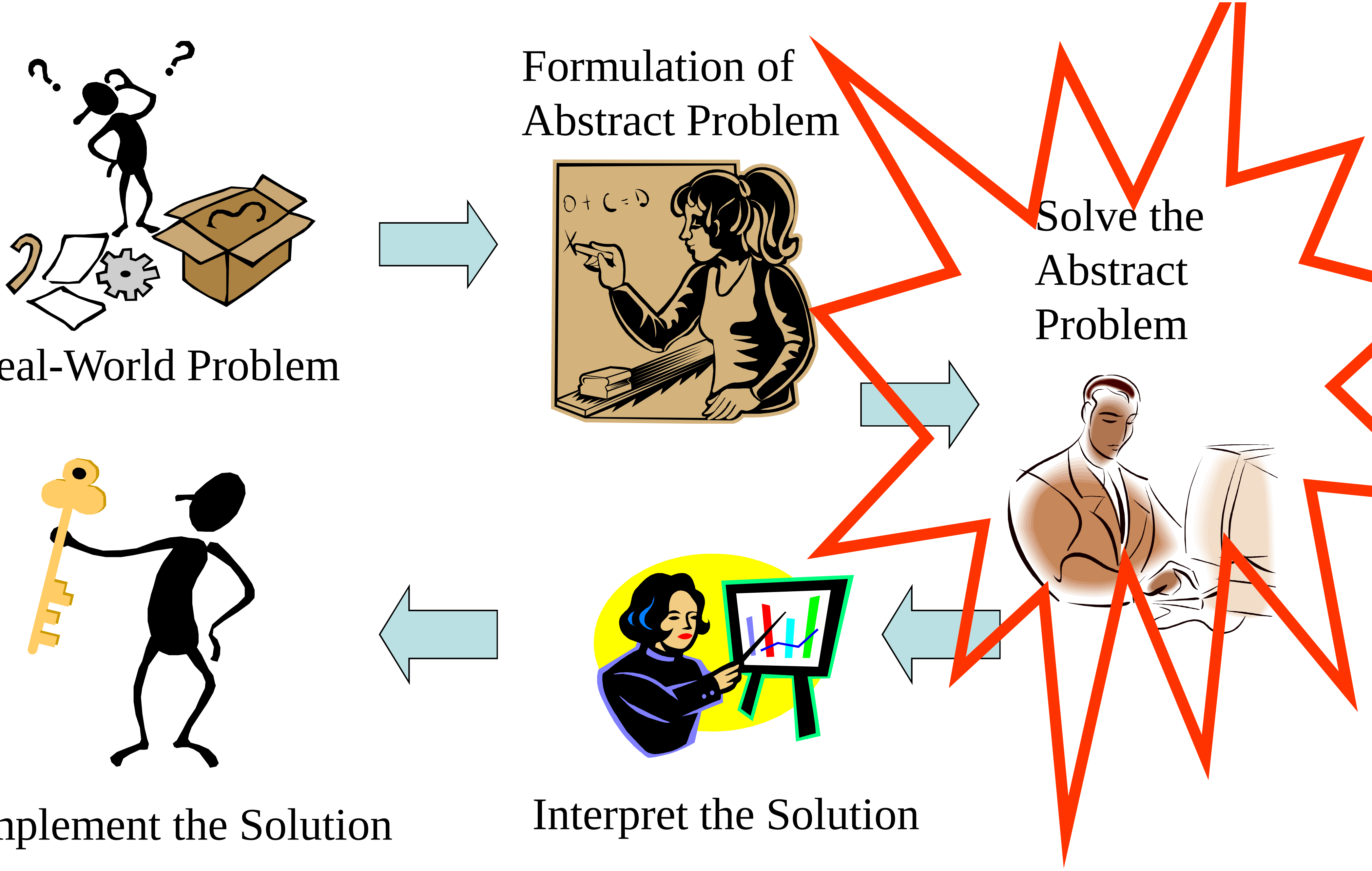
Meta-Heuristic Optimization Methods

Introduction

Metaheuristics

- Overview of, and introduction to, modern heuristic optimization methods that are suitable for solving practical optimization problems within logistics
- Elements
  - Combinatorial Optimization
  - Local Search
  - Heuristics
  - Local search based Metaheuristics
  - Population based Metaheuristics
  - Choice of methods for solving complex problems in a short time is central


Real-World Problem
Formulation of Abstract Problem
0 + C = D
Solve the Abstract Problem
Interpret the Solution
Implement the Solution

Motivation: Operations Research

- WW-II British military
- Scientists and engineers from several fields
- Analysis and Decision Support
  - Placement of Radar Stations
  - Guidance of convoys
  - Bombing raids
  - Anti-submarine campaigns
  - Laying of mines
  - Operational Analysis/Operations Research

Today: OR/Management Science

- Scientific approach to decision making regarding deciding on how a system is to be designed and operated, usually under conditions with limited resources
  - (Winston: Operations Research Applications and Algorithms)

Operations Research

- Quantitative methods for decision support
- Several disciplines
  - Mathematical modeling
  - Optimization
  - Probability theory
  - Game theory
  - Queuing theory
  - Simulation
- Discrete optimization problems (DOP) are central

Example: Choice of Projects

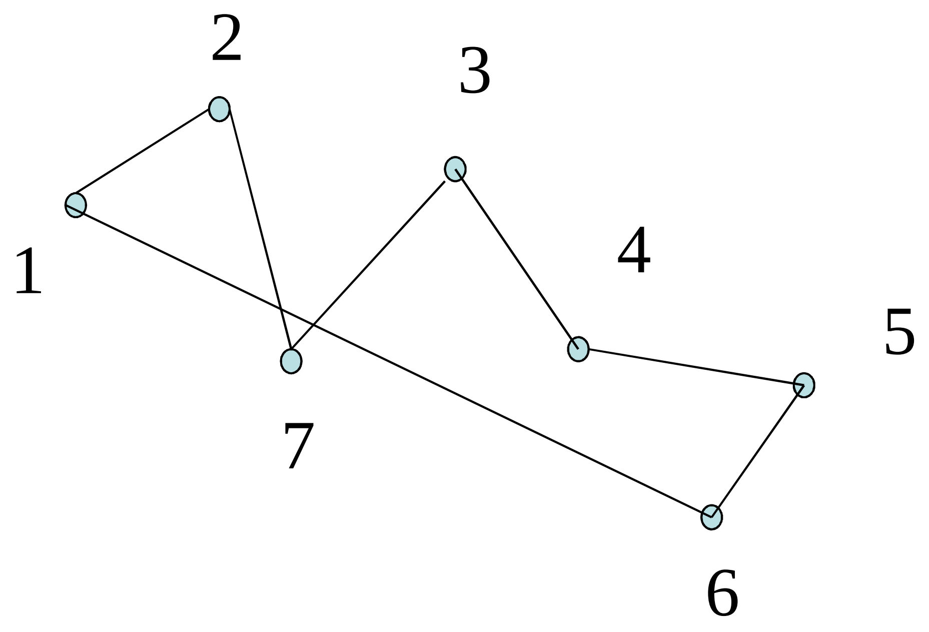
- You are leading a big company
- Your employees has suggested a number of projects to do, each with a known:
  - profit
  - cost
- You have a fixed budget
- Which projects should you select to maximize the profit?
- Strategic / tactical decision
  - Optimization problem
  - The Knapsack Problem

Example:Vehicle Routing

- You are running a transportation company with your own car
- You have a set of pickup and delivery jobs around town to do tomorrow
- What sequence of stops should you choose to finish the jobs as early as possible?

- Operative decision
  - Optimization problem
  - The Travelling Salesman Problem (TSP)

Exampel Optimization Problem: TSP – Travelling Salesman Problem



	1	2	3	4	5	6	7
1	0	**18**	17	23	23	23	23
2	2	0	88	23	8	17	**32**
3	17	33	0	**23**	7	43	23
4	33	73	4	0	**9**	23	19
5	9	65	6	65	0	**54**	23
6	**25**	99	2	15	23	0	13
7	83	40	**23**	43	77	23	0

Feasible Solution: 1 2 7 3 4 5 6 1 with value: 184

Abstract Problems (1)

- We will encounter several different abstract problems
  - Traveling Salesman Problem
  - Vehicle Routing Problem
  - (Multidimensional 0/1) Knapsack Problems
  - … and more
- They are all Combinatorial Optimization Problems (COPs)

What is a COP? (1)

- In a formal problem we usually find
  - Data (parameters)
  - Decision variables
  - Constraints
- The problem is typically to find values for the variables that optimize some objective function subject to the constraints
  - Optimizing over some discrete structure gives a Combinatorial Optimization Problem

What is a COP? (2)

- Can be expressed, very generally, as

$$\max_{x \in \mathcal{F} \subseteq \mathcal{S}} f(x)$$

- Where
  - $\boldsymbol{x}$ is a vector of decision variables
  - f($\boldsymbol{x}$) is the objective function
  - $\boldsymbol{\mathcal{S}}$ is the solution space
  - $\boldsymbol{\mathcal{F}}$ is the set of feasible solutions

What are LPs, IPs, and MIPs? (1)

- A **mathematical programming problem** refers to a problem where one tries to minimize (or maximize) a real function subject to some contraints
- In **L**inear **P**rogramming, the objective function and the constraints are all linear
- In **I**nteger (Linear) **P**rogramming, the variables can only take integer values
- In **M**ixed **I**nteger **P**rogramming, some variables must be integer and some can take real values

What are LPs, IPs, and MIPs? (2)

- Example of IP:

$$\max \sum_{j=1}^{n} c_j x_j$$

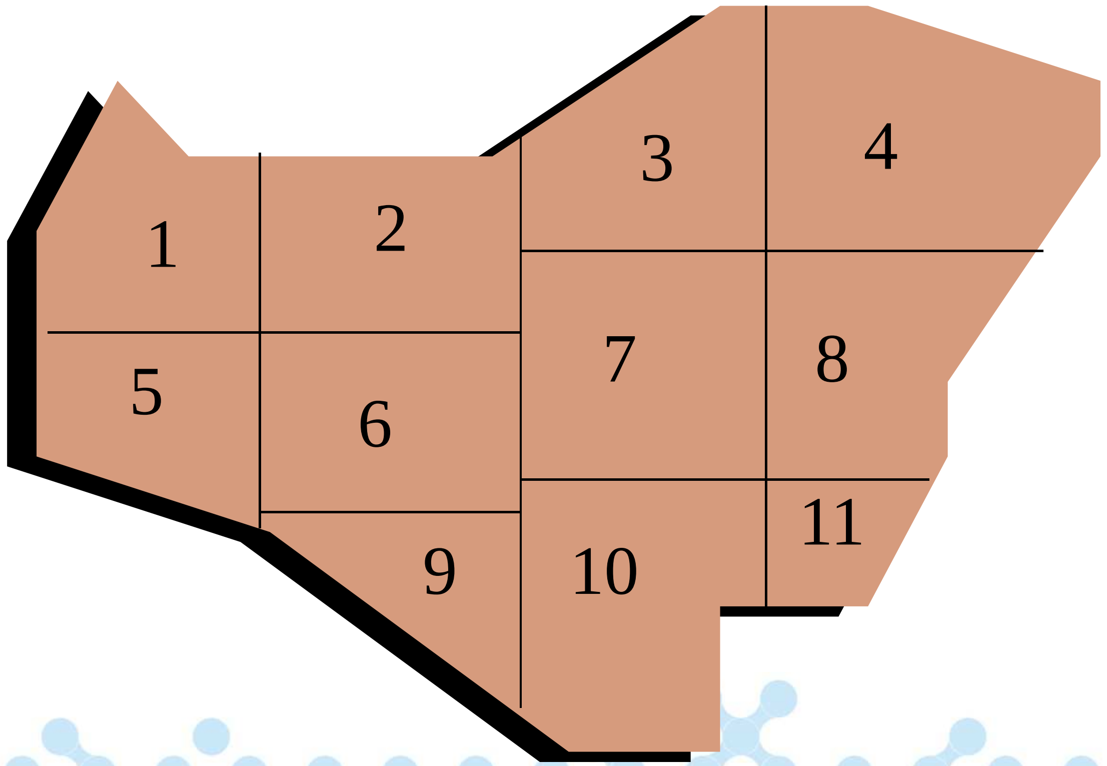
subject to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad , i = 1, \ldots, m$$

$$x_j \in \{0, 1, \ldots\} \qquad , j = 1, \ldots, n$$

Example: Service Centers (1)

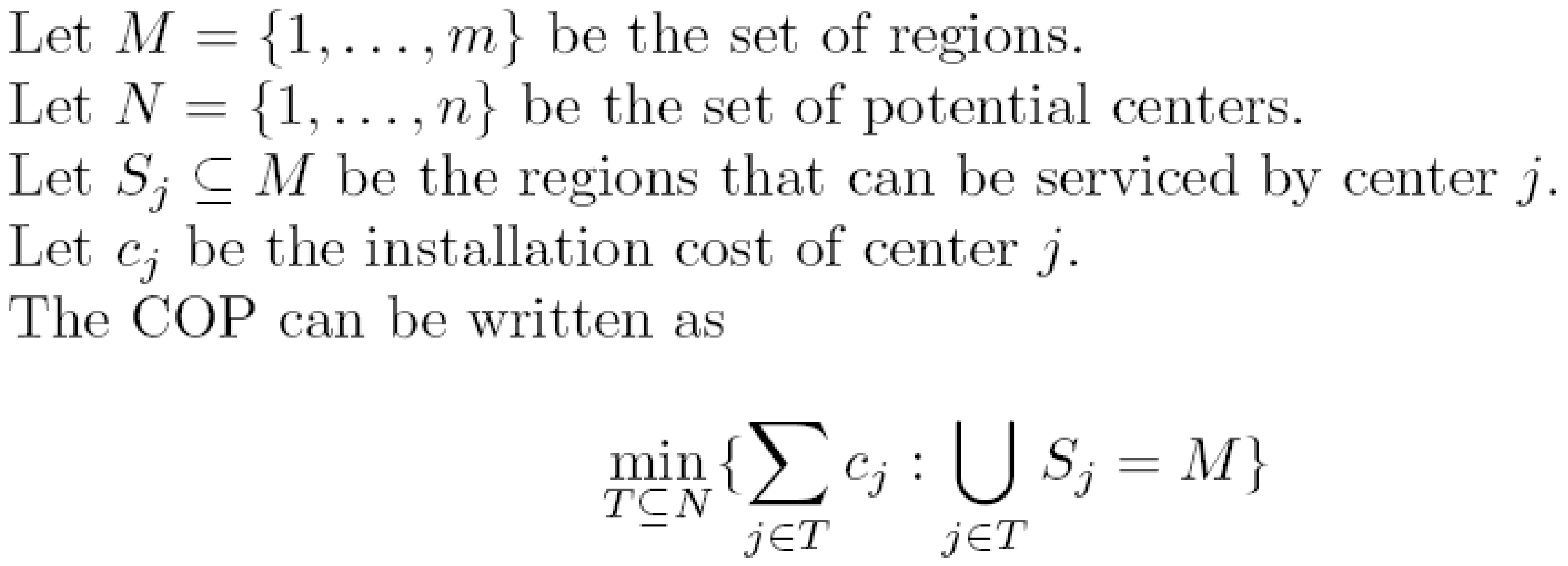
- Task: Install a set of emergency centers, while minimizing the cost and making sure that all regions are covered by at least one center



Center 1 covers regions 1, 2, 5, and 6.
Center 2 covers regions 5, 6, and 9.
Etc…

Example: Service Centers (2)

- Formulation as a COP:



Let $M = \{1, \ldots, m\}$ be the set of regions.
Let $N = \{1, \ldots, n\}$ be the set of potential centers.
Let $S_j \subseteq M$ be the regions that can be serviced by center j.
Let c_j be the installation cost of center j.
The COP can be written as

$$\min_{T \subseteq N} \{ \sum_{j \in T} c_j : \bigcup_{j \in T} S_j = M \}$$

Example: Service Centers (3)

- Formulation as an IP:
- First we need to specify the data

Let A be an incidence matrix,
i.e., $a_{ij} = 1$ if $i \in S_j$, and $a_{ij} = 0$ otherwise.

- Then we must define the decision variables

We have $x_j = 1$ if center j is selected, and $x_j = 0$ otherwise.

Example: Service Centers (4)

- Given data and variables, we can formulate the constraints:
  - At least one center must service each region:

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \text{ for } i = 1, \ldots, m$$

  - The variables are 0 or 1:

$$x_j \in \{0, 1\} \text{ for } j = 1, \ldots, n$$

Example: Service Centers (5)

- Including the objective function, the entire IP becomes:

$$\min \sum_{j=1}^{n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad , i = 1, \ldots, m$$

$$x_j \in \{0, 1\} \qquad , j = 1, \ldots, n$$

Example: Service Centers (6)

- This example illustrates how a formal problem may be defined based on a real world problem
- Gives an example where a COP is relatively easily formulated as an IP
  - Sometimes a COP can be difficult to formulate as a (Mixed) Integer Program!
  - We will not focus much on formulations in this course, however: we want to find **solutions**
  - The problems seen in this course will be **simple**

Terminology: Problems and Instances

- Exampel: TSP
- A *concrete* problem is an *instance*
- An *instance* is given by:
  - n: the number of cities
  - A: a n×n-matrix of travel costs
    (or travel distances or travel time)

Combinatorial Optimization Problem (1)

- Definition:
  - A Combinatorial Optimization Problem (COP) is specified by a set of problem instances
- A COP is either a minimization or maximization problem (can be interchanged)

Combinatorial Optimization Problem (2)

- A COP-instance is a pair ($\mathcal{S}$, f)
  - $\mathcal{S}$ is the set of usable (feasible) solutions (implicit)
  - f: $\mathcal{S}\rightarrow\mathcal{R}$ is the cost function
- The goal is to find a globally optimal solution:

$$i^* \in S : f(i^*) \le f(i), \forall i \in S$$

$$f^* = f(i^*)$$ (globally) optimal cost

$$S^* = \left\{ i \in S : f(i) = f^* \right\}$$ (globally) optimal solutions

Example 1: A TSP Instance

3 cities: 1, 2 ,3

$(\mathbf{S}, f)$ $\qquad \min_{s \in \mathbf{S}} f(s)$

	1	2	3
1	0	15	32
2	13	0	3
3	2	17	0

$$S = \{(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)\} \equiv \{s_1,\ldots,s_6\}$$

$$f(s_1) = 15 + 3 + 2 = 20$$

$$\ldots$$

$$f(s_6) = 17 + 13 + 32 = 62$$

Example 2: Knapsack Problem Instance

Knapsack with capacity 101, 10 ”items” (e.g. projects, ...) 1,...,10

$(\mathbf{S}, f)$

$\max_{s \in \mathbf{S}} \quad f(s)$

	1	2	3	4	5	6	7	8	9	10
Value	79	32	47	18	26	85	33	40	45	59
Size	85	26	48	21	22	95	43	45	55	52

$$S^+ = \{0000000000, \ldots, 1111111111\} \equiv \{s_1, \ldots, s_{1024}\}$$

S = those solutions from S+ that are feasible (given implicitly by the budget constraint)

$$f(s_1) = 0$$

$$f(s_{530}) = 117$$

$$f(s_{1024}) = 464$$

$$f^* = f(s_{530}) = 117$$

$$S^* = \{0100100001\}$$

Comments about COPs

- S is seldom given explicitly
  - Can, e.g., be given by constraints in an IP-formulation
- S is often a subset of feasible solutions in a larger space of possible solutions (feasible + infeasible)
- Often a compact representation of the instance
- A solution is often given by the values of the decision variables $(x_1,\ldots, x_n)$
- Often polynomial time algorithms for checking membership in S and the cost/value for candidate solutions

Example of a COP: TSPTW

- Travelling Salesman Problem with Time Windows (TSPTW)
- Complete graph with n nodes (cities)
- Known travel cost between cities c_{ij}
- Every city has service time windows $[e_i, l_i]$
- The salesman shall make a tour
- A solution can be represented by a permutation π
- S is the set of all (feasible) permutations
- f is the sum of the travel costs $f(\pi) = \sum_{i=1}^{n-1} c_{\pi(i),\pi(i+1)} + c_{\pi(n),\pi(1)}$

Applications of COP

- Decision problems with discrete decisions
- Synthesis problems
  - planning
  - design
- Operations Research, AI
- Logistics, design, planning, robotics

How can we solve problems?

- It can sometimes be advantageous to distinguish between three groups of methods for finding solutions to our abstract problems

- (Exact) Algorithms
- Approximation Algorithms
- Heuristic Algorithms

(Exact) Algorithms

- An algorithm is sometimes described as a set of instructions that will result in the solution to a problem when followed correctly
- Unless otherwise stated, an algorithm is assumed to give the optimal solution to an optimization problem
  - That is, not just a **good** solution, but the **best** solution

Approximation Algorithms

- Approximation algorithms (as opposed to exact algorithms) do not guarantee to find the optimal solution
- However, there is a bound on the quality
  - E.g., for a maximization problem, the algorithm can guarantee to find a solution whose value is at least half that of the optimal value
- We will not see many approximation algorithms here, but mention them as a contrast to heuristic algorithms

Heuristic Algorithms

- Heuristic algorithms do not guarantee to find the optimal solution, however:
- Heuristic algorithms do not even necessarily have a bound on how bad they can perform
  - That is, they can return a solution that is arbitrarily bad compared to the optimal solution
- However, in practice, heuristic algorithms (heuristics for short) have proven successful
- Most of this course will focus on this type of heuristic solution method

What is a heuristic?

- From greek *heuriskein* (meaning ”to find”)
- Wikipedia says:
  - (…)A heuristic is a technique designed to solve a problem that ignores whether the solution can be proven to be correct, but which usually produces a good solution (…).
  - Heuristics are intended to gain computational performance or conceptual simplicity, potentially at the cost of accuracy or precision.

Why don’t we always use exact methods?

- If a heuristic does not guarantee a good solution, why not use an (exact) algorithm that does?
- The running time of the algorithm
  - For reasons explained soon, the running time of an algorithm may render it useless on the problem you want to solve
- The link between the real-world problem and the formal problem is weak
  - Sometimes you cannot properly formulate a COP/IP that captures all aspects of the real-world problem
  - If the problem you solve is not the right problem, it might be just as useful to have one (or more) heuristic solutions, rather than the optimal solution of the formal problem

P vs **NP** (1)

- Computational complexity is sometimes used to motivate the use of heuristics
- Formal **decision** problems are divided into many classes, but two such classes are
  - **P** (**P**olynomial)
    - Includes problems for which there exist algorithms that have a running time that is a polynomial function of the size of the instance
  - **NP** (**N**ondeterministic **P**olynomial)
    - Includes problems for which one can **verify** in polynomial time that a **given solution** is correct

P vs **NP** (2)

- We believe that **P** is different from **NP**, but nobody has proven this
  - $1 000 000 awaits those that can prove it
  - (note that all of **P** is contained in **NP**)
- Some optimization problems can be solved in polynomial time
  - If you have a graph with *n* nodes, the shortest path between any two nodes can be found after f(*n*) operations, where f(*n*) grows as quickly as n^2
  - If you want to distribute *n* jobs among *n* workers, and each combination of job and worker has a cost, the optimal assignment of jobs can be found in g(*n*) operations, where g(*n*) grows as quickly as n^3

P vs **NP** (3)

- However, for some optimization problems we know of no polynomial time algorithm
  - Unless **P**=**NP** there are none!
- Sometimes, finding the optimal solution reduces to examining **all** the possible solutions (i.e., the entire solution space)
  - Some algorithms do **implicit enumeration** of the solution space (but this sometimes reduces to examining all solutions)
- So, how many solutions must we examine in order to find the optimal solution?

The Combinatorial Explosion (1)

- The number of possible solutions for different problems:
  - The Set Covering Problem:
    - (Which includes our emergency center problem!)
    - The number of centers is *n*
    - The number of solutions: 2^n
  - The Traveling Salesman Problem:
    - A salesman must travel between n cities, visiting each once. The salesman can visit the cities in any order
    - The number of solutions:

$\frac{1}{2}*(n-1)!$

The Combinatorial Explosion (2)

- The combinatorial explosion refers to the fact that some functions (such as those that result as the number of solutions for some hard optimization problems) increase **very** quickly!

n	**n^2**	**n^3**	**2^n**	**1/2(n-1)!**
10	100	1000	1024	181440
100	10000	1000000	1.27E+30	4.7E+155
1000	1000000	1E+09	1.1E+301	#NUM!

- How would you solve a Traveling Salesman Problem with 100 customers?

A Small Note on the TSP

- Although the number of solutions of the TSP grows very quickly, surprisingly large instances has been solved to optimality
  - A TSP has been solved for 24978 cities in Sweden (the length was about 72500 kilometers)
  - A TSP for 33810 points on a circuit board was solved in 2005
    - It took 15.7 CPU-years!
- However, we also have heuristic methods that can quickly find solutions within 2-3% of optimality for problems with millions of cities!

Summary

- Some practical information
  - Use the web page for lecture slides etc…
- About formal problems
  - Formulations: COP, IP, MIP, …
  - Problems: TSP, Set Covering, …
- How to solve problems
  - Exact-, Approximation-, Heuristic algorithms
- Why use heuristics?
  - Complexity (**P** vs **NP**), combinatorial explosion