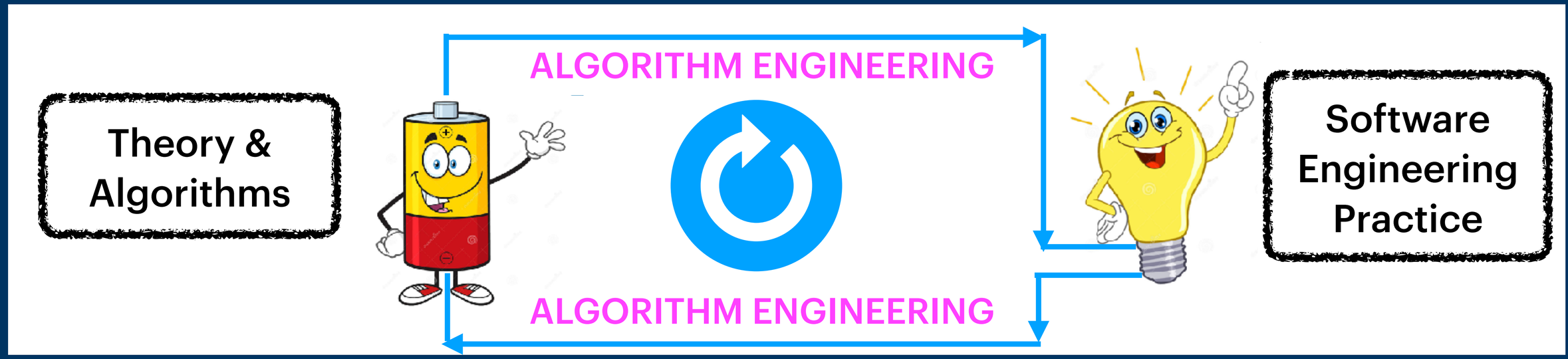


ALGORITHM ENGINEERING

Lecture 2:
Modeling from Algorithm Engineering Perspective

M. Oğuzhan Külekci - kulekci@itu.edu.tr

Modeling

Given the real-world problem, map it to a more general problem class covering the details.

“Modeling is the procedure of abstracting from actual problem instances at hand to problem classes that still contain the essential details of the problem structure”

*** We will be using excerpts and figures from the chapter 2 of the book Müller-Hannemann, M., & Schirra, S. (2001). Algorithm Engineering. Springer.*

Modeling

PRACTICE:
Abstracting is a challenge

THEORY:
Problems are already enough abstracted

- Not oversimplified nor drown in unnecessary details !
- Capture all essentials but avoid less important aspects that are not helpful for the solution.
- For a good model, good knowledge of the particular area is essential.
- Modeling is not free from the algorithmic design.

1. Understand and formalize the problem
2. Try to match a “solution approach model”
3. Investigate how well the model fits to our case

Modeling

How would you tell the project you are working on to another CS professional to take some advise or opinion ?

- What is the application itself ?
- What is the algorithmic problem derived from the application ?
- What is the input data and also the requirements ?
- What solutions have been previously proposed ?
- What should be improved ?
- How this goal can be achieved ?

By experience, it is not easy and hard to communicate in this setting.

*Much easier without **<u>unnecessary</u>** details and a more abstract definition.*

Such a definition is actually modeling our real-world problem.

	5		9		6		3	
4	6		5		1		7	8
			4		7			
3	2	1				7	4	9
7	4	6				8	5	1
			8		2			
2	7		1		5		8	6
	8		3		9		2	

SUDOKU task is to complete the empty cells such that each number occurs exactly once in each row, column, and 3-by-3 subsquare.

Modeling - Fundamental Concepts

- Requirement analysis
- Closest problem class *decision, counting, construction, optimization, etc…*
  - Informal : Customer meeting
  - Semi-formal : Description with some formal concepts, sets, graphs, etc…
  - Formal : The abstract definition of the problem
- Definition of **variables**, **parameters** and **constraints**

Modeling - Problem Specification

Variables, parameters, constraints…

- John spends 30 minutes, while Bill needs 15 to produce an item. How many items can be produced in 8 hours by John and Bill?
- *There are n workers. Each produce an item within a specific time. How many items can be produced by their joint work in a queried time interval t ?*
  - w_i : Time to produce a single item by worker i. (Extracted from the given problem)
  - x_i : Number of items produced per worker i. (Introduced to solve the problem)
  - T : Queried time duration (Given constraint / restriction in the problem)
  - $x_i = \frac{T}{w_i}$, and the answer is $X = \sum x_i$ (The relation between the variables/parameters and the final output)
- Many other aspects may need to be consider in real life, worker performance cannot be constant …

Variable: A certain decision possibility.

Parameter: A value or property of the problem instance, which is used as an **abstraction** in the model.

Constraint: An abstract description of a given requirement

THINK ABOUT WHAT YOU SHOULD DECLARE WITH A #DEFINE PRAGMA or WITH VARIABLE DEFINITIONS IN C/C++

Modeling - Problem Analysis

- Input
- Output
- Constraints (hard or soft)
- Classification of parameters/variables (important, less important…)
- Certain problem class or solution approach

Vague and imprecise data handling:

Possible imperfections on input data and expectations on the change of parameters (frequent in real-life)

An example case

Modeling the SUDOKU game

$N = \{1,2,3,4,5,6,7,8,9\}$ $N' = \{1,2,3\}$

A cell can be addressed by (rowID, colID) as $\langle i, j \rangle$ and its value via $k = f(i, j)$. Thus $f : N^2 \rightarrow N$

The aim is to have all $f(i, j)$ values such that :

1. $\{f(1,j) : j \in N\} = N$
2. $\{f(i,1) : i \in N\} = N$
3. $i, j \in N, \ \{f(3(i-1)+i', 3(j-1)+j') : (i', j') \in N \times N\} = N$

	5		9		6		3	
4	6		5		1		7	8
			4		7			
3	2	1				7	4	9
7	4	6				8	5	1
			8		2			
2	7		1		5		8	6
	8		3		9		2	

SUDOKU task is to complete the empty cells such that each number occurs exactly once in each row, column, and 3-by-3 subsquare.

[Submitted on 1 Jan 2012 (v1), last revised 1 Sep 2013 (this version, v2)]

There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem

Gary McGuire, Bastian Tugemann, Gilles Civario

The sudoku minimum number of clues problem is the following question: what is the smallest number of clues that a sudoku puzzle can have? For several years it had been conjectured that the answer is 17. We have performed an exhaustive computer search for 16-clue sudoku puzzles, and did not find any, thus proving that the answer is indeed 17. In this article we describe our method and the actual search. As a part of this project we developed a novel way for enumerating hitting sets. The hitting set problem is computationally hard; it is one of Karp's 21 classic NP-complete problems. A standard backtracking algorithm for finding hitting sets would not be fast enough to search for a 16-clue sudoku puzzle exhaustively, even at today's supercomputer speeds. To make an exhaustive search possible, we designed an algorithm that allowed us to efficiently enumerate hitting sets of a suitable size.

Modeling a Solution Approach

- **Step 1: Think about the constraints and input**
  - What should be considered and how they effect the solution?
  - Possibilities on pre/post processing
  - A solution model, where all constraints can be identified properly
- **Step 2. Observe and analyze the properties**
  - Subproblems
  - Exact or approximate answer
  - How much time is reasonable to solve the problem
  - How about scaling
  - Requested implementation time

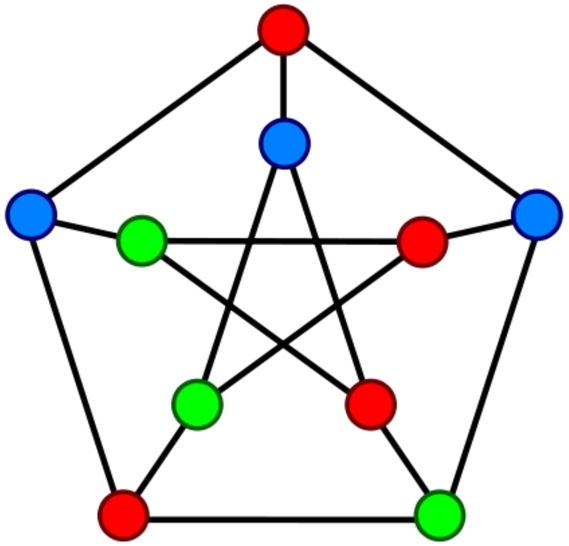
Pay attention to :

- Simplicity
- Existing solutions
- Implementation complexity
- Time line of the project
- Costs
- Patents

An example case

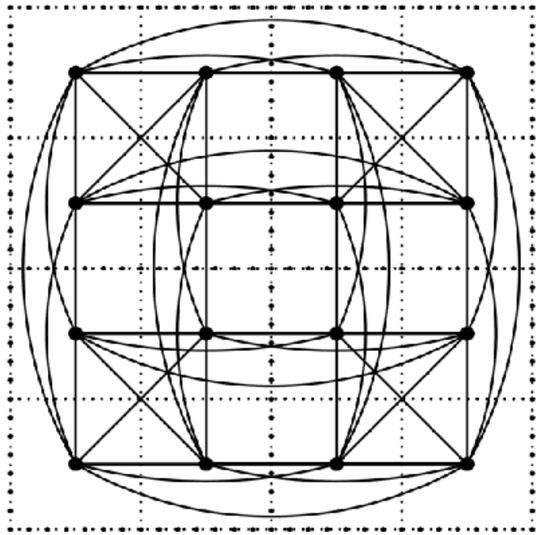
Solution approaches for the SUDOKU

It seems like a graph problem can be thought (like in many hard problems…)



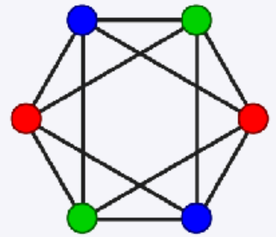
Vertex Coloring Problem: Assign colors to vertices such that no edge connects two vertices of the same color

- Create 81 vertices corresponding to each cell.
- Create the connections properly.
- 9-colorable assignment ?



(a) Sudoku as a vertex coloring problem.

Graph coloring

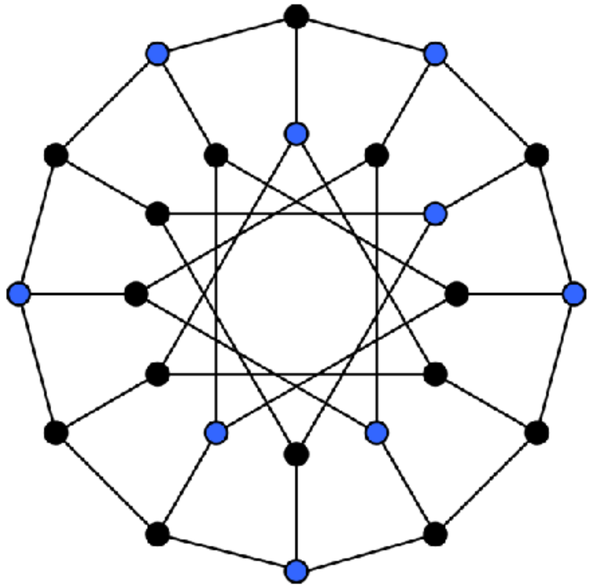


Decision	
Name	Graph coloring, vertex coloring, k-coloring
Input	Graph G with n vertices. Integer k
Output	Does G admit a proper vertex coloring with k colors?
Running time	$O(2^n n)$[10]
Complexity	NP-complete
Reduction from	3-Satisfiability
Garey–Johnson	GT4
Optimisation	
Name	Chromatic number
Input	Graph G with n vertices.
Output	$\chi(G)$
Complexity	NP-hard
Approximability	$O(n(\log n)^{-3}(\log\log n)^2)$
Inapproximability	$O(n^{1-\varepsilon})$ unless P = NP
Counting problem	
Name	Chromatic polynomial
Input	Graph G with n vertices. Integer k
Output	The number $P(G,k)$ of proper k-colorings of G
Running time	$O(2^n n)$
Complexity	#P-complete
Approximability	FPRAS for restricted cases
Inapproximability	No PTAS unless P = NP

An example case

Solution approaches for the SUDOKU

It seems like a graph problem can be thought (like in many hard problems…)



- Create 81 vertices corresponding to each cell.
- Create the connections properly.
- Find 9 independent sets and assign a number to all vertices on the same set.

Independent Set Problem: Find subsets of vertices such that no two vertices on the same set has an edge between.

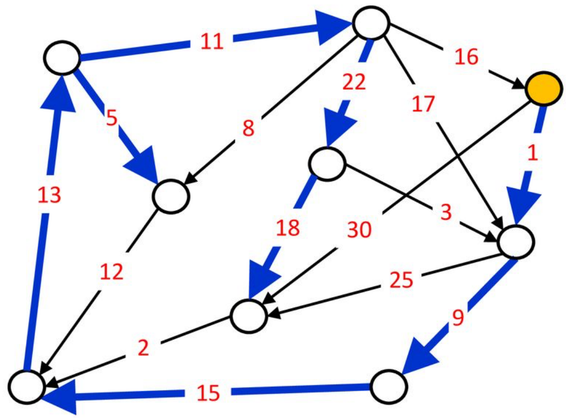
Other models can be surely defined…

Another example for modeling

Compression Network

- Given a repository of n files, compress the repository as much as possible!
- **Relative compression** helps, so make use of it…
- Given n files (or data partitions) create a network of them such that the vertices are the files and an edge from one vertex A to another vertex B indicate how much file B can be compressed relative to A.

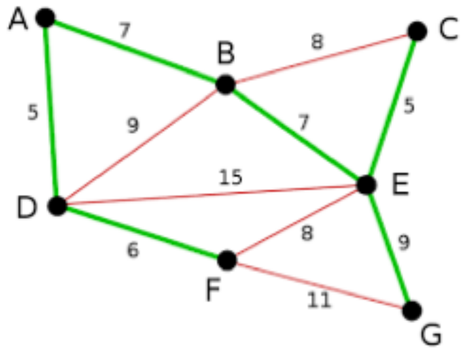
Directed minimum spanning trees



3

MINIMUM SPANNING TREE ???

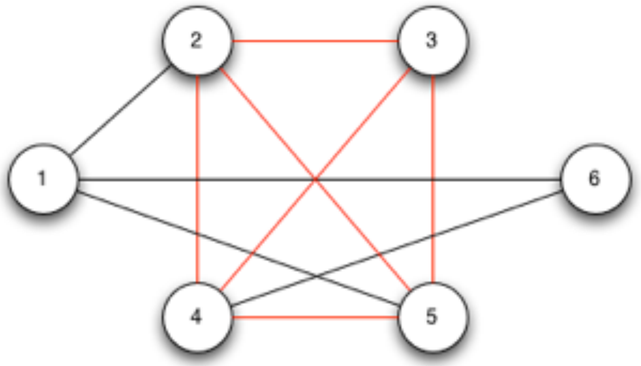
Undirected minimum spanning tree



Another example for modeling

Compression Network

- Despite MST, vertex-cover, shortest path, maximum flow, etc… may work ?



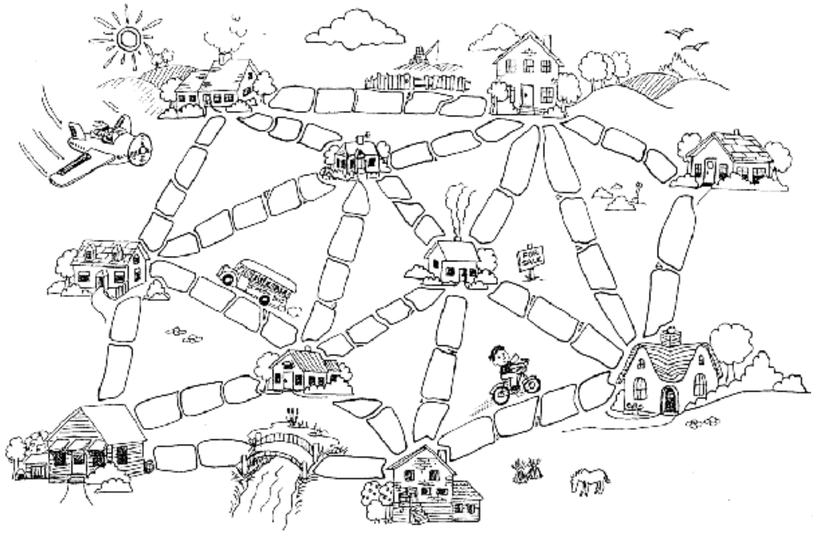
Some challenges :

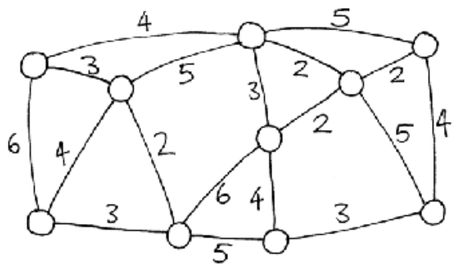
- Heuristics to decrease the quadratic complexity of relative compression ratios
- Finding the best path, cycle, must, …
- Practical requirements, e.g., extract a file with less than k file traversals
- Scalability, implementation etc…

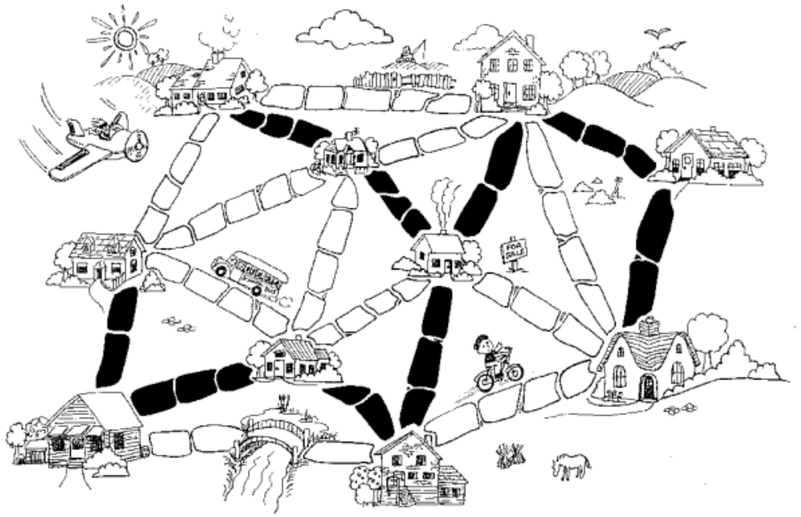
Yet another example

Muddy City problem

- Which roads you prefer to asphalt first in a city with limited budget ?







Modeling Frameworks

- Graph models
- Mixed integer programming (Equations of linear/integer/binary variables)

$$
\begin{aligned}
\min \quad & c_1^T x + c_2^T y \\
\text{s.t.} \quad & A_{11} x + A_{12} y = b_1 \\
& A_{21} x + A_{22} y \leq b_2 \\
& x \in \mathbb{R}^{n_1} \\
& y \in \mathbb{Z}^{n_2}
\end{aligned}
$$

Choose x from n1 real-numbers and y from n2 integers

- Constraint programming

$$
\begin{aligned}
x_{ij} &\neq x_{ik} && \forall k \neq i, j \in N, && \text{(rows)} \\
x_{ij} &\neq x_{kj} && \forall k \neq i, j \in N, && \text{(columns)} \\
x_{i_1 j_1} &\neq x_{i_2 j_2} && \forall (i_1, j_1) \neq (i_2, j_2) \in C_{ij}, \forall i, j \in N', && \text{(subsquares)} \\
x_{ij} &\in N && \forall i, j \in N. &&
\end{aligned}
$$

- Algebraic Modeling Languages

Modeling with Respect to Computing Architecture

Packed String Matching example

- SIMD is a hidden treasure…
- Vectorization helps in many cases, but string matching ???
- SSEF/EPSM algorithms…
- Let me tell you the story regarding how to create a solution approach by considering the computing architecture….

NEXT WEEK ?

- Choose a real-life problem and post it as a reply to my message on Ninova until Thursday to avoid people studying the same problem
- Study it and propose a model and a solution approach
- Write a report of 1-2 pages that describe your problem and model
- Present it in the class within 10 minutes
- Submit your report and presentation before the next class