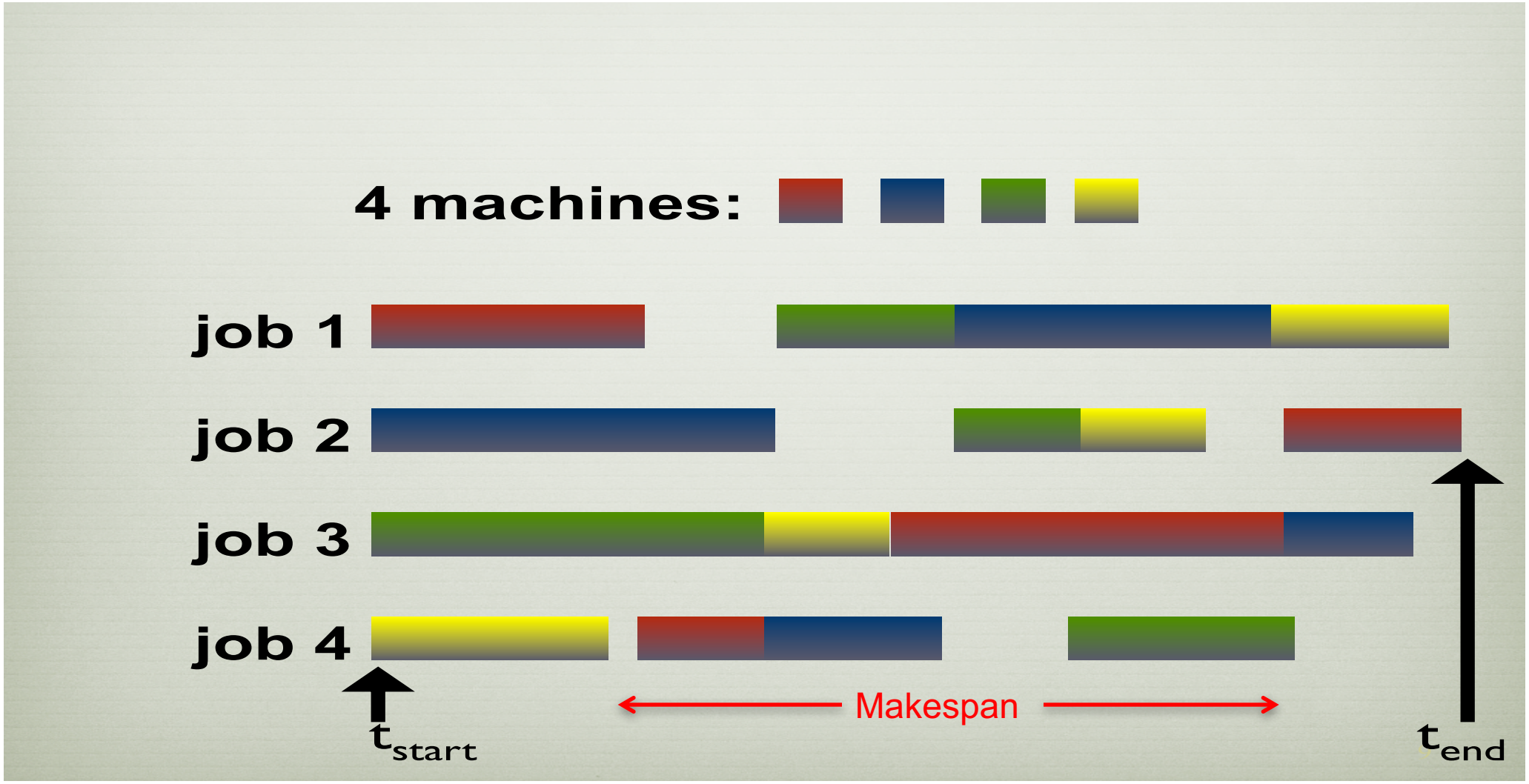
An *n x m* open shop problem (OSP):

- *n* jobs, *m* machines.
- *m* tasks per job.
- Resource (disjunctive) constraint on tasks of a job or machine:

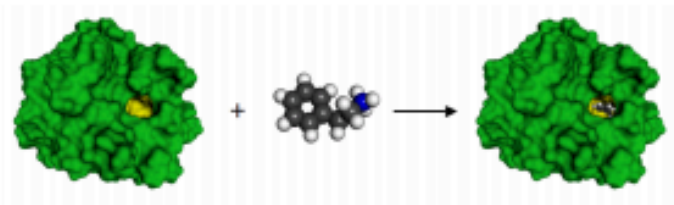
$$(t_i + p_i \leq t_j) \vee (t_j + p_j \leq t_i)$$

- Objective: Minimize the makespan.

Open Shop Scheduling



Optimization problems arise everywhere!



Most such problems are very hard (***NP-hard!***)

How to solve Combinatorial Problems?

- Many possible approaches
- Systematic enumeration is probably not realistic
- Some people may eliminate certain assignment or partial tours through careful reasoning
- Other intuitive approach: start with some good guess and then try to improve it iteratively

The latter is an example of heuristic approach to optimization

Solving (combinatorial) optimization problems

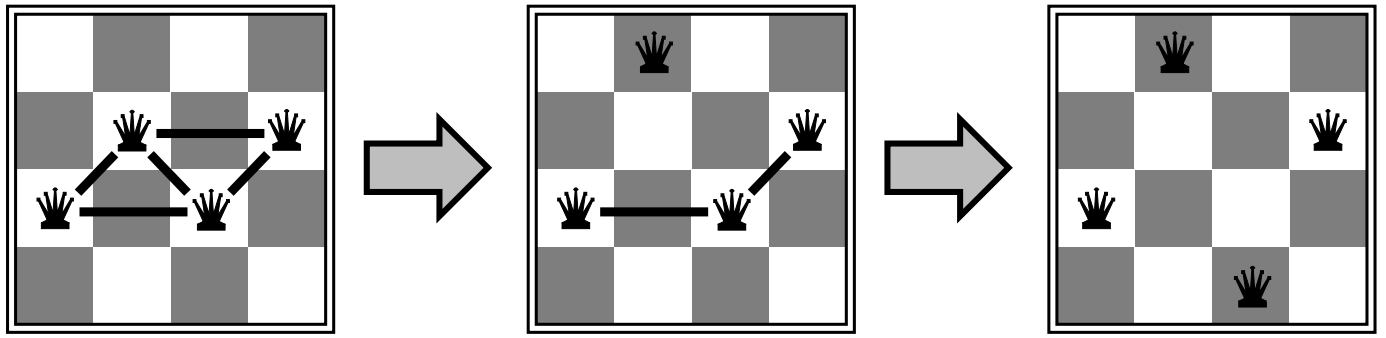
- Systematic enumeration
- Problem specific, dedicated algorithms
- Generic methods for exact optimization
- **Heuristic methods**

Heuristic example from Russel Norvig

Example: n-queens

Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal

Move a queen to reduce number of conflicts



h = 5 h = 2 h = 0

Almost always solves n-queens problems almost instantaneously for very large n, e.g., $n = 1million$

What is a heuristic?

- A heuristic is an alternative optimization methods able to determine not a perfectly accurate solution, but a set of good quality approximations to exact solutions.
- Heuristics, were initially based essentially on experts' knowledge and experience and aimed to explore the search space in a particularly convenient way

Heuristic methods

- **Heuristic methods** intend to compute efficiently, good solutions to a problem with no guarantee of optimality
- Range from rather simple to quite sophisticated approaches
- Inspiration often from
  - human problem solving
  - rules of thumb, common sense rules
  - design of techniques based on problem-solving experience
  - natural processes
    - evolution, swarm behaviors, annealing, ...
- usually used when there is no other method to solve the problem under given time or space constraints
- Often simpler to implement / develop than other methods

Main characteristics

- A heuristic is designed to provide better computational performance as compared to conventional optimization techniques, at the expense of lower accuracy
- The ***rules of thumb*** underlying a heuristic are often very specific to the problem under consideration
- Heuristics use domain-specific representations

What are Metaheuristics?

- The term metaheuristics was proposed by Glover at mid-80s as a family of searching algorithms able to define a high level heuristic used to guide other heuristics for a better evolution in the search space
- The most attractive feature of a metaheuristic is that its application requires no special knowledge on the optimization problem to solve

Fred W. Glover

This module

Provide Answers to these questions:

- Which metaheuristic methods are available and what are their features?
- How can metaheurists be used to solve computationally hard problems?
- How should heuristics methods be studied and analyzed empirically?
- How can heuristic algorithms be designed, developed, and implemented?