Introduction to Combinatorial Optimization

Zdeněk Hanzálek
zdenek.hanzalek@cvut.cz
Acknowledgments to: Přemysl Šůcha, Jan Kelbel
Jiří Trdlička, Zdeněk Baumelt, Roman Čapek

CTU in Prague

February 15, 2022

Table of Contents

Lectures

1. Introduction of Basic Terms, Example Applications.
2. Integer Linear Programming - Algorithms.
3. Problem Formulation by Integer Linear Programming.
4. Shortest Paths.
5. Problem Formulation by Shortest Paths.
6. Flows and Cuts - Algorithms and Problem Formulation.
7. Bipartite Matching. Multi-commodity Network Flows.
8. Knapsack Problem, Pseudo-polynomial and Approximation Algs.
9. Traveling Salesman Problem and Approximation Algorithms.
10. Mono-processor Scheduling.
11. Scheduling on Parallel Processors.
12. Project Scheduling with Time Windows.
13. Constraint Programming.

1 - motivation; 4,5,6,7 - mostly polynomial complexity
8,9,10,11,12 - NP-hard problems, 2,3,13 - declarative programming
Test 0, Test I, and Test II are related to the lectures.

Labs

- Five programming exercises
- Programming test related to exercises
- Semester Project
  - CoContest
  - Individual Research Project
- Credits

Course Organization

https://cw.fel.cvut.cz/wiki/courses/ko/start

- Courses
- Labs
- Project
- Classification
- Literature

What is Combinatorial Optimization?

Optimization is a term for the mathematical discipline that is concerned with the minimization/maximization of some objective function subject to constraints or decision that no solution exists.

Combinatorics is the mathematics of discretely structured problems.

Combinatorial optimization is an optimization that deals with discrete variables.

It is very similar to **operation research** (a term used mainly by economists, originated during WW II in military logistics).

Application Areas

Many real-life problems can be formulated as combinatorial optimization problems.

Typical application areas:

- Production (production speed up, cost reduction, efficient utilization of resources...)
- Transportation (fuel saving, reduction of delivery time...)
- Employees scheduling (reduction of human resources...)
- Hardware design (acceleration of computations...)
- Communication network design (end-to-end delay reduction...)
- ...

This course will help you to understand and solve such problems:

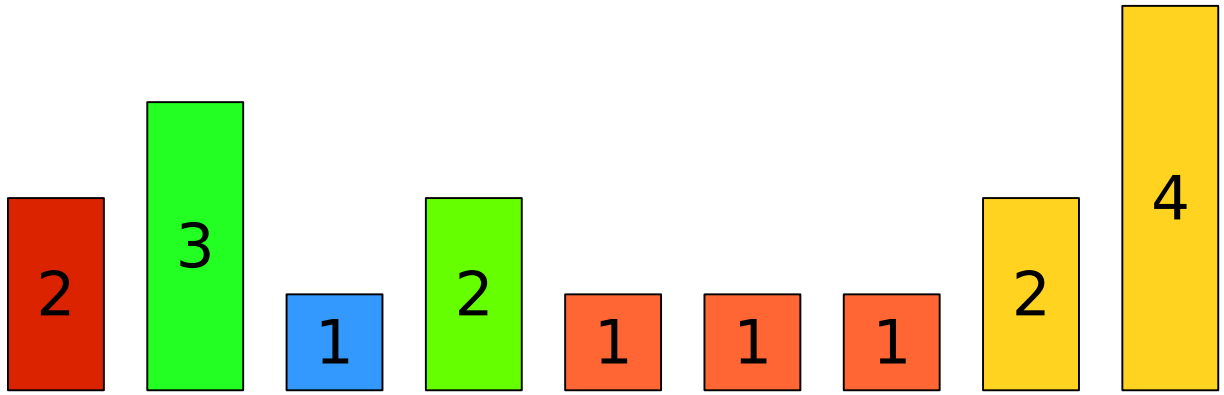
- Formalization - transformation to a known optimization problem
- Algorithms - the problem solution

Bin Packing Problem

Indivisible objects of different size and bins of equal capacity.
Goal:

- Assign the objects to the bins, minimize the number of bins.

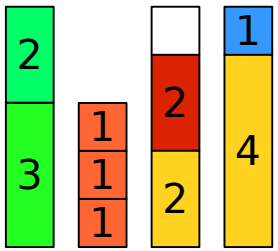
Example



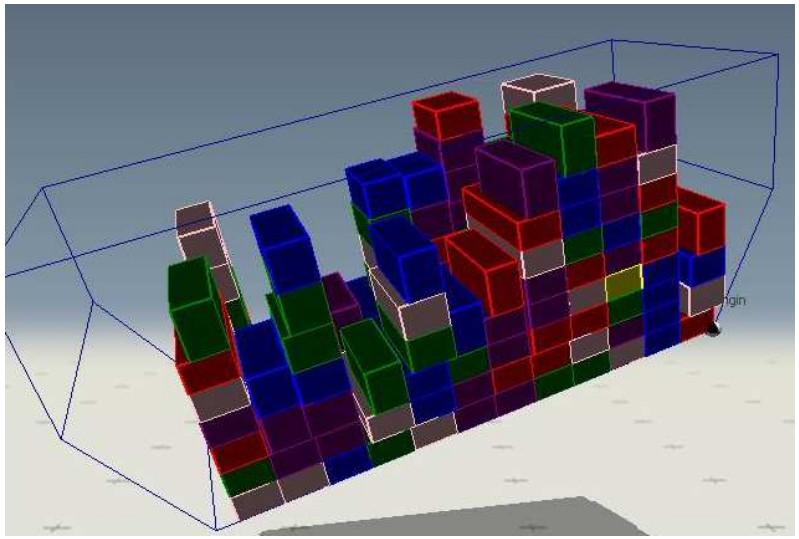
Different versions of the problem:

- Bins of different sizes $\{3, 5\}$
- Minimize the number of bins for each color

Result:



Container Loading



Goal:

- To store as much boxes as possible in a container.

Constraints:

- size of the container
- sizes of the boxes
- loading process
- stability, orientation of the boxes
- requested order of the boxes when unloading the cargo

Similar problems: **2-D cutting**, **Guillotine 2-D cutting**.

Assignment of Shifts to Employees

Goal:

- create acceptable shift roster,
  so that all required shifts are assigned

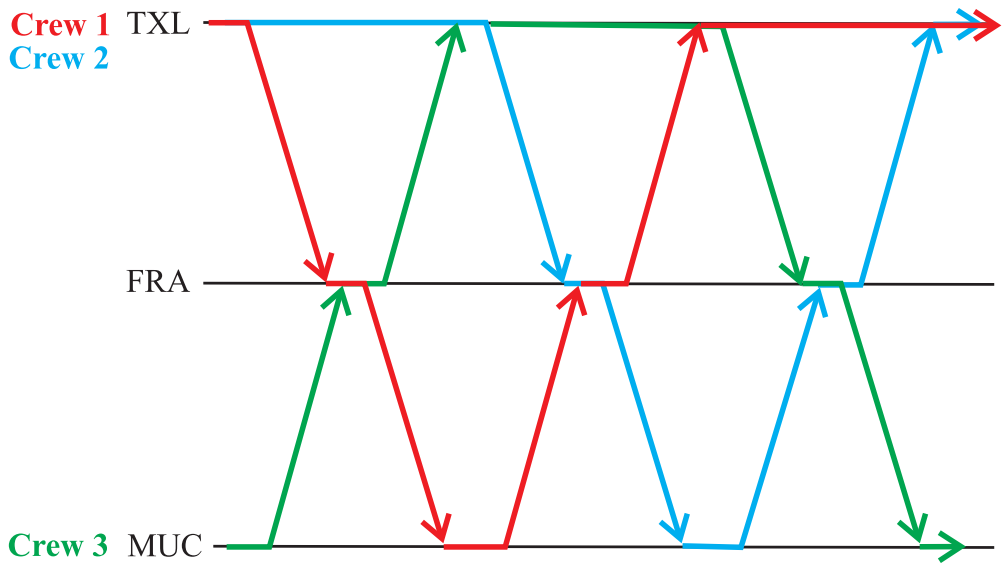
Constraints:

- qualification of employees
- labor code restrictions (e.g. at least 12 hours of rest during 24 hours)
- collective agreement restrictions (e.g. maximum number of night shifts in a block)
- employees demands (e.g. required day-off)
- fair assignment of shifts (same number of weekend shifts)

Assignment of Shifts to Employees

Assignment of human resources is often formalized as a **matching in a bipartite graph**

The problem becomes harder when we consider the geographic position of the employees (e.g. stewards and pilots in an airline company).



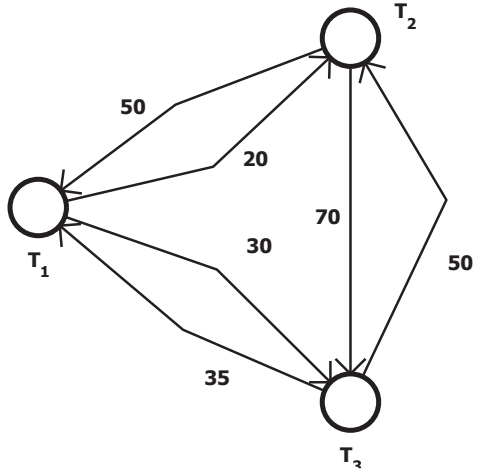
Storage System in Automated Warehouse

Goal:

- reduce time of the vehicle trips

The problem is represented by the digraph.
The nodes of the graph represent the picking tasks.
Edge (i, j) means the possibility to perform task j right after i. The edge cost represents the duration of the trip from i to j.

Can be formulated as
Asymmetric Traveling Salesman Problem.



Vehicle Routing

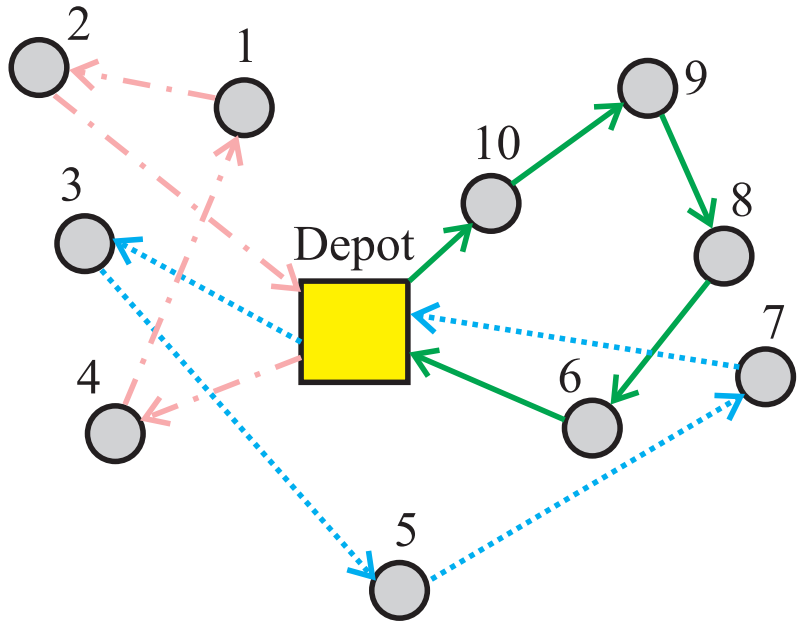
Goods, customers and fleet of cars.

Goal:

- fulfill demands of customers
- minimize transportation cost

Constraints:

- car capacity
- time windows
- traffic jams
- shifts, breaks



Surface Mount Technology

The placement machines are scarce resource of the Printed Circuit Board (PCB) production, due to their high cost.

Goal:

- Maximize production speed

Constraints:

- Assembly line configuration
- Description of produced PCB

Problem can be divided into two subproblems.

A) Allocation of the components to the placement heads

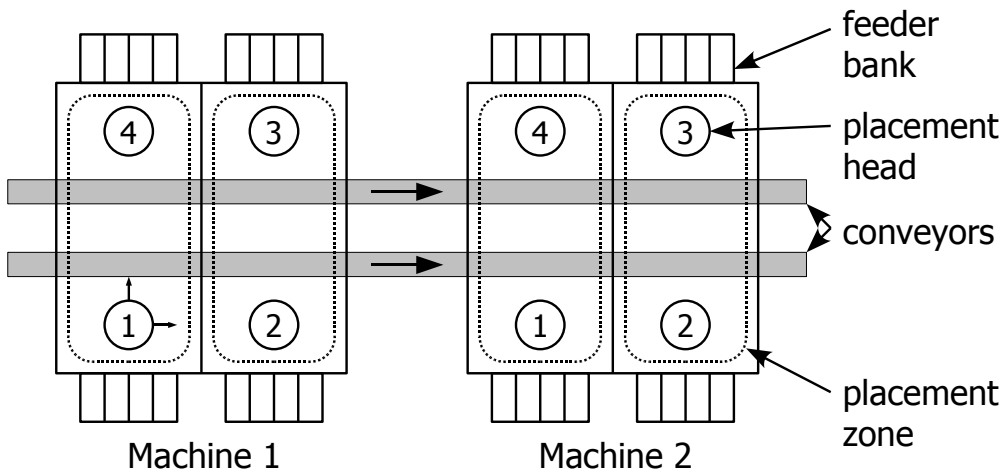
Can be formulated as an **Assignment Problem**

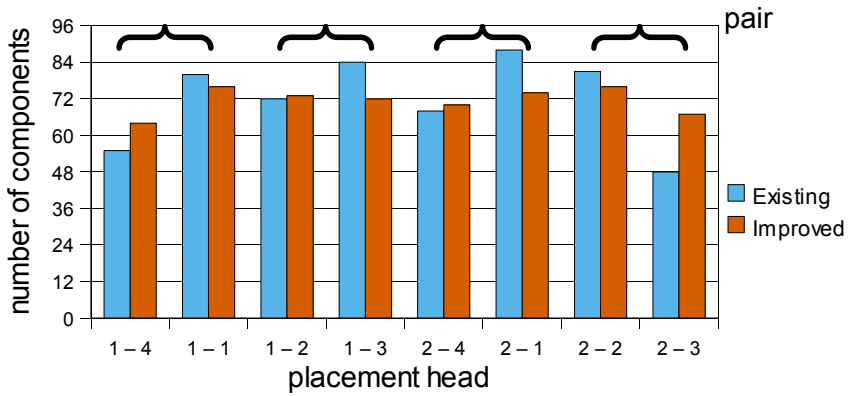
Input:

- types of SMT components
- number of components of a given type
- precedence relations among some components
- machine parameters

Output:

- allocation of components to the placement heads





Surface Mount Technology

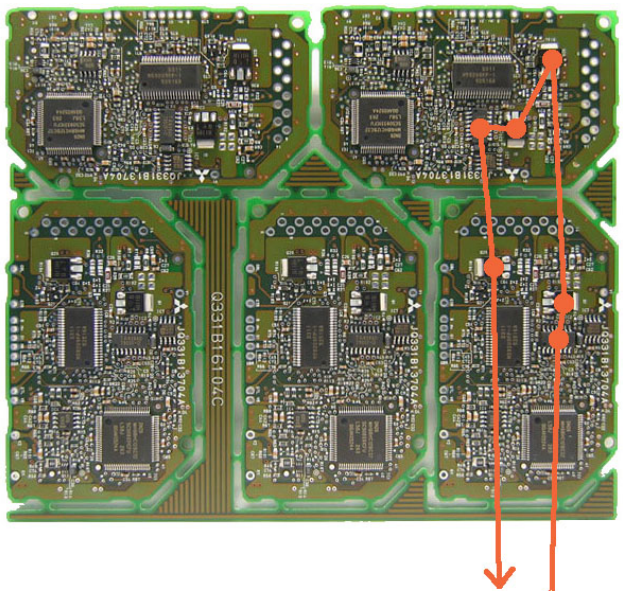
B) Sequencing for a given head can be formulated as a (capacitated multi-trip) **Traveling Salesman Problem**

Input:

- allocation of components to the assembly head
- position of components on the PCB
- capacity of the head

Output:

- assembly sequence
- operation time



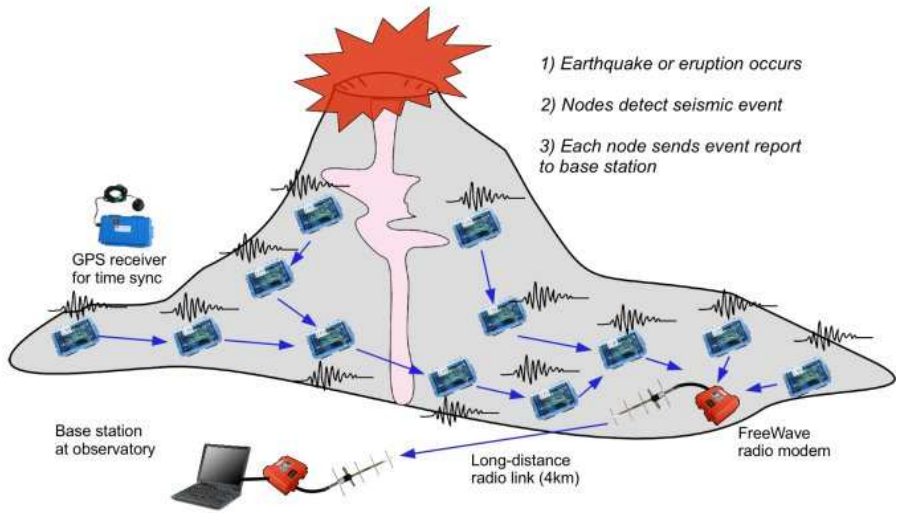
Routing in Wireless Sensor Network - Specification

A volcano monitoring using autonomous devices equipped with:

- own power supply
- wireless short range communication
- temperature sensors

Goal:

- create routing tables
- minimize energy consumption



Constraints:

- capacity of each communication link
- limited transmitter performance
- maximal allowed end-to-end delay of communication
- memory capacity of devices

Routing in Wireless Sensor Network - Formalization

Can be formalized as a **Multi-Commodity Network Flows problem**.
WSN represented by a graph (devices = vertices, links = edges)
Communication demand = commodity flow:

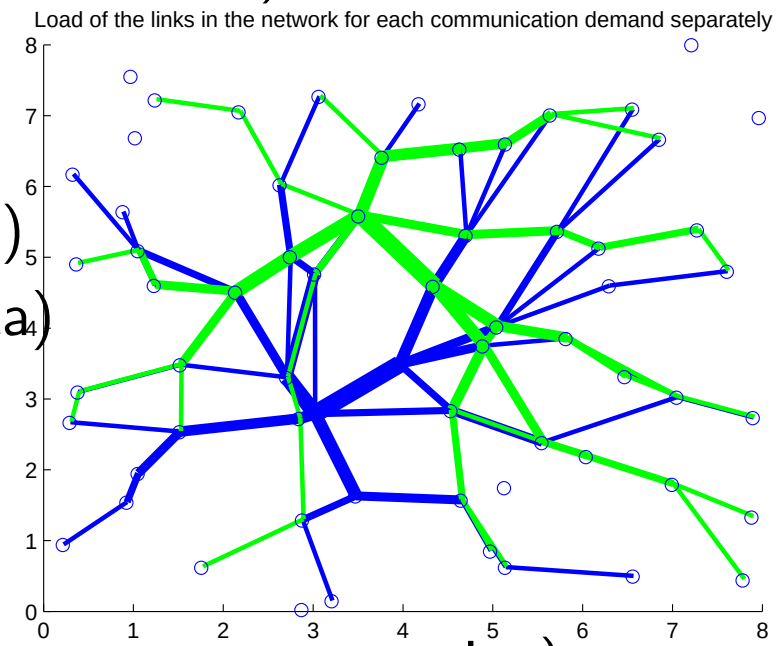
- source devices and destination devices
- volume of demand (quantity of data per time unit)
- deadline (maximum number of hops)

Communication link:

- capacity (quantity of data per time unit)
- price (energy to transfer one unit of data)

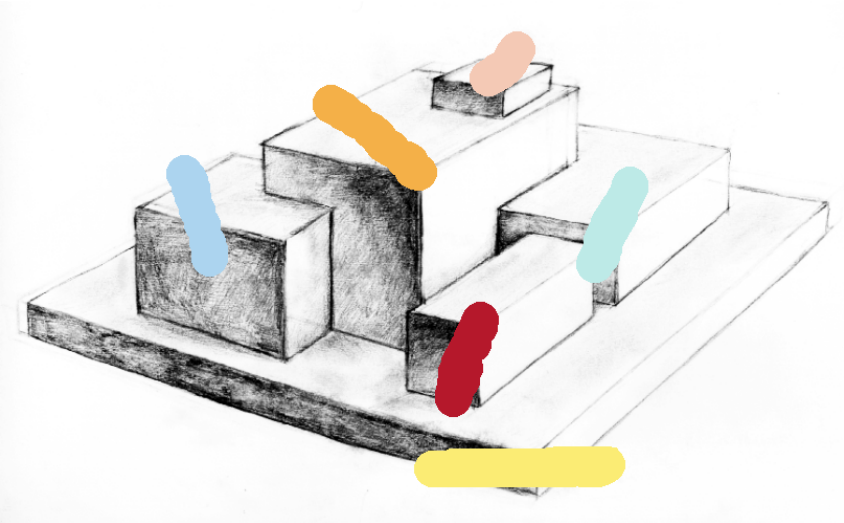
Other variants:

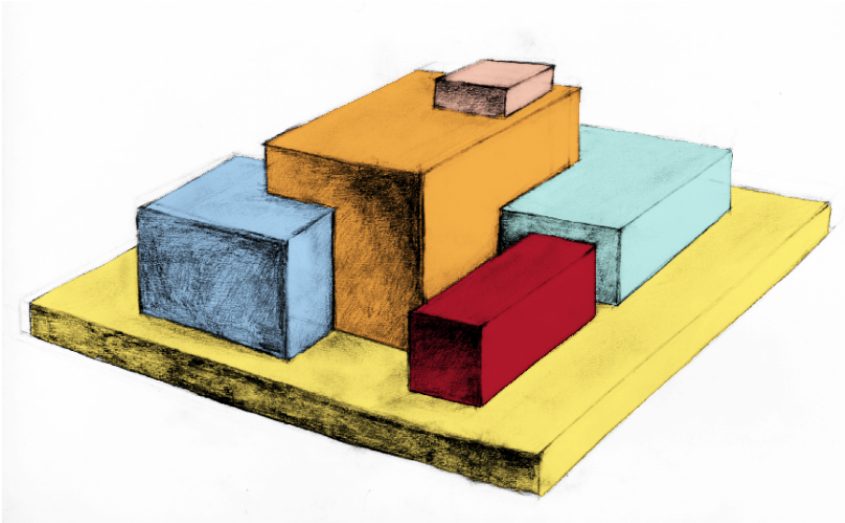
- various delays on links
- indivisible flows
- maximize the network lifetime (minimize energy consumption)
- distributed version



Coloring Book - Specification

The aim is to color hand-drawn black-and-white image with minimal effort and still achieve accuracy comparable to precise manual pixel-wise coloring.



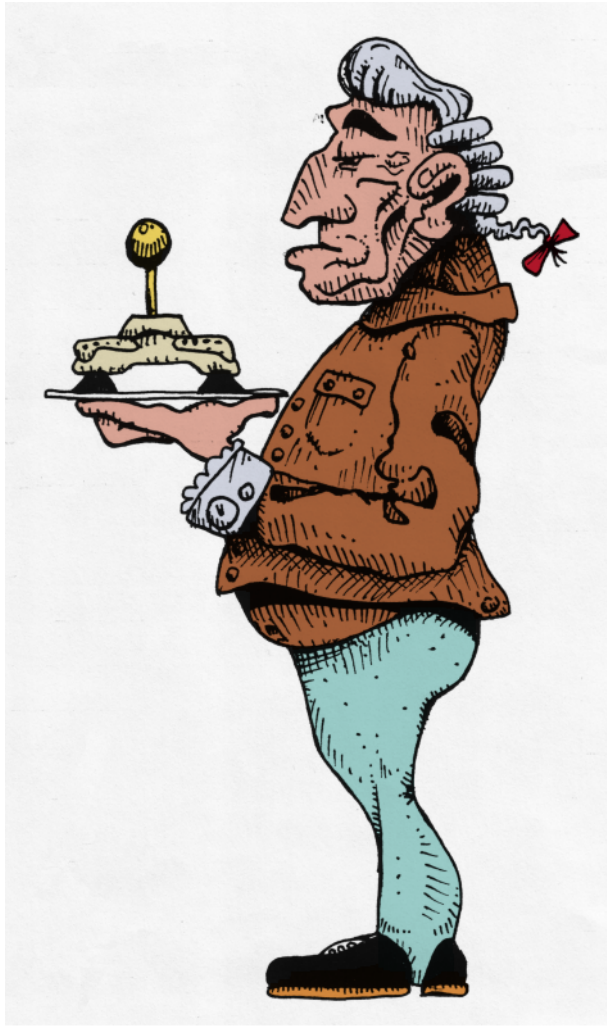


Problems:

- color leakage through small gaps in outlines
- too many small regions
- color flooding into outlines (zapít barvu do vnitřku konturky)
- robustness to inaccuracy (přetahy)

Coloring Book - Application

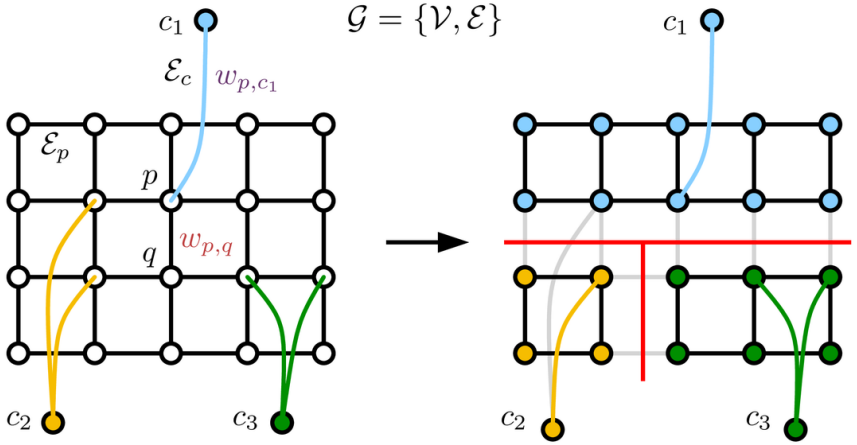
Used by illustrators and for production of cartoon animations like Rumcajs, Dr. Animo,and Husiti.



Coloring Book Formulated as Multiway Cut Problem

Multiway cut (given a graph with a set of terminals, find the minimum-weight set of edges whose removal separates each pair of terminals) is NP-hard problem.

- terminal nodes with different colors are specified by the user
- edge capacity is high (two neighbor pixels are white) or low (two neighbor pixels are black)
- a cut is the boundary between two colors



It can be approximated by multiple binary minimum cut problems which are polynomial. GridCut U.S. patent by Jamriška and Sýkora, DCGI.

Region Covering with Satellite Images

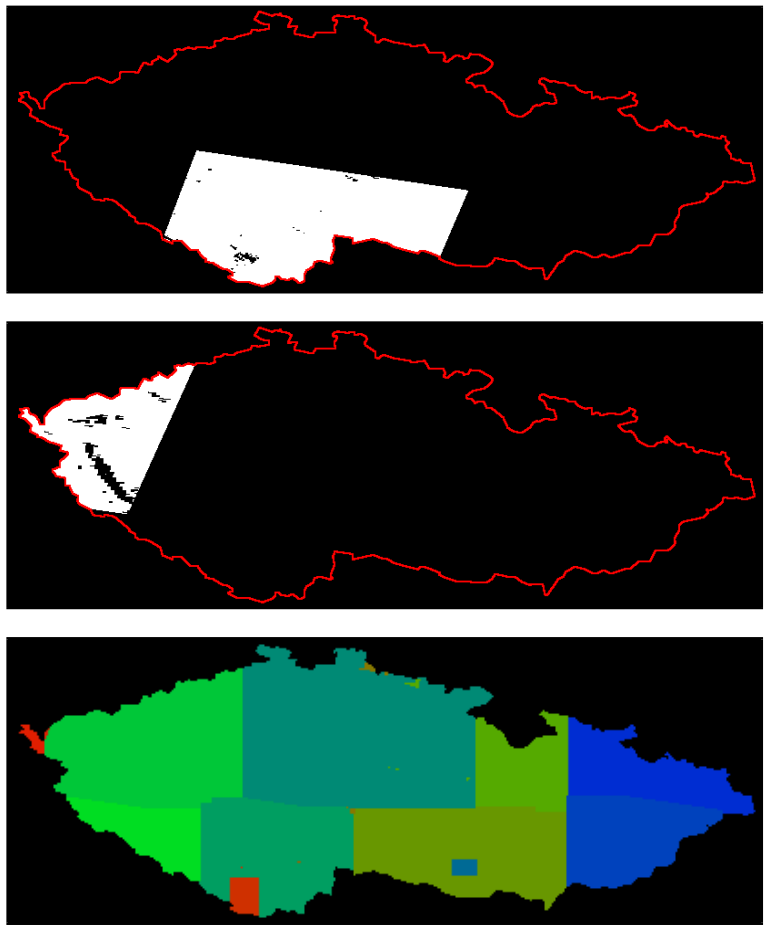
Input: Large database with geo-referenced satellite images and visibility masks (clouds!) including time-stamps. Customer defined spatial region (ROI) & time interval.

Output: set of images s.t. each pixel of the ROI is visible in one image.

Objective function: weighted combination of the two criteria:

a) minimize number of used images (i.e., set-cover)

b) penalize time difference across image borders

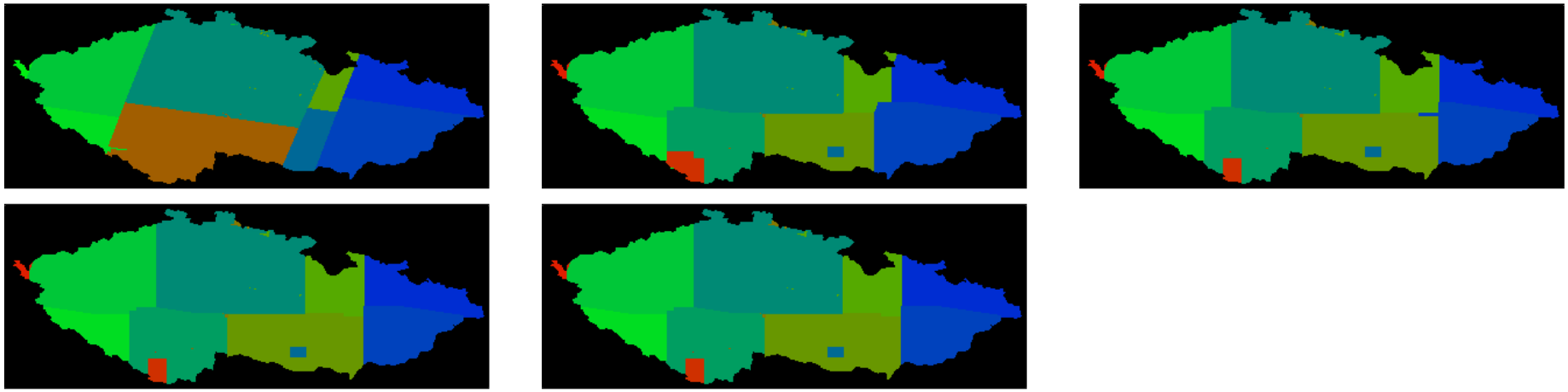


Two visibility masks and final cover (similar colours encode similar dates)

Region Covering with Satellite Images

Generic ILP solvers are too slow. The problem was solved by an approximation algorithm, which

- starts from solution obtained by greedy set-cover algorithm (neglecting criterion b)
- iteratively improves the solution by solving minimum cut problems in each iteration as shown below.

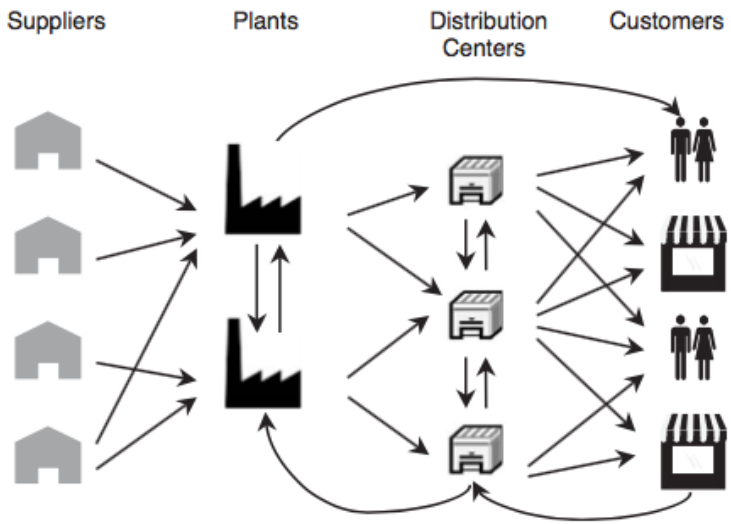


The problem was solved as an industrial project for the geo-information company GISAT s.r.o. by Boris Flach, CMP.

Supply Network Management

Given a prediction of customers demands, the aim is to plan **production and transport** of products in order to maximize the profit.

- Each product in each layer of the network has different production rate.
- Each order has different delivery date.

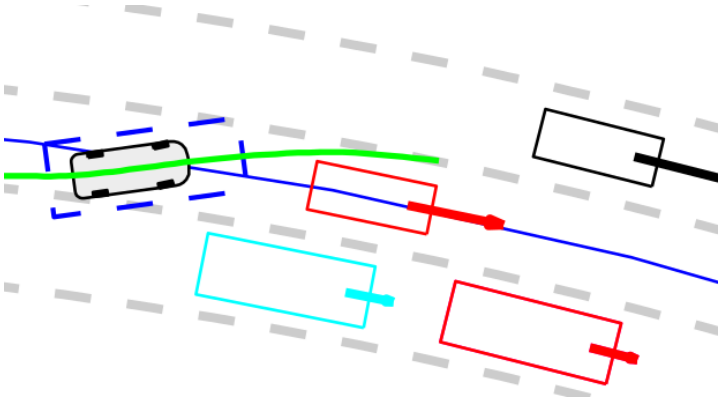


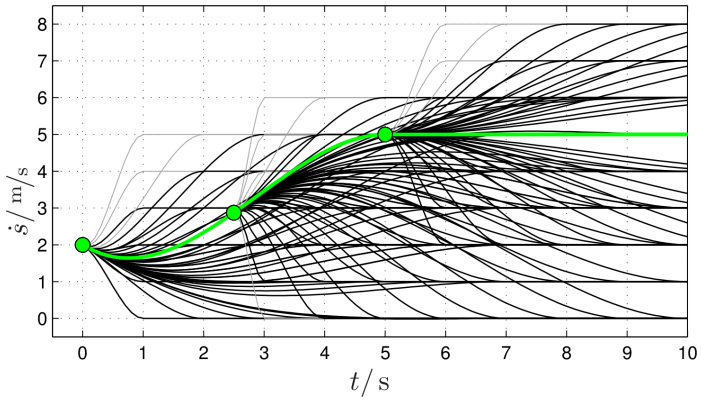
A solution is the amount of products in time, such that the profit is maximized and the risk of looses (unsold products) is mitigated.

Current project funded by an IT company: solve a supply network management problem given **historical data** of a customer.

Trajectory Panning for Autonomous Vehicle

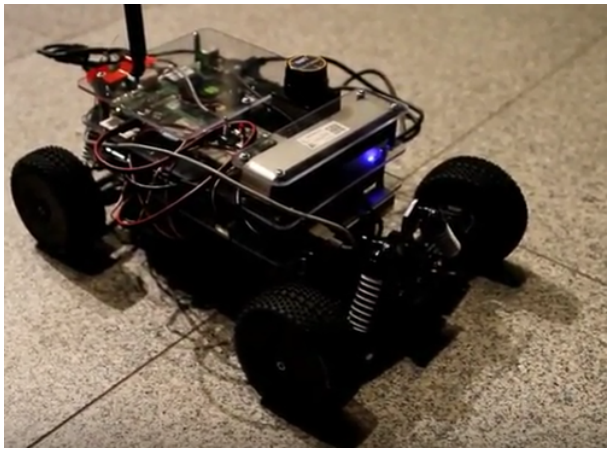
Given a manoevre to be performed the aim is to find an optimal trajectory such that the constraints given by vehicle dynamics are respected.





Project work:

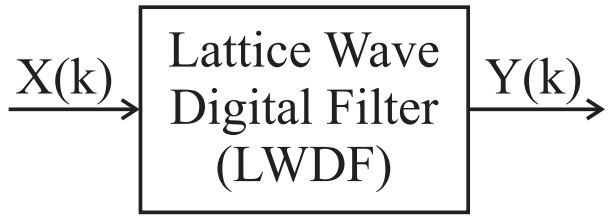
- reaction to dynamic changes
- reliability of perception
- data fusion
- computing power vs reliability



People have driving schools, robots have Combinatorial Optimization.

Configurable HW for Specific DSP Application Design

An example application is a simple digital filter LWDF. $X(k)$ are samples of input signal, $Y(k)$ are samples of output signal.



for k=1 **to** N **do**

T_1: $a(k) = X(k) - c(k-2)$

T_2: $b(k) = a(k) * \alpha$

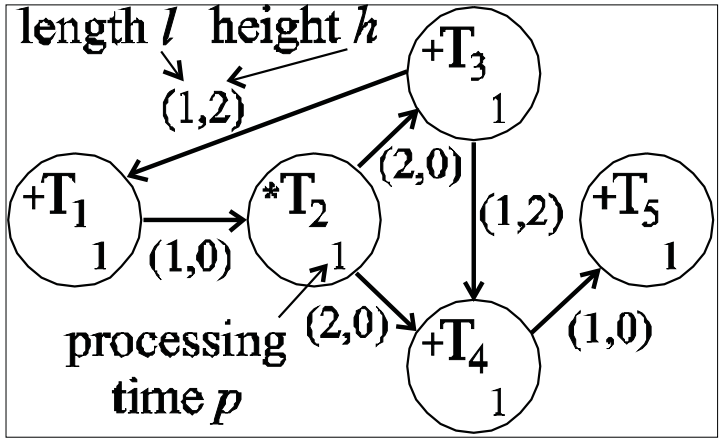
T_3: $c(k) = b(k) + X(k)$

T_4: $d(k) = b(k) + c(k-2)$

T_5: $Y(k) = X(k-1) + d(k)$

end

LWDF filter algorithm



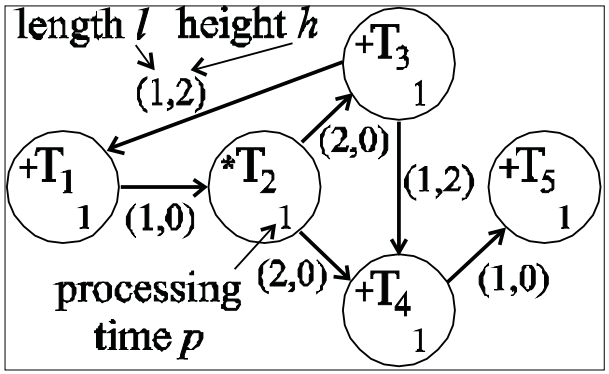
Atomic operation dependencies

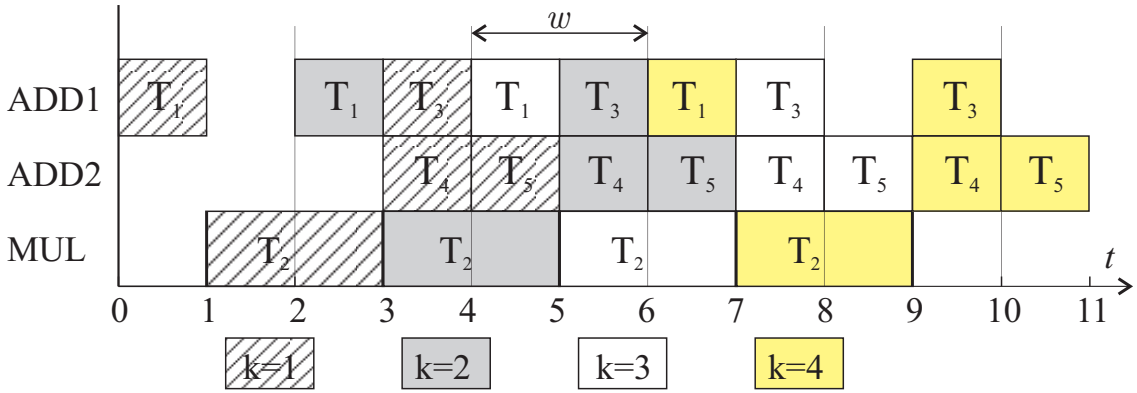
Configurable HW for Specific DSP Application Design

Can be formalized as a cyclic extension of $PS\,|temp|\,C_{max}$

Used hardware features

unit	count [−]	computation time [*clk*]
ADD	2	1
MUL	1	2





Convincing Arguments

Explain the typical goals of optimization:

- automation of the design/decission process
- increase the volume of the production (shorter production-line cycle)
- cost reduction (fuel saving, less machines)
- risk reduction (error elimination due to automated creation of production schedule)
- lean manufacturing (supply and stores reduction, outgrowths reduction when delay in supply)
- increase of the flexibility (faster reaction to structure or constraint change)
- user-friendly solutions (balanced schedule for all employees)

How does this course cover the important skills?

An engineer is usually hired to systematically solve the problem.

Skill	Example
Business case	Attract a customer
Specification	Production scheduling
Formalization	Flow-shop
Algorithms	Johnson's algorithm
Prototype solution	Matlab OPL, ...
Implementation	C#, dB, ...

How does this course cover the important skills?

An engineer is usually hired to systematically solve the problem.

Skill	Example	Lectures
Business case	Attract a customer	-
Specification	Production scheduling	Application examples
Formalization	Flow-shop	Formulation of the opt. problem
Algorithms	Johnson's algorithm	Pseudocode iteration with data
Prototype solution	Matlab OPL, ...	-
Implementation	C#, dB, ...	-

How does this course cover the important skills?

An engineer is usually hired to systematically solve the problem.

Skill	Example	Lectures	Seminars
Business case	Attract a customer	-	Project?
Specification	Production scheduling	Application examples	Project Exercises
Formalization	Flow-shop	Formulation of the opt. problem	Project Exercises
Algorithms	Johnson's algorithm	Pseudocode iteration with data	Project
Prototype solution	Matlab OPL, ...	-	Project Exercises
Implementation	C#, dB, ...	-	Project?

How does this course cover the important skills?

An engineer is usually hired to systematically solve the problem.

Skill	Example	Lectures	Seminars	Exam
Business case	Attract a customer	-	Project?	-
Specification	Production scheduling	Application examples	Project Exercises	Project
Formalization	Flow-shop	Formulation of the opt. problem	Project Exercises	Project Exam
Algorithms	Johnson's algorithm	Pseudocode iteration with data	Project	TestI,II Exam
Prototype solution	Matlab OPL, ...	-	Project Exercises	Project Prac.Test
Implementation	C#, dB, ...	-	Project?	-

Algorithms are used to solve the problems

Combinatorial optimization uses combinatorial algorithms

Many problems can be formalized by:

- constraints
- optimization criterion

It is not always easy to find the optimal solution efficiently.
In the case of exhaustive search while enumerating all solutions, the computation time for bigger instances can be enormous.
For example, the permutation Flow-shop problem has complexity of $n!$, where n is the number of jobs.

Time Complexity of Algorithms

n	$100n \log n$	$10n^2$	$n^{3.5}$	$n^{\log n}$	2^n	$n!$
10	3 μs	1 μs	3 μs	2 μs	1 μs	4 ms
20	9 μs	4 μs	36 μs	420 μs	1 ms	76 years
30	15 μs	9 μs	148 μs	20 ms	1 s	$8 \cdot 10^{15}$ y.
40	21 μs	16 μs	404 μs	340 ms	1100 s	
50	28 μs	25 μs	884 μs	4 s	13 days	
60	35 μs	36 μs	2 ms	32 s	37 years	
80	50 μs	64 μs	5 ms	1075 s	$4 \cdot 10^7$ y.	
100	66 μs	100 μs	10 ms	5 hours	$4 \cdot 10^{13}$ y.	
200	153 μs	400 μs	113 ms	12 years		
500	448 μs	2.5 ms	3 s	$5 \cdot 10^5$ y.		
1000	1 ms	10 ms	32 s	$3 \cdot 10^{13}$ y.		
10^4	13 ms	1 s	28 hours			
10^5	166 ms	100 s	10 years			
10^6	2 s	3 hours	3169 y.			
10^7	23 s	12 days	10^7 y.			
10^8	266 s	3 years	$3 \cdot 10^{10}$ y.			
10^{10}	9 hours	$3 \cdot 10^4$ y.				
10^{12}	46 days	$3 \cdot 10^8$ y.				

Graph Theory Overview

Graphs informally:

- A graph consists of nodes and edges.
- Each edge joins two nodes, it is directed or undirected.
- In a directed graph, the edge leaves one node and enters another one.
- In an undirected graph, an edge is a symmetric join of two nodes.

Directed Graph

Directed graph (digraph) is a triplet (V, E, Ψ):

- V is a finite set of **nodes**
- E is a finite set of **directed edges**
- Ψ is a mapping from the set of edges to the ordered pair of nodes, i.e. $\Psi : E \to \{(v, w) \in V \times V : v \neq w\}$

Undirected Graph

Undirected graph is a triplet (V, E, Ψ):

- V is a finite set of nodes.
- E is a finite set of **undirected edges**
- Ψ is a projection from the set of edges to the 2-element subset of V, i.e. $\Psi : E \to \{X \subseteq V : |X| = 2\}$

Edges

- Two edges e, e' are called **parallel edges** if $\Psi(e) = \Psi(e')$.
- A graph containing parallel edges is called a **multigraph** - we usually do not work with a multigraph.
- A graph that does not contain parallel edges is called **simple graph**.
- We usually denote simple graphs by pair $G = (V(G), E(G))$, where $V(G)$ is a set of nodes and $E(G)$ is set of (unordered or ordered) pairs of nodes that describes the edges (i.e. we identify an edge with its image $\Psi(e)$).
- Undirected edge $e = \{v, w\}$ or directed edge $e = (v, w)$ connects v and w. Nodes v and w are the endpoints of e or we can say they are incident with e. In case of a directed graph we say that e leaves v and enters w.

Comparison of Graphs

For directed graph G we can have an **underlaying undirected graph** G', with the same set of vertices and undirected edge $\{v, w\}$ for every directed edge (v, w) from G.
We call graph H a **subgraph** (podgraf) of graph G, if we can create it by omitting the nodes (zero or more of them) or edges (when an edge is in the subgraph, it's endpoints must be there as well). Special cases of subgraphs:

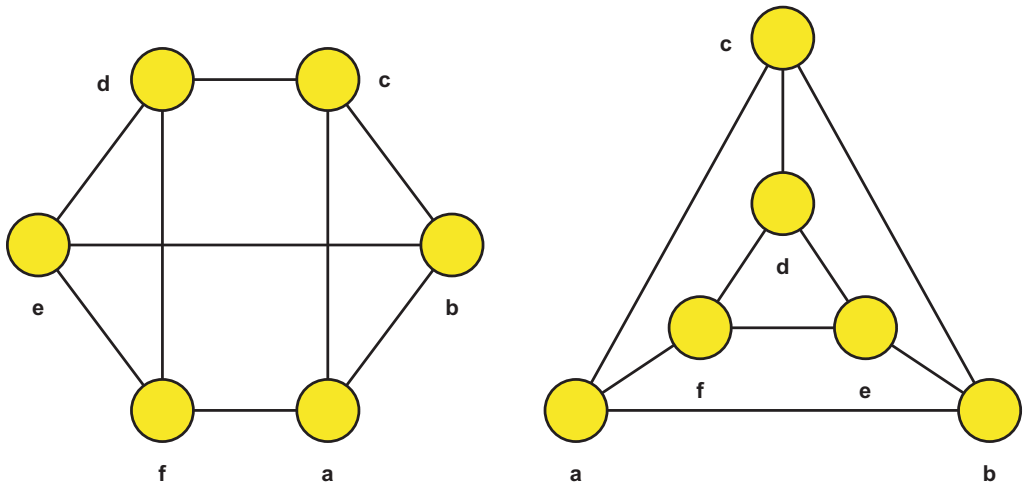
- H of G is called **spanning** (faktor) if $V(G) = V(H)$ and we omit some or zero edges.
- H is called **subgraph induced by set of vertices** $V(H) \subseteq V(G)$ if we omit some (or zero) vertices and incident edges,
  i.e. H contains all edges from G whose both endpoints are in $V(H)$.

Comparison of Graphs – Isomorphic Graphs

Two graphs G and H are called **isomorphic** if there are bijections: $\Phi_V : V(G) \to V(H)$ and $\Phi_E : E(G) \to E(H)$ such that:

- for directed graphs: $\Phi_E((v, w)) = (\Phi_V(v), \Phi_V(w))$ for all $(v, w) \in E(G)$
- for undirected graphs: $\Phi_E(\{v, w\}) = \{\Phi_V(v), \Phi_V(w)\}$ for all $\{v, w\} \in E(G)$

We usually don't deal with isomorphic graphs when talking about algorihms, but it is good to be aware of them during modelling.



Other Terms for Digraphs

For node v of digraph G we define:

- a set of **successors** of v is a set of nodes such that there is an edge from v to each of these nodes, i.e. $\{w \in V : (v, w) \in \Psi(E)\}$
- a set of **predecessors** v is a set of nodes such that there is an edge from each of these nodes to v, i.e. $\{w \in V : (w, v) \in \Psi(E)\}$
- $\Gamma(v)$, a set of **neighbors** of v, is set of nodes connected by an edge with v, i.e. a union of successors and predecessors
- $\delta^+(v)$, a set of edges leaving v
- $\delta^-(v)$, a set of edges entering v
- $\delta(v)$, a set of edges incident with v
- $|\delta^+(v)|$, **out-degree**
- $|\delta^-(v)|$, **in-degree**
- $|\delta(v)|$, **degree of node**

Notice: $\sum_{v \in V(G)} |\delta(v)| = 2\,|E(G)|$.
the number of nodes with an odd degree is even

Other Terms for Digraphs

For sets $X, Y \subseteq V(G)$ of directed graph G we define:

- $E^+(X, Y)$ a set of edges from X to Y, i.e. $E^+(X, Y) = \{(x, y) \in E(G) : x \in X \setminus Y, y \in Y \setminus X\}$
- $\delta^+(X)$, a set of edges leaving set X, i.e. $\delta^+(X) := E^+(X, V(G) \setminus X)$
- $\delta^-(X)$, a set of edges entering set X, i.e. $\delta^-(X) := \delta^+(V(G) \setminus X)$
- $\delta(X)$, a set of "border" edges of set X, i.e. $\delta(X) := \delta^+(X) \cup \delta^-(X)$
- $\Gamma(X)$, a set of neighbor nodes of set X, i.e. $\Gamma(X) := \{v \in V(G) \setminus X :$ "border" edge $e \in \delta(X)$ exists and it is incident with node $v\}$

Similar Terms for an Undirected Graph

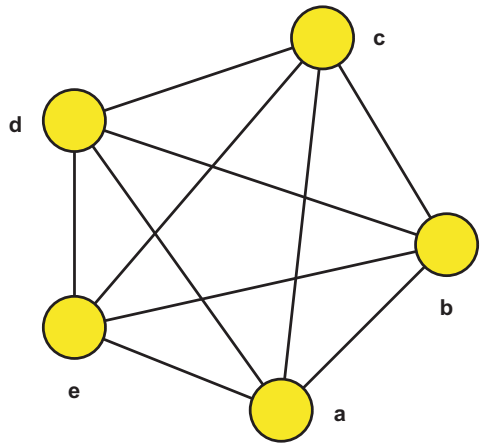
These terms are used in undirected graphs:

- $\Gamma(v)$, a set of neighbors of node v
- $\delta(v)$, a set of edges incident with v
- $|\delta(v)|$, a degree of v
- $E(X, Y)$, a set of edges between X and Y, i.e. $E(X, Y) = \{\{x, y\} \in E(G) : x \in X \setminus Y, y \in Y \setminus X\}$
- $\delta(X)$, a set of "border" edges of set X, i.e. $\delta(X) := E(X, V(G) \setminus X)$
- $\Gamma(X)$, set of neighbours of set X, i.e. $\Gamma(X) := \{v \in V(G) \setminus X : E(X, v) \neq \emptyset\}$

Mostly, we will use digraphs (from an undirected graph we can make a digraph by replacing every undirected edge by a pair of inverse directed edges).

Special Graphs

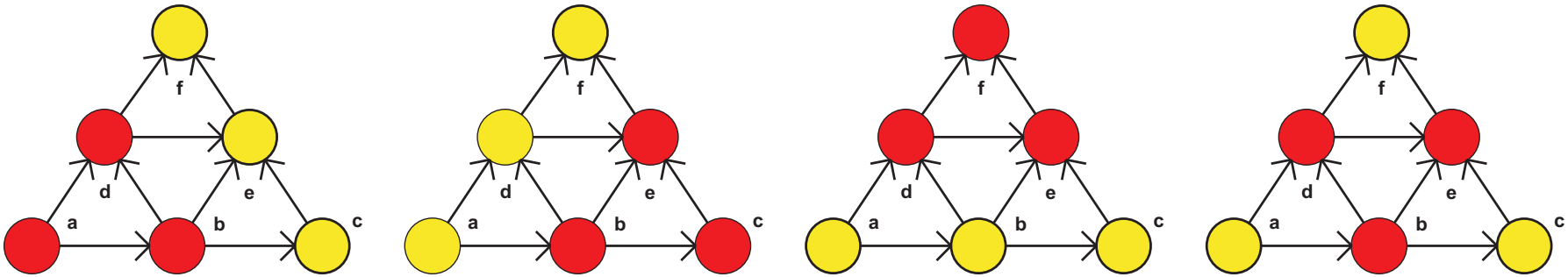
A complete digraph (úplný graf) is a simple graph $G = (V, E)$, where E is a set of all possible pairs of different nodes of V.
A complete undirected graph is a simple graph, in which every pair of vertices is connected by a unique edge. We denote it K_n, where n is the number of nodes.
A graph is called **regular** (regulární) if all its nodes have the same degree. If the degree of all nodes is k, the graph is called **k-regular**.



Special Graphs

A complement of simple graph G is simple graph H such that $G + H$ is the complete graph. A pair of nodes in the complement is connected if it is not connected in G.

A clique is a subgraph that is complete. The number of nodes in the maximum (biggest) clique is called **the clique number**



Edge Progression, Walk and Path

- **Edge progression (sled)** is a sequence $v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1}$ such that $e_i = (v_i, v_{i+1}) \in E(G)$ or $e_i = \{v_i, v_{i+1}\} \in E(G)$ for all i.
- Edge progression is called **closed** if $v_1 = v_{k+1}$.
- Directed (undirected) **walk (tah)** is directed (undirected) edge progression, where no edge appears more than once, i.e. $e_i \neq e_j$ for all $1 \leq i < j \leq k$.
- Directed (undirected) **path (cesta)** is directed (undirected) walk, where no node appears more than once, i.e. $v_i \neq v_j$ for all $1 \leq i < j \leq k + 1$.
- **The circuit (also called cycle)** is a subgraph $(v_1, \cdots, v_k, e_1, \cdots, e_k)$ such that the sequence $v_1, e_1, v_2, \cdots, v_k, e_k, v_1$ is a closed undirected walk (tah) and $v_i \neq v_j$ for $1 \leq i < j \leq k$.
- **The cycle (also called circuit)** is a subgraph $(v_1, \cdots, v_k, e_1, \cdots, e_k)$ such that the sequence $v_1, e_1, v_2, \cdots, v_k, e_k, v_1$ is a closed directed walk (tah) and $v_i \neq v_j$ for $1 \leq i < j \leq k$.

Undirected Graph - Tree, ...

- An undirected graph is called **connected**, if every pair of nodes is connected by an undirected path.
- The maximal connected subgraphs of G are its **connected components**.
- Every node of the graph is included in exactly one connected component.
- The connected component containing node x can be found as a complete subgraph induced by the set of all nodes which can be reached from x via the undirected path.
- **A forest** is an undirected graph G without a circuit.
- **A tree** is an undirected graph G without a circuit that is connected.
- For every connected graph there exists a spanning that is the tree and it is called the **spanning tree**.

Undirected Graph - Tree

Let G be an undirected graph with n nodes, then the following statements are equivalent:

- (a) G is a tree.
- (b) G has $n-1$ edges and no circuit.
- (c) G has $n-1$ edges and is connected.
- (d) G is connected and while removing any edge it will not be connected anymore.
- (e) G is a minimal graph which has $\delta(X) \neq \oslash$ for all $\oslash \neq X \subset V(G)$
- (f) G is circuit-free and the addition of any edge creates a circuit.
- (g) G contains a unique path between any pair of vertices.

Undirected Graph - Tree

Proof:
(a)⇒(g): follows from the fact that the union of two distinct paths with the same endpoints contains a circuit.
(f)⇒(b)⇒(c): follows from the fact that for a forest with n nodes, m edges and p connected components $n = m + p$. (The proof is a trivial induction on m).
(g)⇒(e)⇒(d): see [1] page 17 Proposition 2.3.
(d)⇒(f): trivial.
(c)⇒(a): G is connected with $n - 1$. As long as there are any circuits in G, we destroy them by deleting any edge of the circuit. Suppose we have deleted k circuits, the resulting graph G' is a tree (contains no circuit and is connected) and has $m = n - 1 - k$ edges. So
$n = m + p = n - 1 - k + 1$, implying $k = 0$.

Connectivity and Trees in Digraphs

- Digraph is connected if the underlying undirected graph is connected.
- Digraph G is **strongly connected** if there is a path from x to y and from y to x for all x, y in G.
- **The strongly connected component** of G is every maximal strongly connected subgraph H of G.
- Node $x \in V(G)$ is a **root** of graph G, if there is a directed path from x to every node of G.
- **An out-tree** (called also arborescence or branching) is digraph G that contains a root. No edge enters the root, but exactly one edge enters every other node.
- Properties of trees in a directed graph - see [1] page 18
- **A binary tree** is tree in which each node has at most two child nodes.
- **A topological ordering** of a digraph is a linear ordering of its vertices such that for every directed edge from vertex u to vertex v, u comes before v in the ordering.
  Any **directed acyclic graph** has at least one topological ordering.

Summary

A graph is:

- often used to formalize optimization problems
- very general
- easy to represent

Literature

B. H. Korte and Jens Vygen.
Combinatorial Optimization: Theory and Algorithms.
Springer, third edition, 2006.