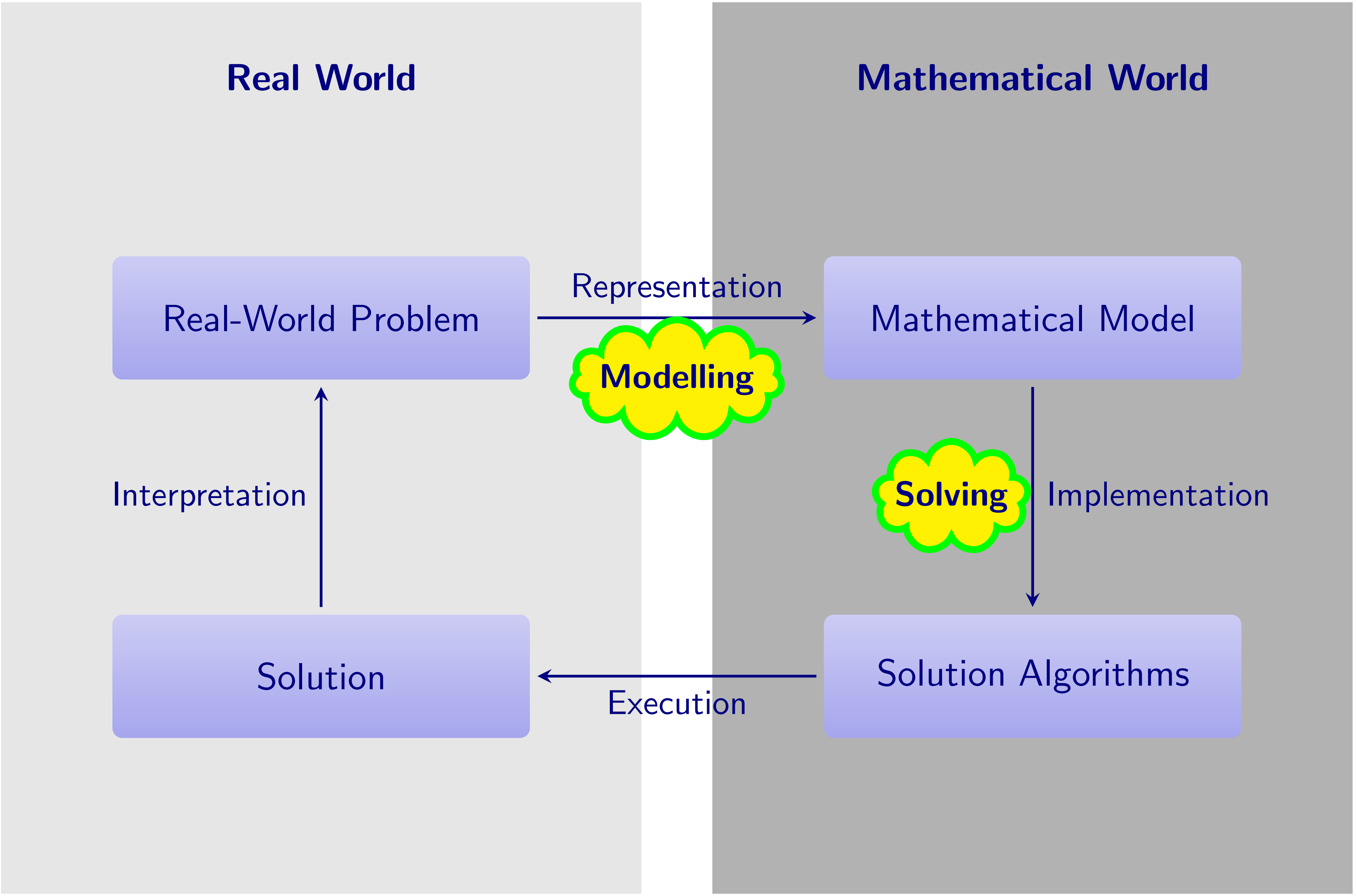
Mathematical Tools for Engineering and Management

Revision

08 Feb 2012

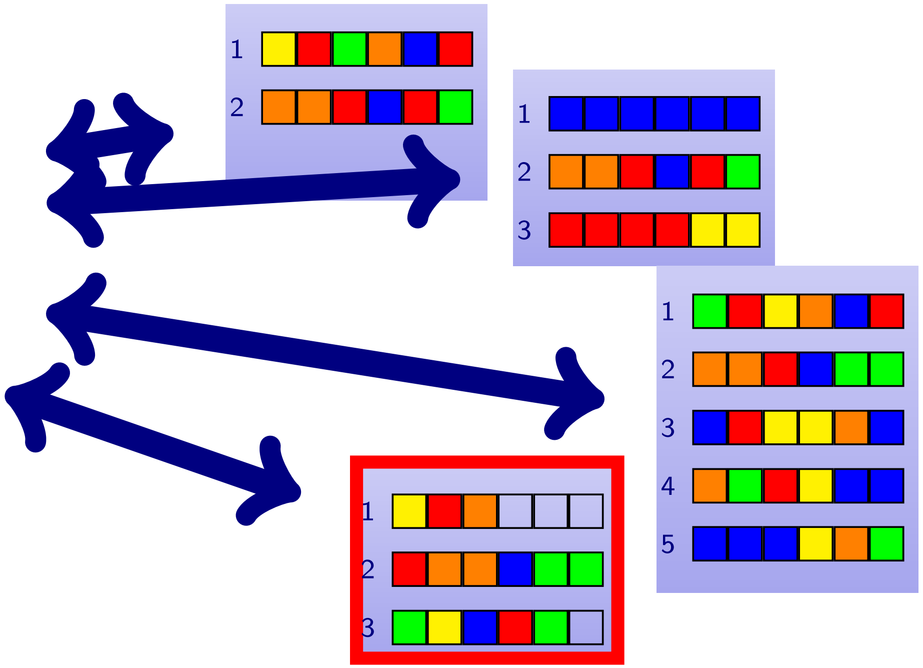
- Models, Data and Instances
- Linear Optimization
  - Modelling as a linear program
  - Solving a linear program (graphically, and in princple by the simplex algorithm)
  - Sensitivity analysis
- (Mixed) Integer Programming
  - Modelling as a (mixed) integer program
  - How to solve a (mixed) integer program (in principle)
- Combinatorial Optimization
  - Exemplary problems, algorithms, and runtimes
- Nonlinear Optimization
  - Local and global optima, convex optimization
- Scheduling
- Lot Sizing
- Multicriteria Optimization


Real World
Mathematical World
Real-World Problem
Representation
Mathematical Model
Modelling
Interpretation
Solving
Implementation
Solution
Execution
Solution Algorithms

Given n sequences of colours $c_{i,j}$ $(1 \leq i \leq n,\ j \in \mathbb{N})$ find a mapping $f : \mathbb{N} \to \{0, \ldots, n\}$ such that $\#\{k \mid c_{f(k),j(k)} \neq c_{f(k+1),j(k+1)}\}$ is minimized, where $j(k) :=$

$$\begin{cases} 1 & \text{if } k{=}1 \text{ or } f(k'){\neq}f(k)\ \forall k'{<}k \\ j(k'){+}1 & \text{otherwise, where } k' \text{ is maximal with } k'{<}k,\ f(k'){=}f(k) \end{cases}$$

Model



Data Sets

- ▷ A model is a mathematical formulation of the problem, independent of any concrete data (as possible input)
- ▷ An instance is a mathematical model, together with one associated data set

Mathematical optimization

objective to maximize/minimize — constraints to respect — solution: variable assignment

Mixed integer programming

linear objective

linear constraints

both continuous and integer variables

Linear programming

only continuous variables

Integer programming

only integer variables

Nonlinear optimization

non-linear objective allowed

non-linear constraints allowed

continuous and/or integer variables

S

Sets of relevant elements
(for example: products, cities, machines, types of raw material, ...)

S Sets of relevant elements
(for example: products, cities, machines, types of raw material, ...)

P Parameters: Values specified for (combinations of) elements of the sets
(for example: profits for products, demand for products, distances between cities, capacity of machines, prices of one unit of raw material, ...)

S Sets of relevant elements
(for example: products, cities, machines, types of raw material, ...)

P Parameters: Values specified for (combinations of) elements of the sets
(for example: profits for products, demand for products, distances between cities, capacity of machines, prices of one unit of raw material, ...)

V Variables: Unknowns to be determined
(for example: number of items to produce, number of shops to open in a certain city, decision to buy a certain machine or not, amount of raw material to use, ...)

S Sets of relevant elements
(for example: products, cities, machines, types of raw material, ...)

P Parameters: Values specified for (combinations of) elements of the sets
(for example: profits for products, demand for products, distances between cities, capacity of machines, prices of one unit of raw material, ...)

V Variables: Unknowns to be determined
(for example: number of items to produce, number of shops to open in a certain city, decision to buy a certain machine or not, amount of raw material to use, ...)

C Constraints: Relationships that have to hold between variables and parameters
(for example: maximal number of items that can be produced by a machine, minimal number of shops to open, budget for buying raw material, ...)

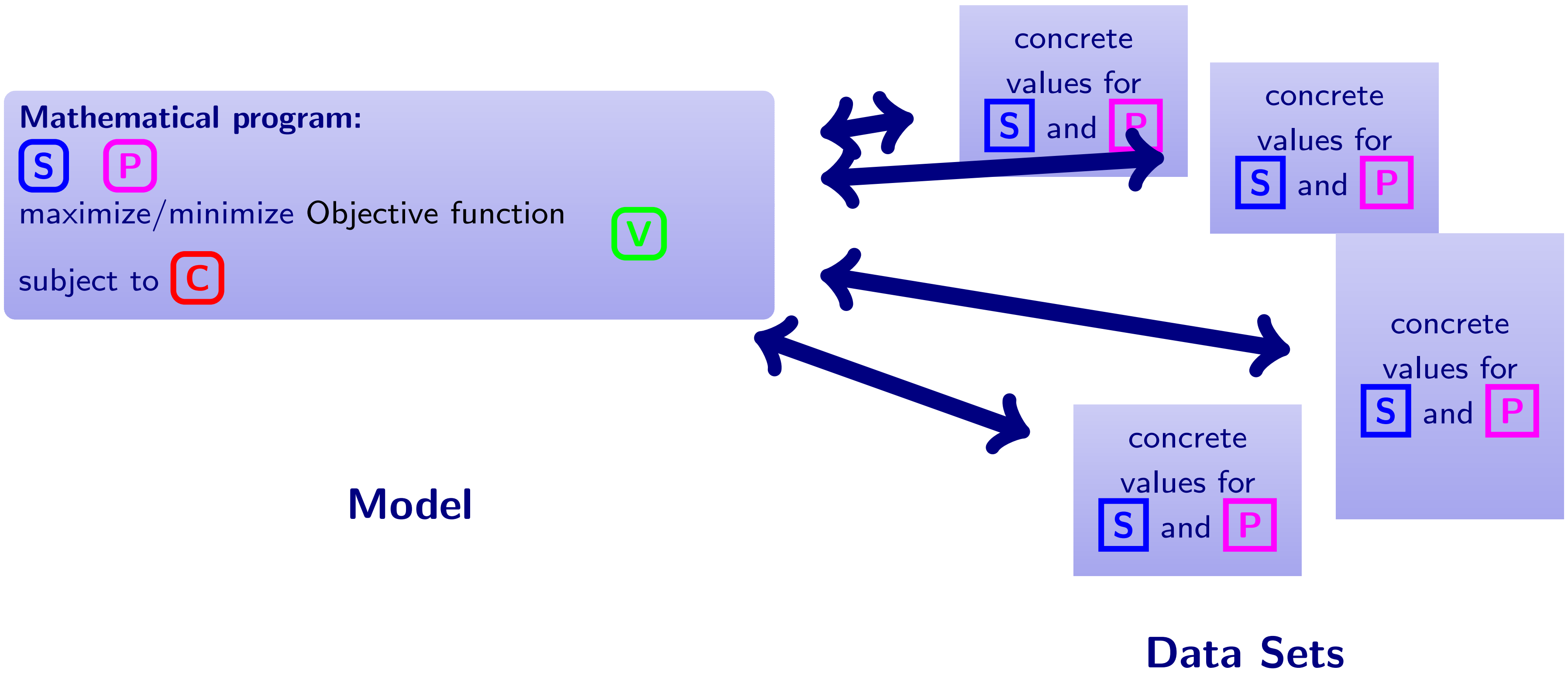
S Sets of relevant elements
(for example: products, cities, machines, types of raw material, ...)

P Parameters: Values specified for (combinations of) elements of the sets
(for example: profits for products, demand for products, distances between cities, capacity of machines, prices of one unit of raw material, ...)

V Variables: Unknowns to be determined
(for example: number of items to produce, number of shops to open in a certain city, decision to buy a certain machine or not, amount of raw material to use, ...)

C Constraints: Relationships that have to hold between variables and parameters
(for example: maximal number of items that can be produced by a machine, minimal number of shops to open, budget for buying raw material, ...)

▷ Mathematical Program: Collection of constraints and variables together with an Objective function to be maximized/minimized



- Input: One data set $\widehat{=}$ Values for sets and parameters.
- Output: Variable assignment such that the objective function value is maximal/minimal and the constraints are respected

1. **Identify variables** **V**

   ➡ Which decisions have to be made?

   ➡ In which numbers are they best represented?

1. **Identify variables** $\boxed{\mathsf{V}}$
   - ➡ Which decisions have to be made?
   - ➡ In which numbers are they best represented?
2. **Identify sets and parameters** $\boxed{\mathsf{S}}$ $\boxed{\mathsf{P}}$
   - ➡ Which objects influence the problem?
   - ➡ Which values define these objects and are relevant?

1. **Identify variables** **V**
   - ➡ Which decisions have to be made?
   - ➡ In which numbers are they best represented?
2. **Identify sets and parameters** **S** **P**
   - ➡ Which objects influence the problem?
   - ➡ Which values define these objects and are relevant?
3. **Identify objective function** **Objective**
   - ➡ Which quantity has to be optimized, and in which direction: minimize or maximize?
   - ➡ How can this quantity be written in terms of the variables and parameters?

1. **Identify variables** **V**
   - ➡ Which decisions have to be made?
   - ➡ In which numbers are they best represented?
2. **Identify sets and parameters** **S** **P**
   - ➡ Which objects influence the problem?
   - ➡ Which values define these objects and are relevant?
3. **Identify objective function** **Objective**
   - ➡ Which quantity has to be optimized, and in which direction: minimize or maximize?
   - ➡ How can this quantity be written in terms of the variables and parameters?
4. **Identify constraints** **C**
   - ➡ Which restrictions have to be taken into account?
   - ➡ How can these restrictions be expressed in terms of variables and parameters?

- ▷ Models, Data and Instances
- ▷ Linear Optimization
  - ➡ Modelling as a linear program
  - ➡ Solving a linear program (graphically, and in princple by the simplex algorithm)
  - ➡ Sensitivity analysis
- ▷ (Mixed) Integer Programming
  - ➡ Modelling as a (mixed) integer program
  - ➡ How to solve a (mixed) integer program (in principle)
- ▷ Combinatorial Optimization
  - ➡ Exemplary problems, algorithms, and runtimes
- ▷ Nonlinear Optimization
  - ➡ Local and global optima, convex optimization
- ▷ Scheduling
- ▷ Lot Sizing
- ▷ Multicriteria Optimization

Fruit	**Bananas**	**Pineapples**
Revenue	\$10000	\$20000
Land use	5a	3a
Time use	4h	7h
Water consumpt.	400l	400l

Available capacities and water resources:

- Land: 50a
- Working time: 70h
- Water supply: 4500l

▷ Question: How much of each fruit should be produced to maximize the profit?

Fruit	**Bananas**	**Pineapples**
Revenue	\$10000	\$20000
Land use	5a	3a
Time use	4h	7h
Water consumpt.	400l	400l

Available capacities and water resources:

- Land: 50a
- Working time: 70h
- Water supply: 4500l

▷ Question: How much of each fruit should be produced to maximize the profit?

➡ Modelled as a linear program:

Fruit	Bananas	Pineapples
Revenue	\$10000	\$20000
Land use	5a	3a
Time use	4h	7h
Water consumpt.	400l	400l

Available capacities and water resources:

- Land: 50a
- Working time: 70h
- Water supply: 4500l

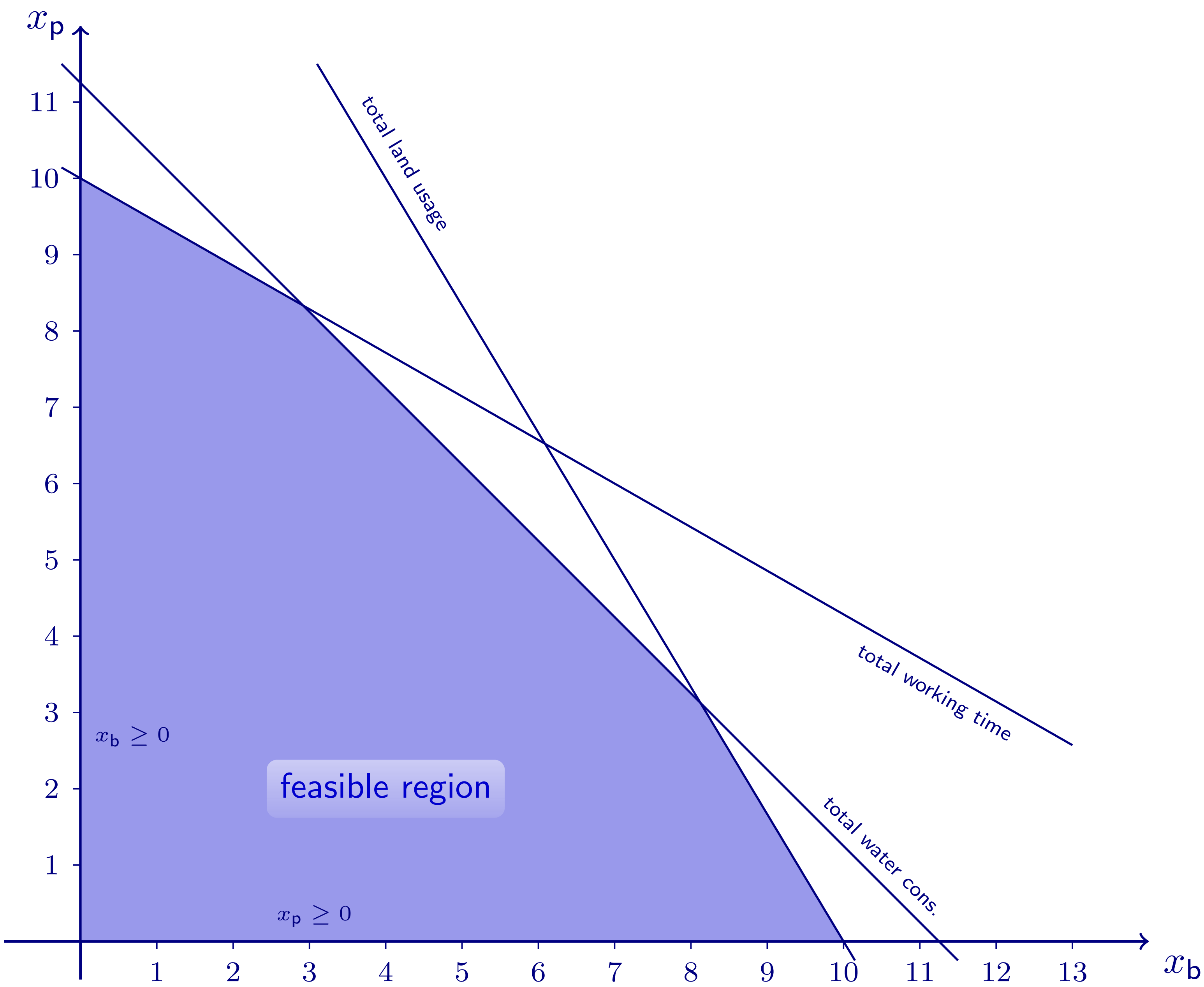
▷ Question: How much of each fruit should be produced to maximize the profit?

➡ Modelled as a linear program:

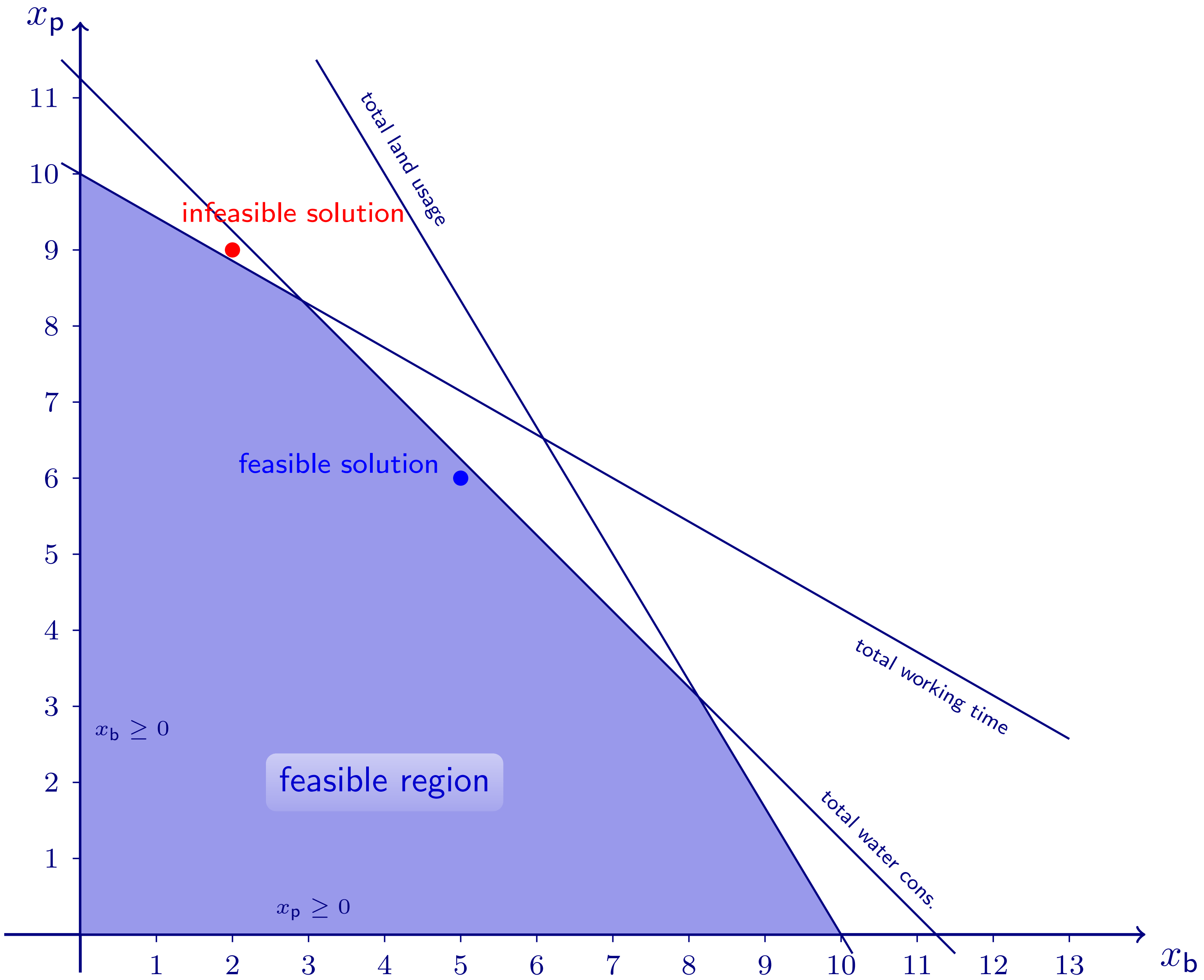
maximize	(total revenue)	$10x_b + 20x_p$			**Objective**
subject to	(total land usage)	$5x_b + 3x_p$	$\leq$	50	C
	(total working time)	$4x_b + 7x_p$	$\leq$	70	
	(total water consumption)	$4x_b + 4x_p$	$\leq$	45	
	(non-negativity)	x_b, x_p	$\geq$	0	V

	x_b	x_p	revenue	total land use	total working time	total water cons.
feasible	5	6	170	43	62	44
feasible	4	7	180	41	65	44
infeasible	2	9	200	37	**71**	44
optimal	0	10	200	30	70	40
			available:	50	70	45

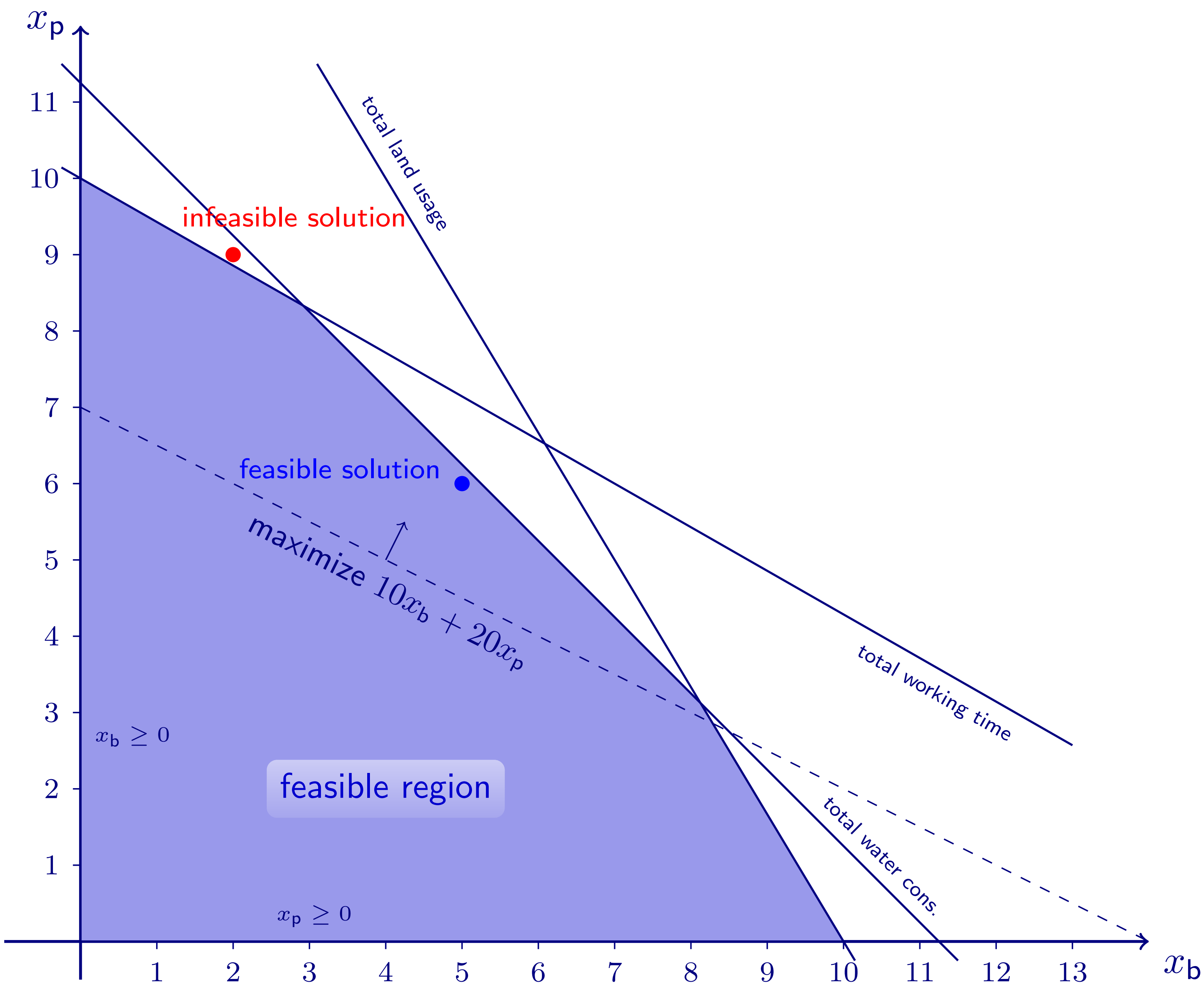
$$
\begin{array}{llrrcl}
\text{maximize} & \text{(total revenue)} & 10x_b + 20x_p & & & \\
\text{subject to} & & \text{(total land usage)} & 5x_b + 3x_p & \leq & 50 \\
& & \text{(total working time)} & 4x_b + 7x_p & \leq & 70 \\
& & \text{(total water consumption)} & 4x_b + 4x_p & \leq & 45 \\
& & \text{(non-negativity)} & x_b, x_p & \geq & 0
\end{array}
$$



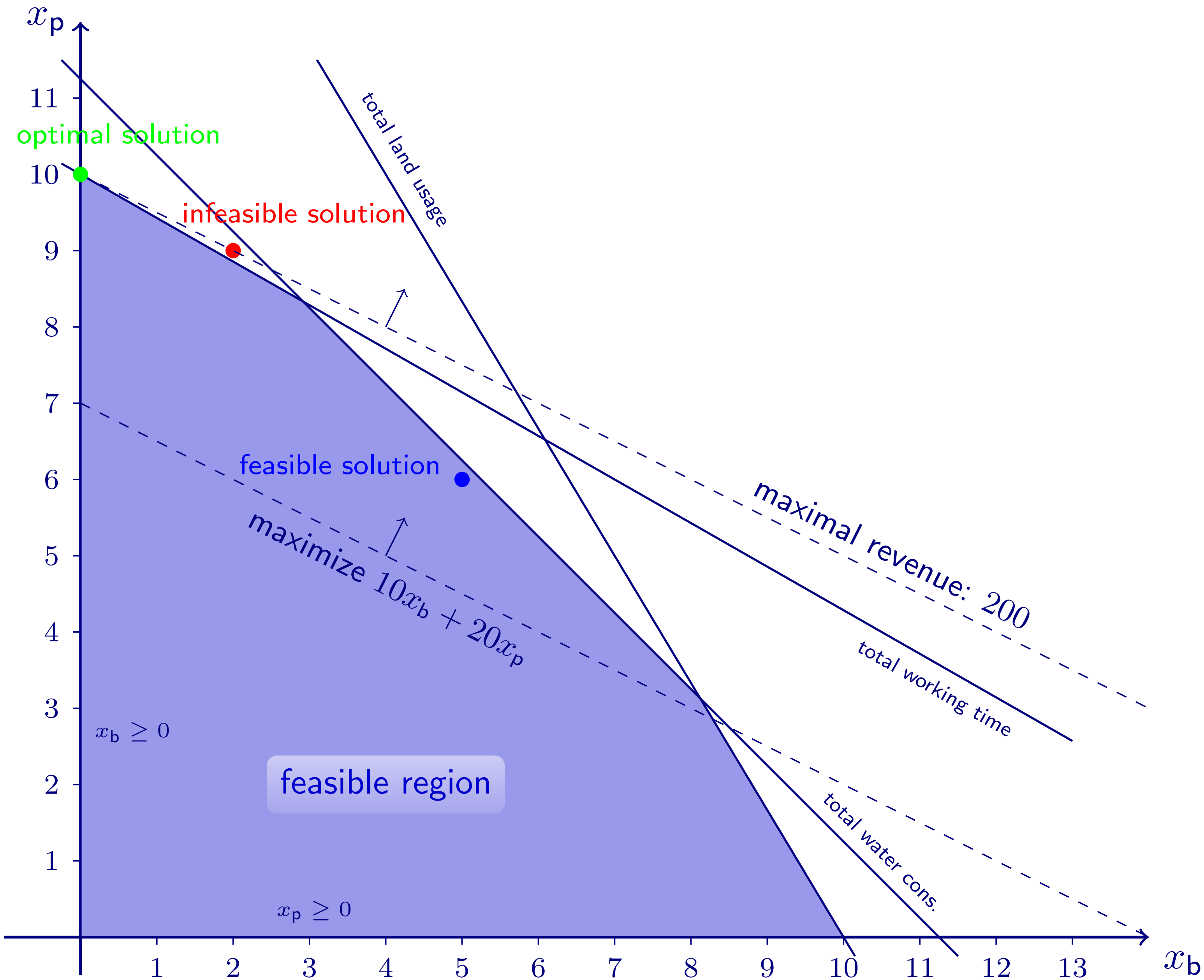
x_p
x_b
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13
total land usage
total working time
total water cons.
$x_b \geq 0$
$x_p \geq 0$
feasible region



x_p
x_b
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13
total land usage
infeasible solution
feasible solution
total working time
total water cons.
$x_b \geq 0$
$x_p \geq 0$
feasible region



x_p
x_b
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13
total land usage
infeasible solution
feasible solution
maximize $10x_b + 20x_p$
total working time
$x_b \geq 0$
feasible region
total water cons.
$x_p \geq 0$



x_p
optimal solution
infeasible solution
total land usage
feasible solution
maximize $10x_b + 20x_p$
maximal revenue: 200
total working time
$x_b \geq 0$
feasible region
total water cons.
$x_p \geq 0$
x_b
1 2 3 4 5 6 7 8 9 10 11
1 2 3 4 5 6 7 8 9 10 11 12 13

- Geometric solving only works for at most 2 (maybe 3) variables

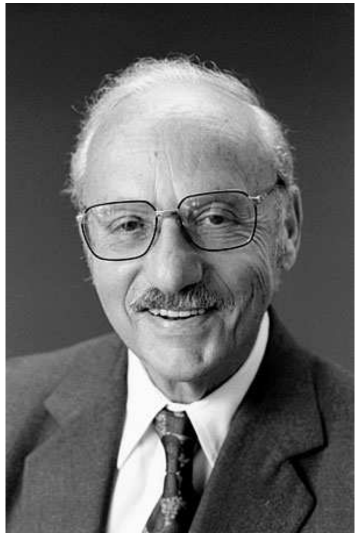
- Geometric solving only works for at most 2 (maybe 3) variables
- More generally: Simplex Algorithm
- Idea: Jump from vertex to vertex in the direction of the objective vector until an optimal vertex is reached

- Geometric solving only works for at most 2 (maybe 3) variables
- More generally: Simplex Algorithm
- Idea: Jump from vertex to vertex in the direction of the objective vector until an optimal vertex is reached
- More precisely:
  - Search for some vertex (basic feasible solution)
  - If there is a neighbouring vertex with a better objective...
  - ...jump to this vertex and repeat
  - Otherwise: stop – an optimal solution is reached!

- ▷ Geometric solving only works for at most 2 (maybe 3) variables
- ▷ More generally: Simplex Algorithm
- ▷ Idea: Jump from vertex to vertex in the direction of the objective vector until an optimal vertex is reached
- ▷ More precisely:
  - Search for some vertex (basic feasible solution)
  - If there is a neighbouring vertex with a better objective...
  - ...jump to this vertex and repeat
  - Otherwise: stop – an optimal solution is reached!
- ▷ Special cases:
  - No starting vertex can be found ➡ Problem is infeasible
  - No neighbouring vertex in some objective-increasing direction ➡ Problem is unbounded

LP algorithm zoo

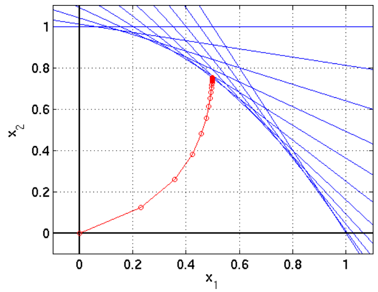
- Simplex Algorithm
  - ➡ developed by George B. Dantzig in 1947
  - ➡ Variants:
    - Dual Simplex Algorithm
    - Network Simplex

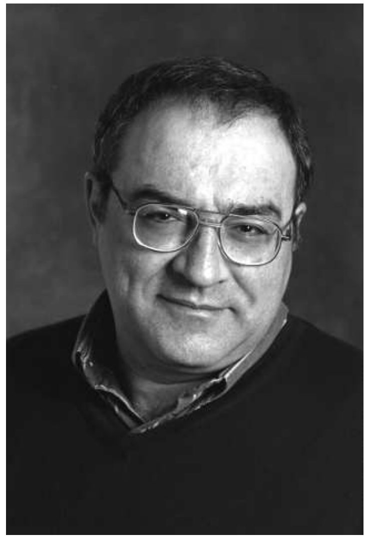


George Bernard Dantzig
(1914–2005)

- Ellipsoid Method
  - ➡ developed by L.G. Khachiyan in 1979
  - ➡ theoretically fast (polynomial), but practically useless

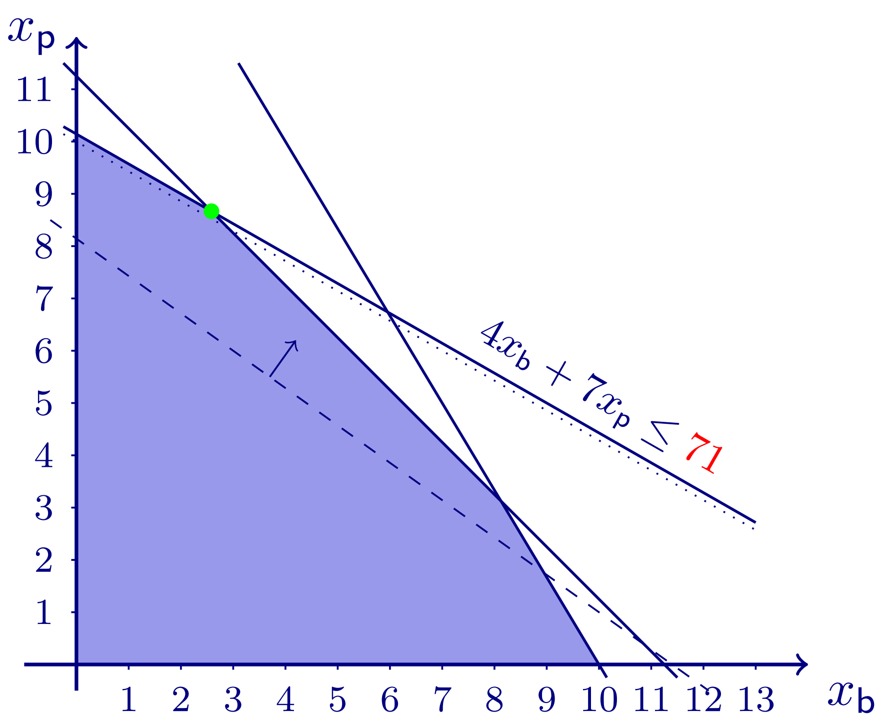
- Interior Point Methods
  - ➡ Barrier Method (Karmarkar, 1984)
  - ➡ theoretically and practically fast
  - ➡ used for large-scale LPs





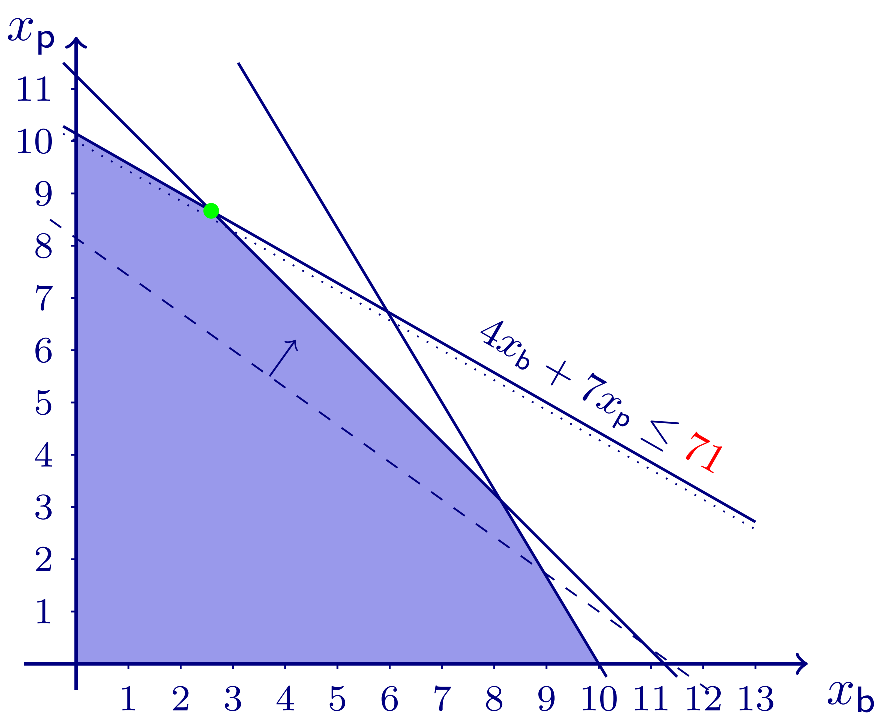
Leonid Genrikhovich Khachiyan
(1952–2005)

The shadow price of a constraint is the rate of change in the objective function per unit increase of the constraint's right-hand side



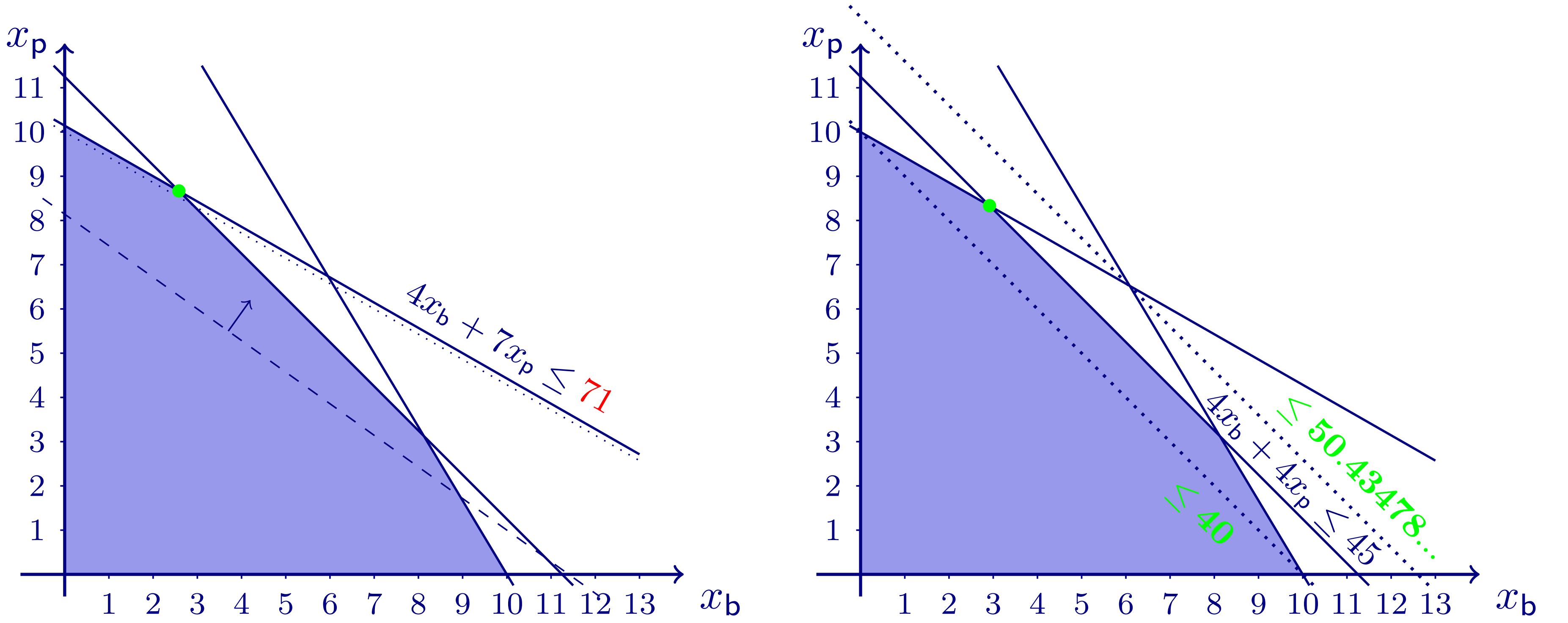
The shadow price of a constraint is the rate of change in the objective function per unit increase of the constraint's right-hand side

- Shadow prices for non-binding constraints are always 0

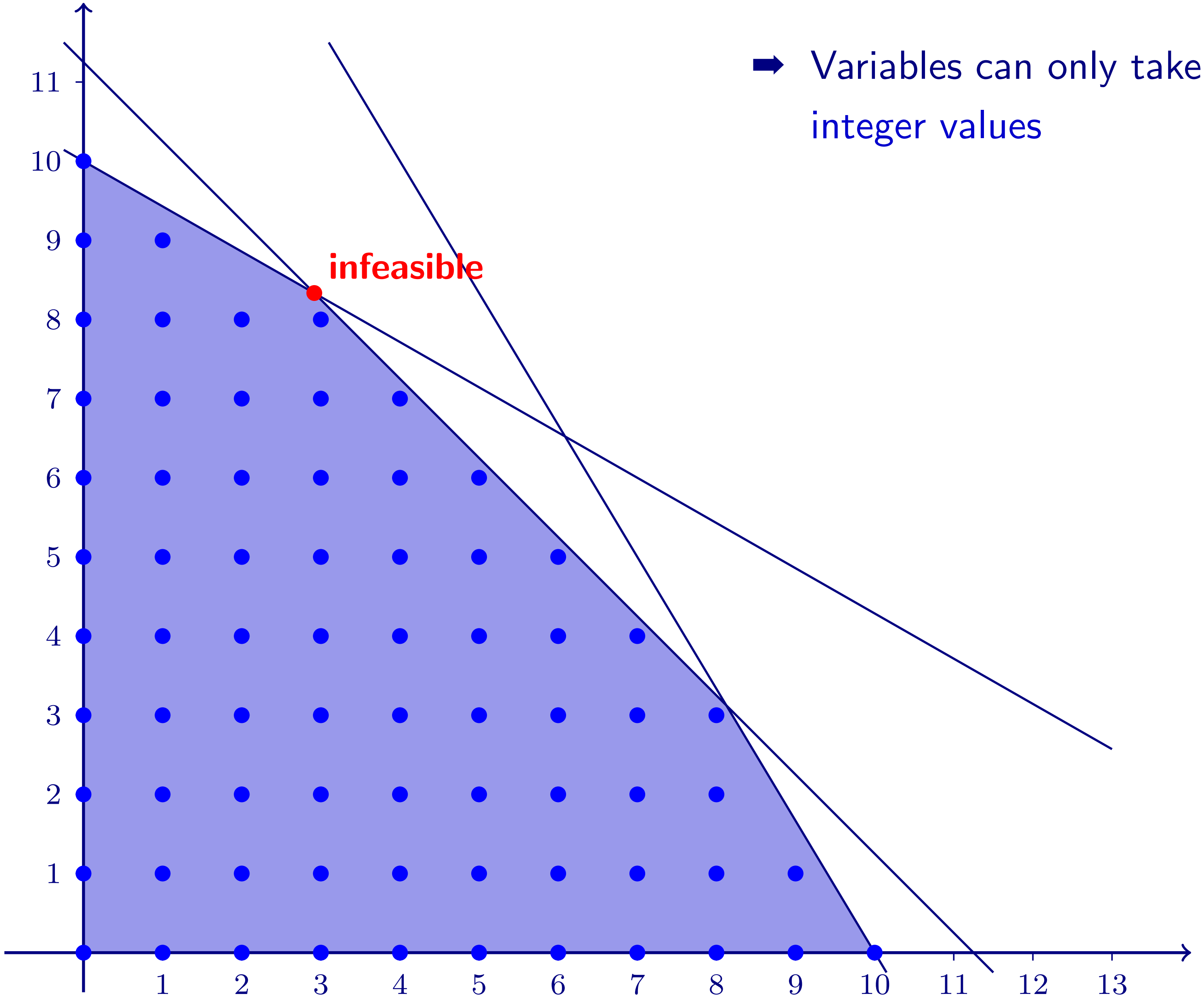


The shadow price of a constraint is the rate of change in the objective function per unit increase of the constraint's right-hand side

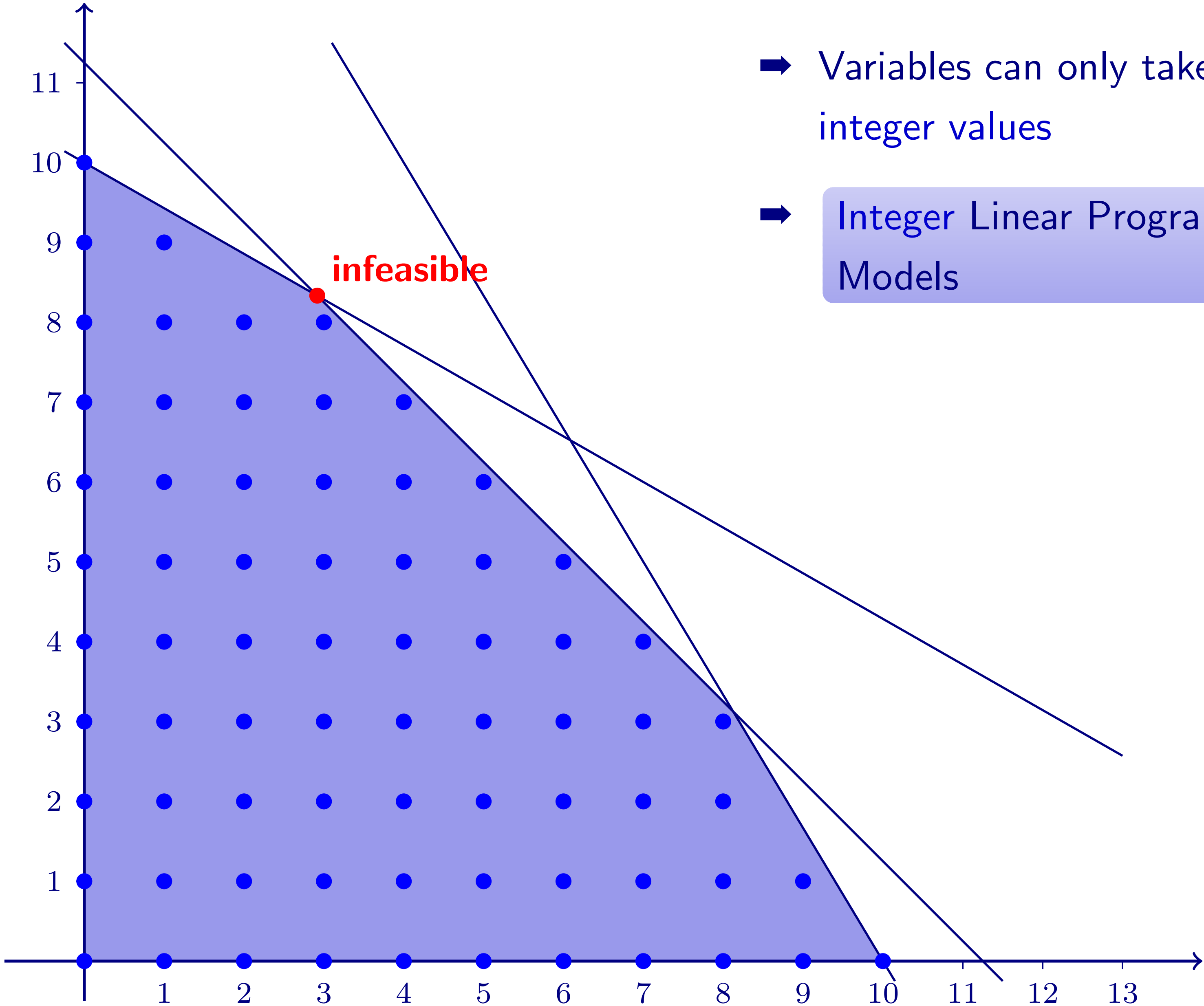
- ▷ Shadow prices for non-binding constraints are always 0
- ▷ Shadow price for a constraint is only valid if the RHS is in a certain range



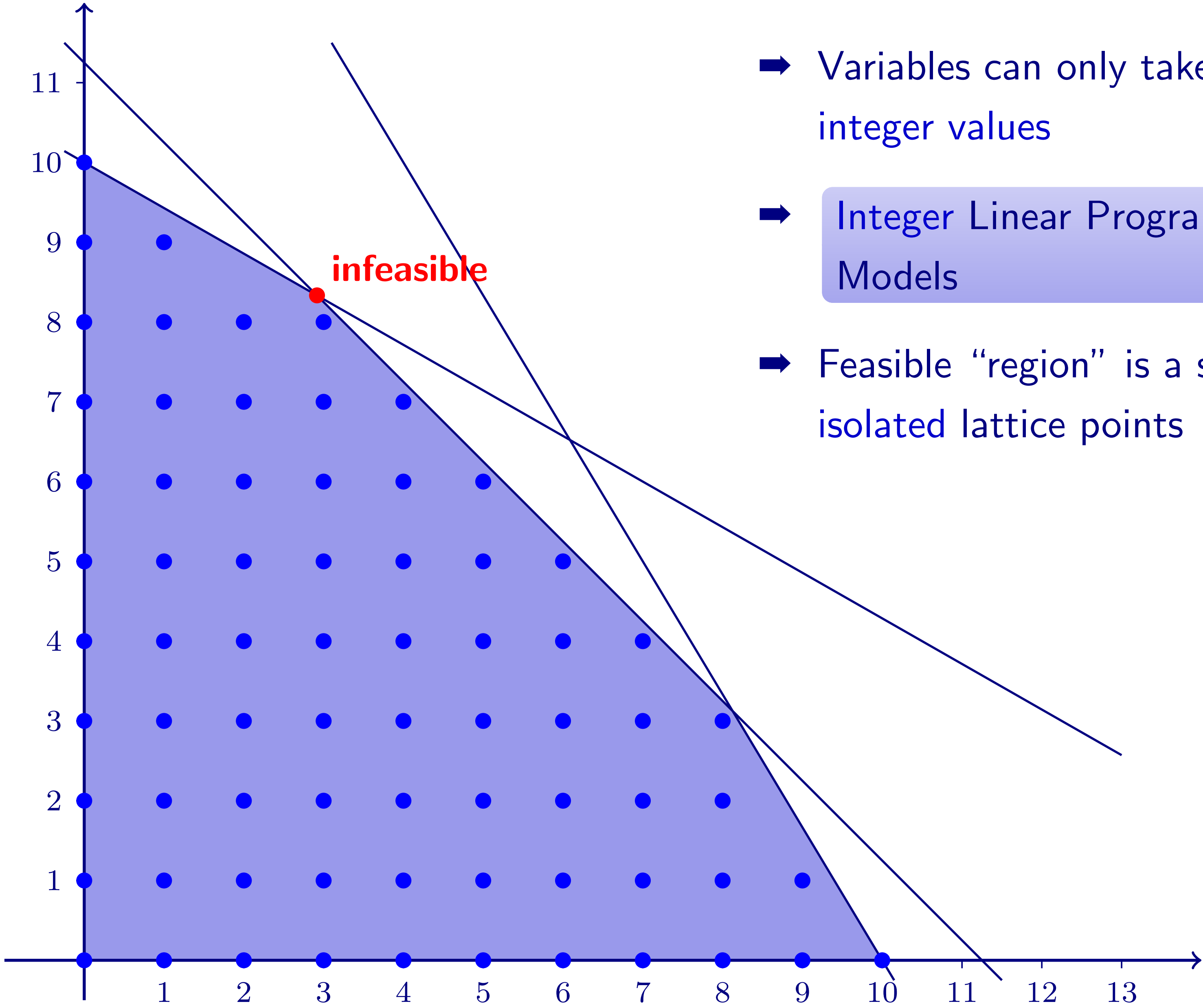
- ▷ Models, Data and Instances
- ▷ Linear Optimization
  - ➡ Modelling as a linear program
  - ➡ Solving a linear program (graphically, and in princple by the simplex algorithm)
  - ➡ Sensitivity analysis
- ▷ (Mixed) Integer Programming
  - ➡ Modelling as a (mixed) integer program
  - ➡ How to solve a (mixed) integer program (in principle)
- ▷ Combinatorial Optimization
  - ➡ Exemplary problems, algorithms, and runtimes
- ▷ Nonlinear Optimization
  - ➡ Local and global optima, convex optimization
- ▷ Scheduling
- ▷ Lot Sizing
- ▷ Multicriteria Optimization



➡ Variables can only take integer values



- Variables can only take integer values
- Integer Linear Programming Models



- ➡ Variables can only take integer values
- ➡ Integer Linear Programming Models
- ➡ Feasible "region" is a set of isolated lattice points

LP-relaxation

maximize/minimize $\sum_{j=1}^{n} c_j x_j$ **Objective function**

subject to $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ for all $i = 1, \ldots, m$ **C**

$\ell_j \leq x_j \leq u_j$ for all $j = 1, \ldots, n$ **V**

➡ Integer Program: x_j integer for all $j = 1, \ldots, n$

➡ Mixed Integer Program: x_j integer for all $j = 1, \ldots, \ell$ $(\ell < n)$

➡ Binary variables: $x_j \in \{0, 1\}$ for all $j = 1, \ldots, \ell$

- ▷ Trigger a yes/no-decision if some quantity reaches some value V
  - ➡ binary variable $y \in \{0, 1\}$, meaning: $y = 1 \Leftrightarrow$ “yes”
  - ➡ (...linear expression for quantity...) $\leq V + M \cdot y$ (M: large number)

- ▷ Trigger a yes/no-decision if some quantity reaches some value V
  - ➡ binary variable $y \in \{0, 1\}$, meaning: $y = 1 \Leftrightarrow$ "yes"
  - ➡ (...linear expression for quantity...) $\leq V + M \cdot y$ (M: large number)
- ▷ Logical constraints: if decision A is taken, then also decision B has to be taken
  - ➡ binary variables $y_{\mathsf{A}}, y_{\mathsf{B}} \in \{0, 1\}$, meaning: $y_* = 1 \Leftrightarrow$ "yes" for decision $*$
  - ➡ $y_{\mathsf{A}} \leq y_{\mathsf{B}}$

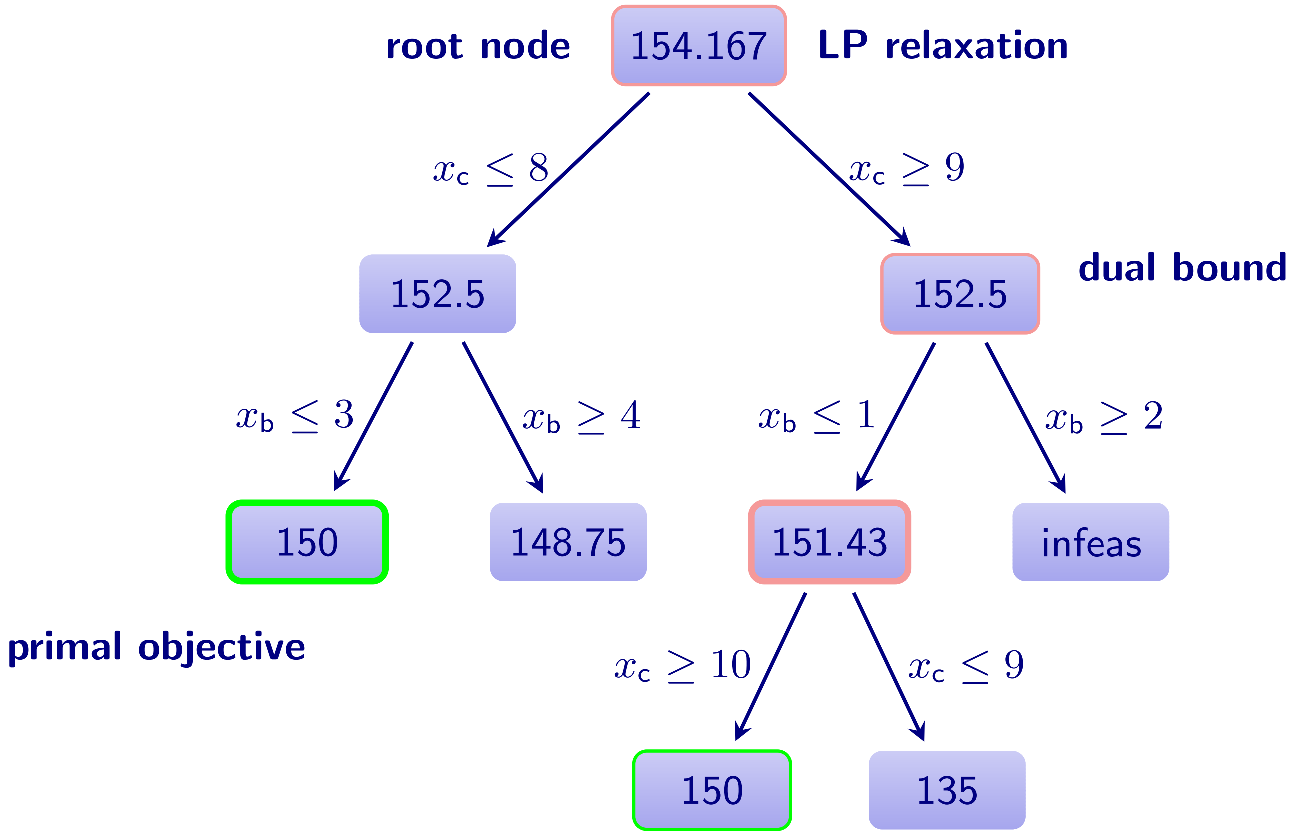
- ▷ Trigger a yes/no-decision if some quantity reaches some value V
  - ➡ binary variable $y \in \{0,1\}$, meaning: $y = 1 \Leftrightarrow$ "yes"
  - ➡ (...linear expression for quantity...) $\leq V + M \cdot y$ (M: large number)
- ▷ Logical constraints: if decision A is taken, then also decision B has to be taken
  - ➡ binary variables $y_\mathsf{A}, y_\mathsf{B} \in \{0,1\}$, meaning: $y_* = 1 \Leftrightarrow$ "yes" for decision $*$
  - ➡ $y_\mathsf{A} \leq y_\mathsf{B}$
- ▷ Set packing constraints:
  - ➡ choose at most/at least/exactly one of the binary variables $y_1, \ldots, y_n$
  - ➡ $y_1 + y_2 + \ldots + y_n \leq 1$ / ≥ 1 / $= 1$

- ▷ Trigger a yes/no-decision if some quantity reaches some value V
  - ➡ binary variable $y \in \{0, 1\}$, meaning: $y = 1 \Leftrightarrow$ "yes"
  - ➡ (...linear expression for quantity...) $\leq V + M \cdot y$ (M: large number)
- ▷ Logical constraints: if decision A is taken, then also decision B has to be taken
  - ➡ binary variables $y_\mathsf{A}, y_\mathsf{B} \in \{0, 1\}$, meaning: $y_* = 1 \Leftrightarrow$ "yes" for decision $*$
  - ➡ $y_\mathsf{A} \leq y_\mathsf{B}$
- ▷ Set packing constraints:
  - ➡ choose at most/at least/exactly one of the binary variables $y_1, \ldots, y_n$
  - ➡ $y_1 + y_2 + \ldots + y_n \leq 1$ / ≥ 1 / $= 1$
- ▷ Lots of other types...

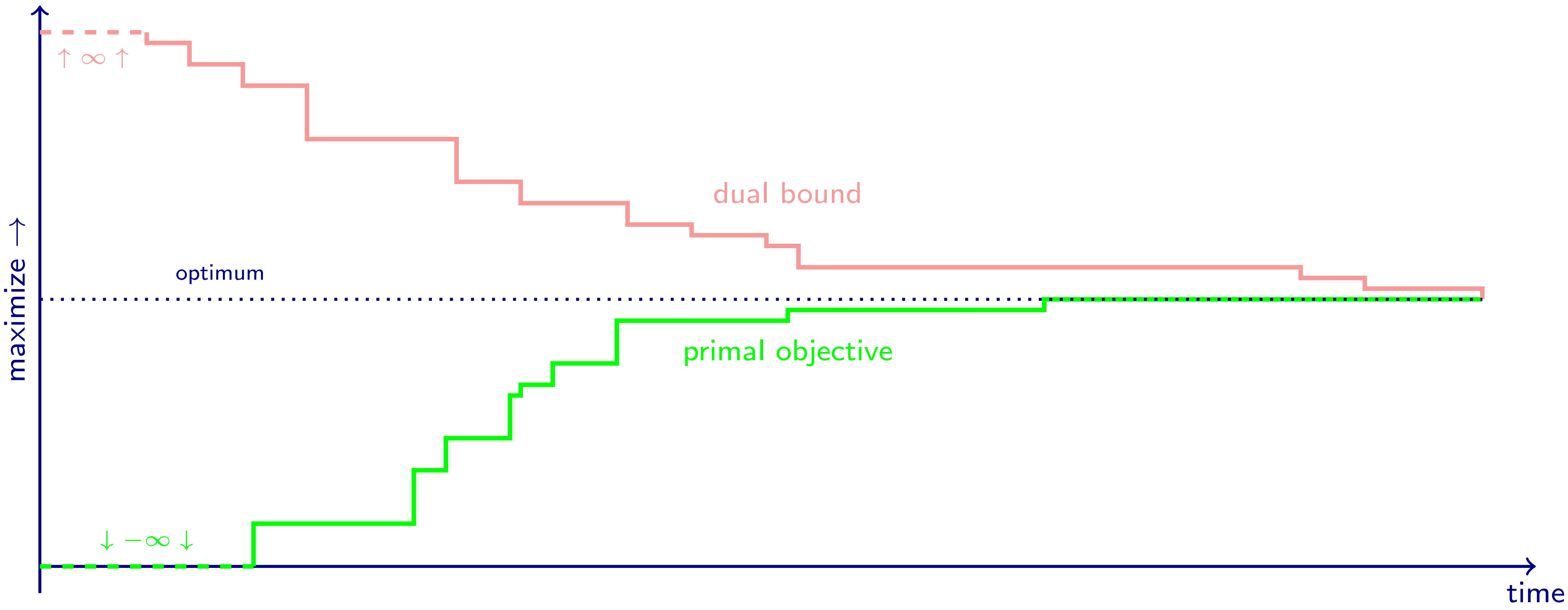
- Start by solving the LP relaxation

- Start by solving the LP relaxation
- If the LP-optimum is not integer: split the problem into two subproblems and iterate

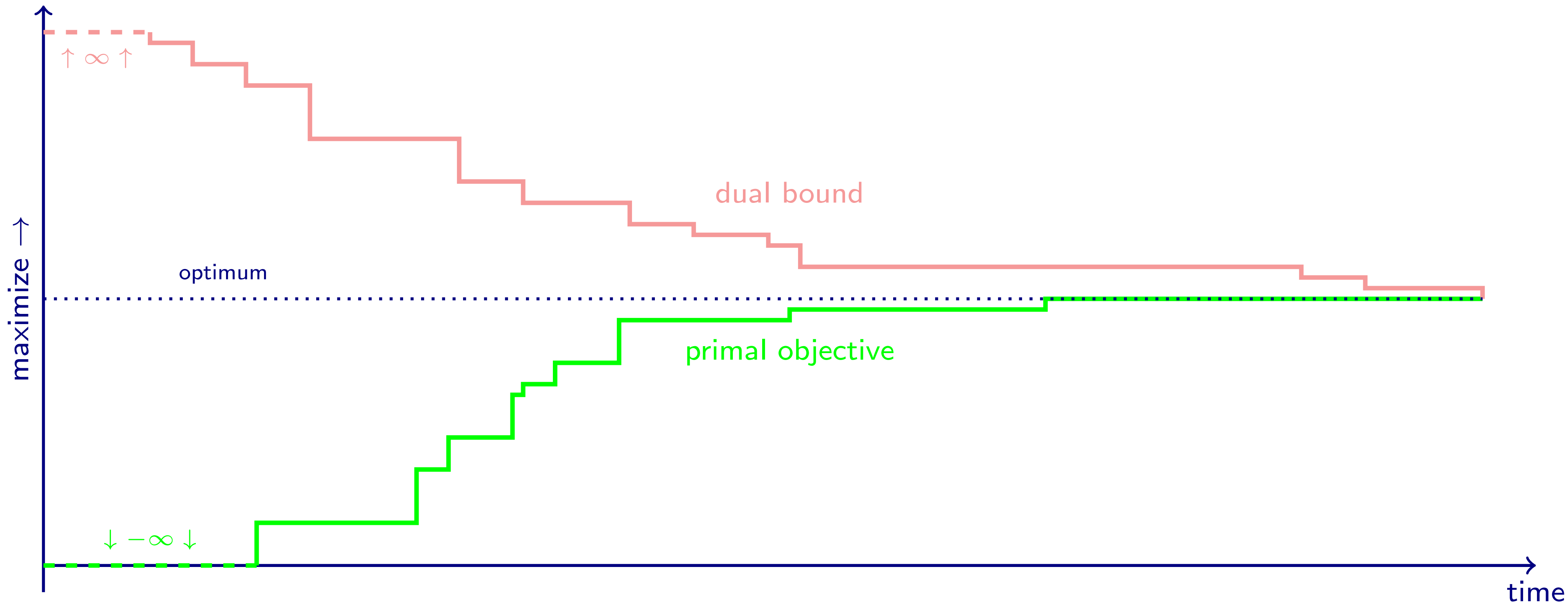
- ▷ Start by solving the LP relaxation
- ▷ If the LP-optimum is not integer: split the problem into two subproblems and iterate
- ➡ Branch-and-bound tree (example for maximization problem):



▷ The (absolute) gap during branch and bound: gap := bestdual − bestprimal



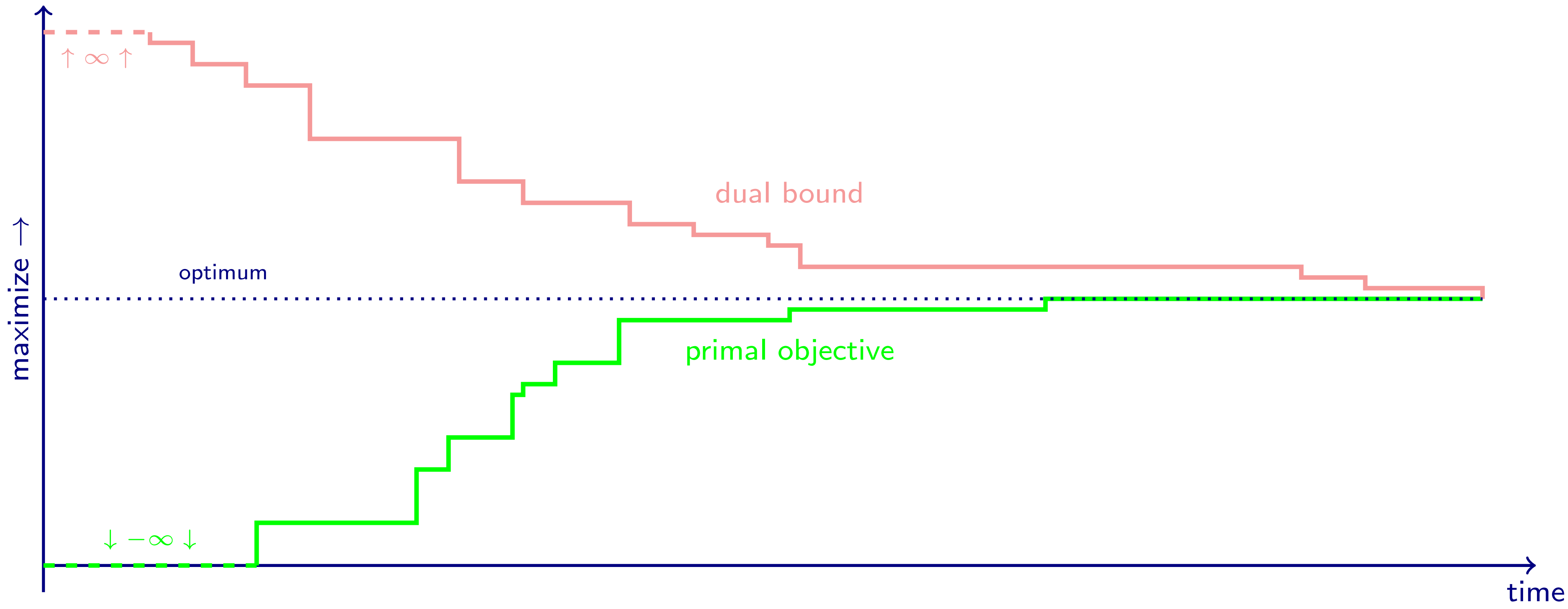
- The (absolute) gap during branch and bound: gap := bestdual − bestprimal



- Stop traversing the tree, if the gap is 0, i.e. the value of the best primal solution and the dual bound coincide

Gap

- The (absolute) gap during branch and bound: gap $:=$ bestdual $-$ bestprimal



- Stop traversing the tree, if the gap is 0, i.e. the value of the best primal solution and the dual bound coincide
- In practise (and for large-scale MIPs): stop traversing the tree already if the relative gap $\dfrac{\text{gap}}{|\,\text{bestdual}\,|}$ is below a certain target (e.g. 5%, 1%, 0.5%, ...)

- Models, Data and Instances
- Linear Optimization
  - Modelling as a linear program
  - Solving a linear program (graphically, and in princple by the simplex algorithm)
  - Sensitivity analysis
- (Mixed) Integer Programming
  - Modelling as a (mixed) integer program
  - How to solve a (mixed) integer program (in principle)
- Combinatorial Optimization
  - Exemplary problems, algorithms, and runtimes
- Nonlinear Optimization
  - Local and global optima, convex optimization
- Scheduling
- Lot Sizing
- Multicriteria Optimization

- Combinatorial optimization problems:
  - Finite, but huge number of feasible solutions
    - ➡ Complete enumeration is not an option

- ▷ Combinatorial optimization problems:
  - Finite, but huge number of feasible solutions
    - ➡ Complete enumeration is not an option
- ▷ Problems can often be solved with integer programming models

▷ Combinatorial optimization problems:

- Finite, but huge number of feasible solutions

  ➡ Complete enumeration is not an option

▷ Problems can often be solved with integer programming models

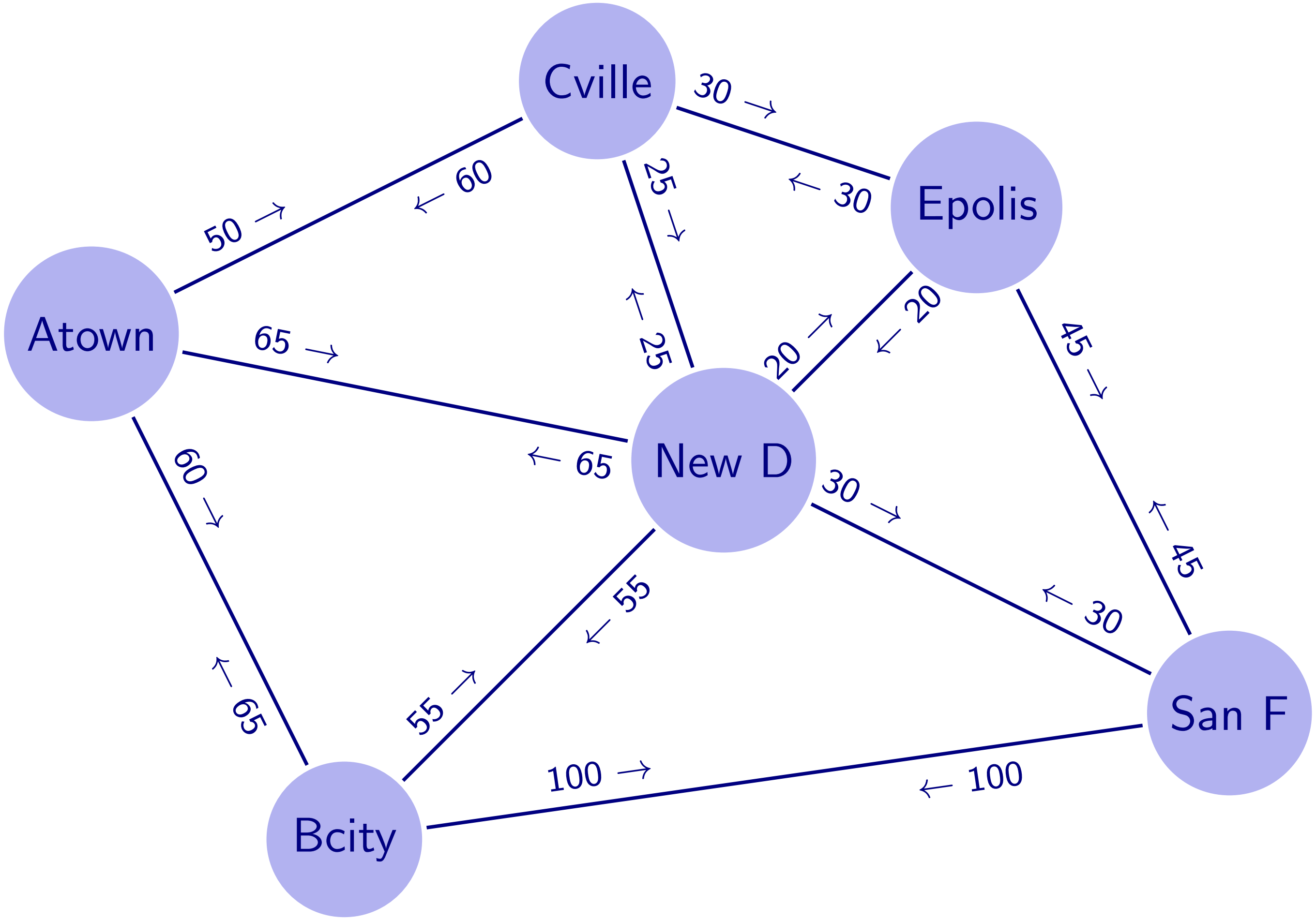
▷ Usually, other methods are more efficient:

- Specially designed algorithms, approximation algorithms
- Primal/dual methods, combining IP with heuristics

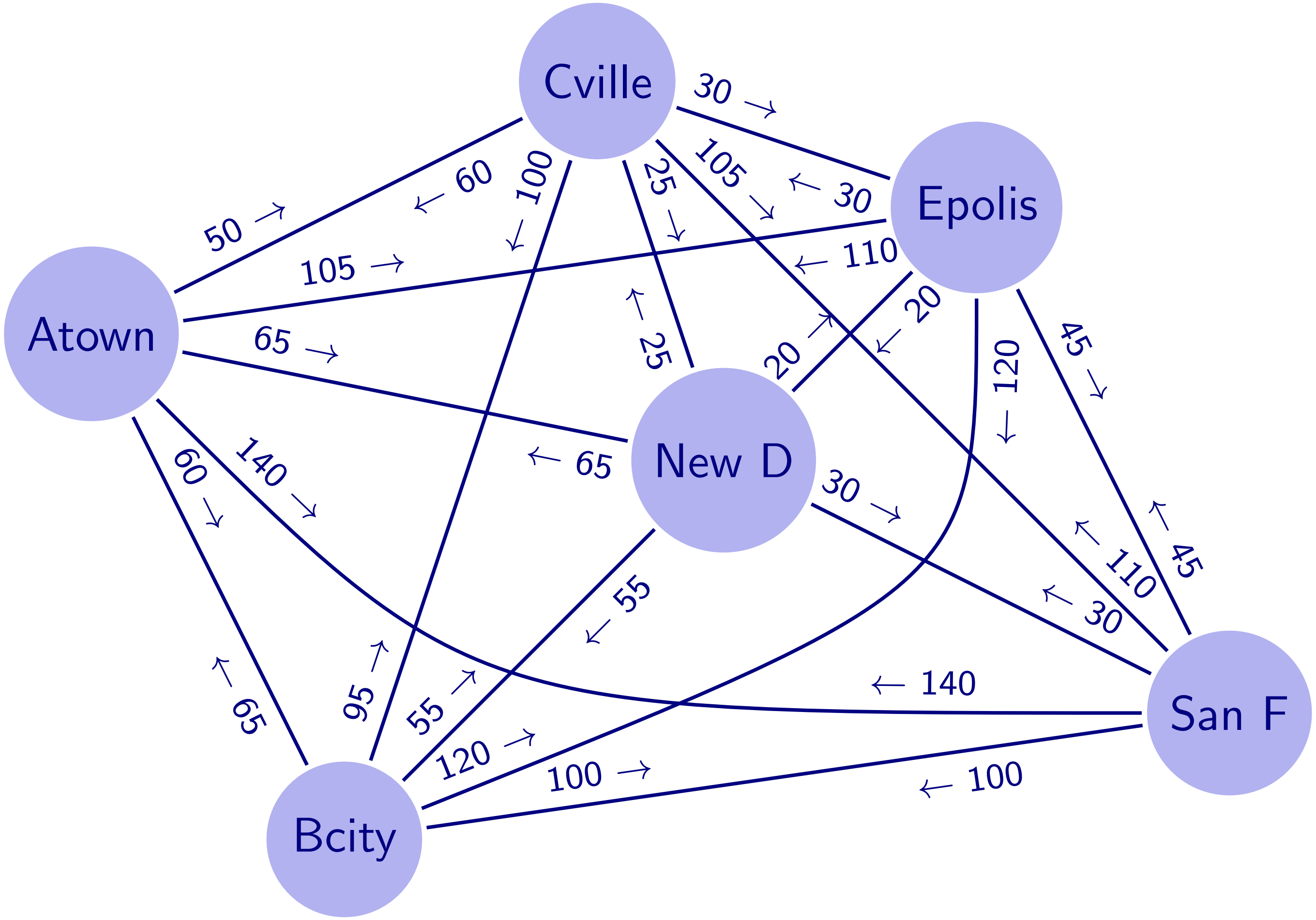
- Combinatorial optimization problems:
  - Finite, but huge number of feasible solutions
    - ➡ Complete enumeration is not an option
- Problems can often be solved with integer programming models
- Usually, other methods are more efficient:
  - Specially designed algorithms, approximation algorithms
  - Primal/dual methods, combining IP with heuristics
- Examples:
  - Travelling Salesman Problem (TSP)
  - Minimum Spanning Tree (MST)
  - Shortest Path Problem (SPP)
  - Network Flow, Knapsack Problem, Bin Packing, Stable Set Problem, ...

Problem formulation: Given a set of cities together with travel times to travel from every city to every other, find a tour leading through every city such that the total travel time is minimized.

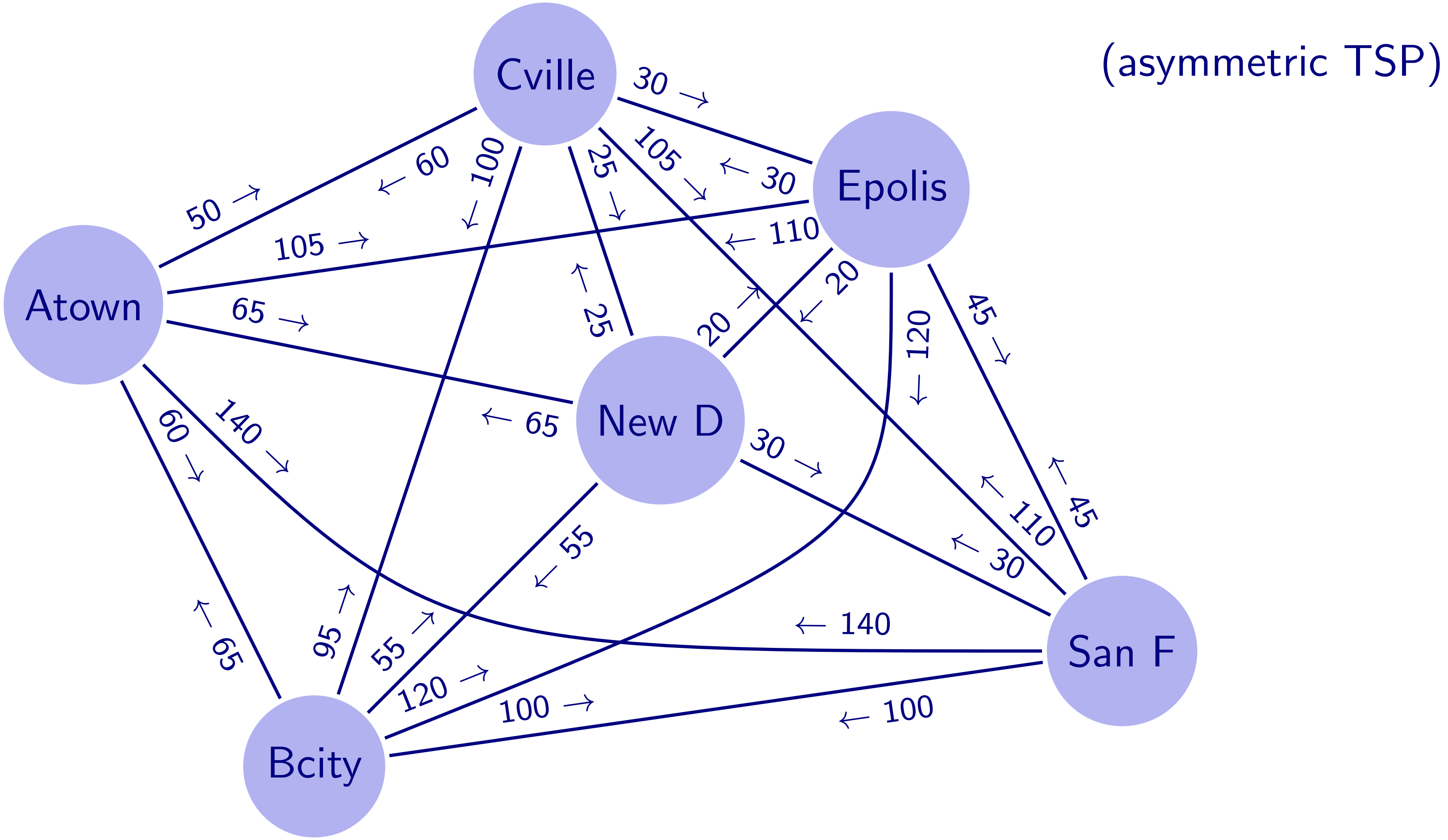
Problem formulation: Given a set of cities together with travel times to travel from every city to every other, find a tour leading through every city such that the total travel time is minimized.



Problem formulation: Given a set of cities together with travel times to travel from every city to every other, find a tour leading through every city such that the total travel time is minimized.



Problem formulation: Given a set of cities together with travel times to travel from every city to every other, find a tour leading through every city such that the total travel time is minimized.



▷ Combinatorial explosion:

# cities	# possible tours	time to try out all tours
5	24	0.012 milliseconds
10	362,880	0.18 seconds
15	87,178,291,200	12 hours
20	121,645,100,408,832,000	1927 years

- Combinatorial explosion:

# cities	# possible tours	time to try out all tours
5	24	0.012 milliseconds
10	362,880	0.18 seconds
15	87,178,291,200	12 hours
20	121,645,100,408,832,000	1927 years

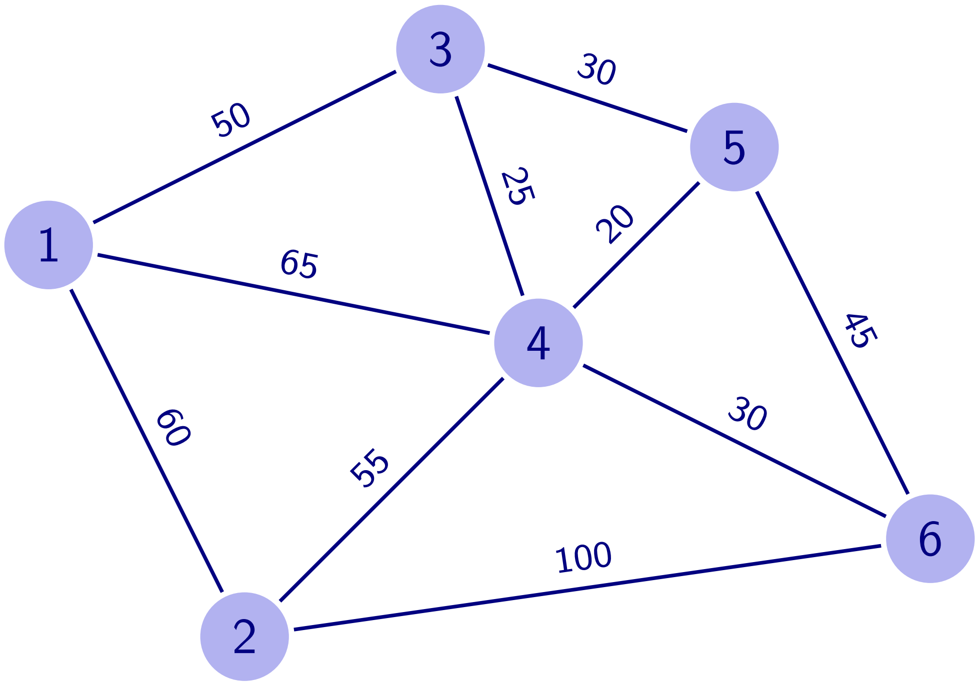
- TSP can be formulated as a binary integer program
  - ➡ Exponentially many (subtour elimination) constraints
  - ➡ Dual method to provide upper bounds for the solution

- ▷ Combinatorial explosion:

# cities	# possible tours	time to try out all tours
5	24	0.012 milliseconds
10	362,880	0.18 seconds
15	87,178,291,200	12 hours
20	121,645,100,408,832,000	1927 years

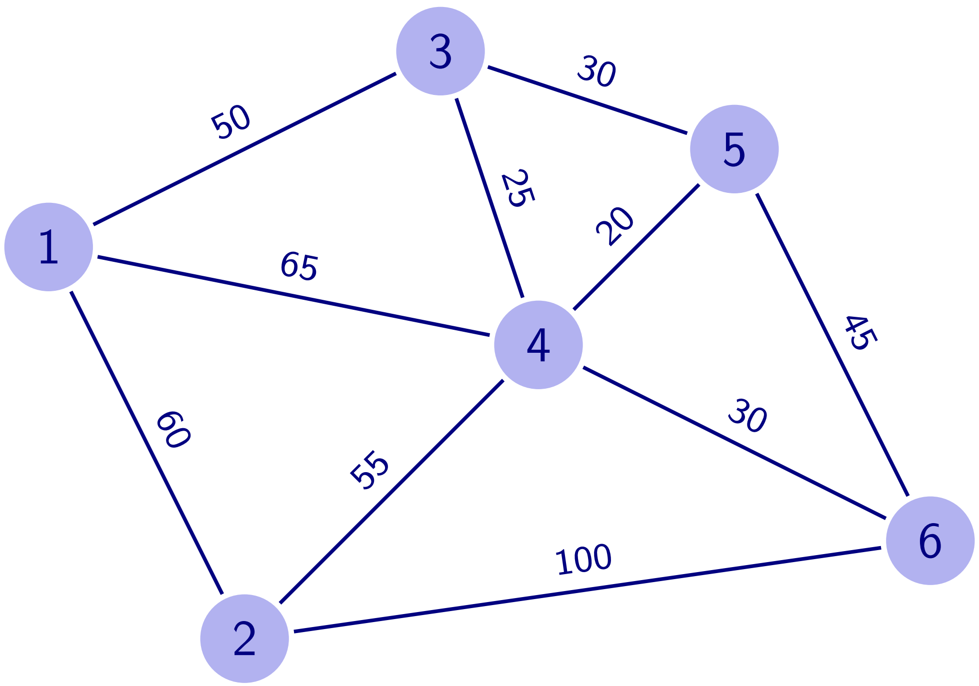
- ▷ TSP can be formulated as a binary integer program
  - ➡ Exponentially many (subtour elimination) constraints
  - ➡ Dual method to provide upper bounds for the solution
- ▷ Approximation algorithms and heuristics
  - ➡ Graph algorithms
  - ➡ Primal methods to find (good) feasible solutions

▷ Given a graph $G = (V, E)$ with non-negative edge-weights w_e for all $e \in E$...



▷ Given a graph $G = (V, E)$ with non-negative edge-weights w_e for all $e \in E$...

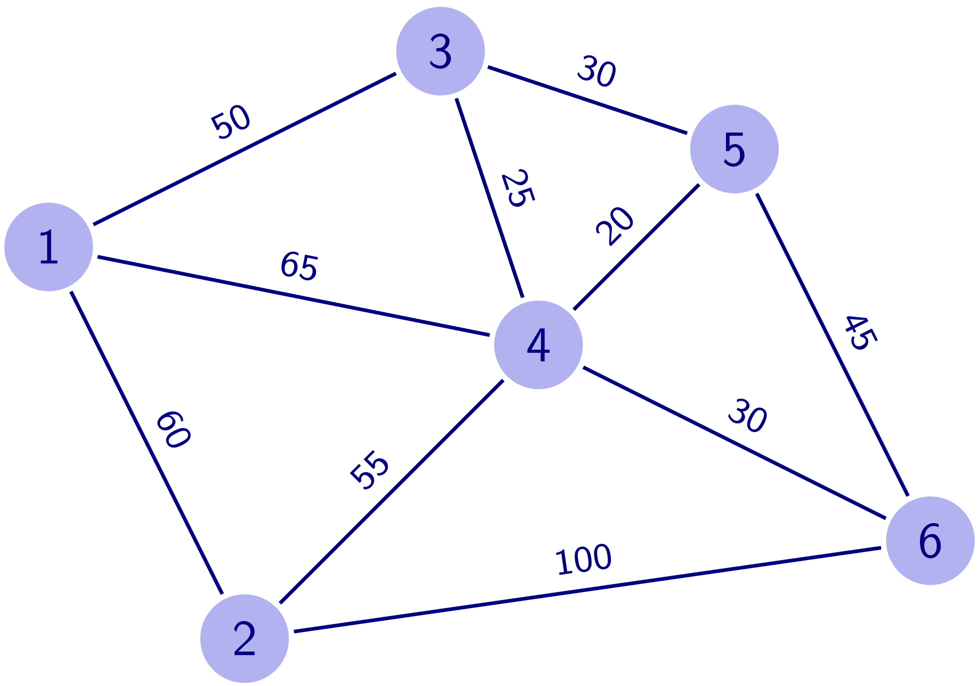
...find a minimum spanning tree for G



▷ Given a graph $G = (V, E)$ with non-negative edge-weights w_e for all $e \in E$...

...find a minimum spanning tree for G, that is: a subset E' of the edges such that

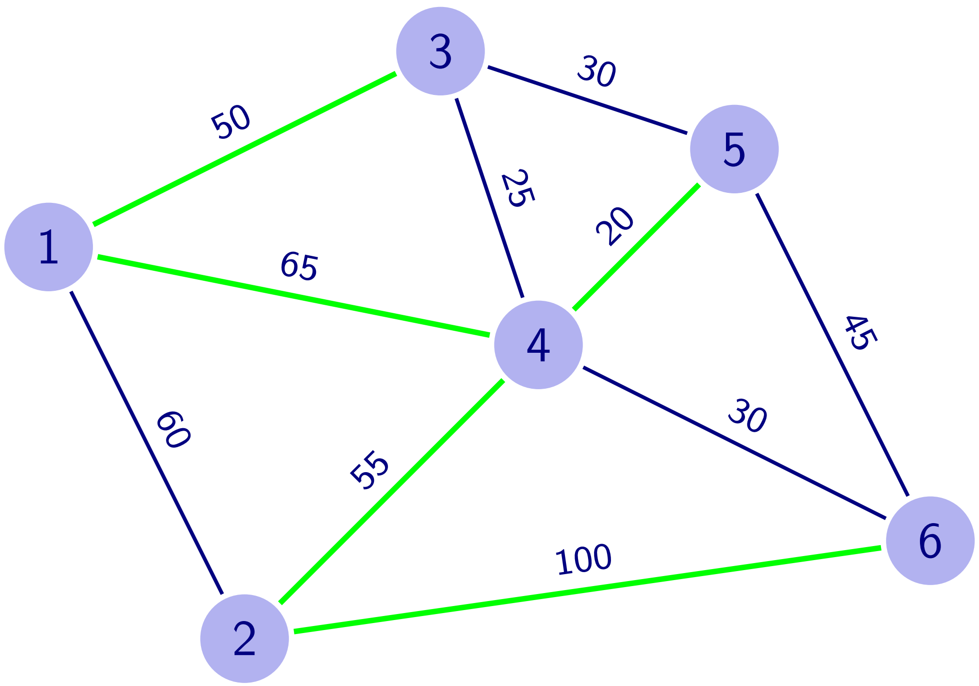
- the edges in E' form a tree (connected and no cycles)
- all vertices of G are in the tree
- the total weight of the tree edges is minimal



▷ Given a graph $G = (V, E)$ with non-negative edge-weights w_e for all $e \in E$...

...find a minimum spanning tree for G, that is: a subset E' of the edges such that

- the edges in E' form a tree (connected and no cycles)
- all vertices of G are in the tree
- the total weight of the tree edges is minimal

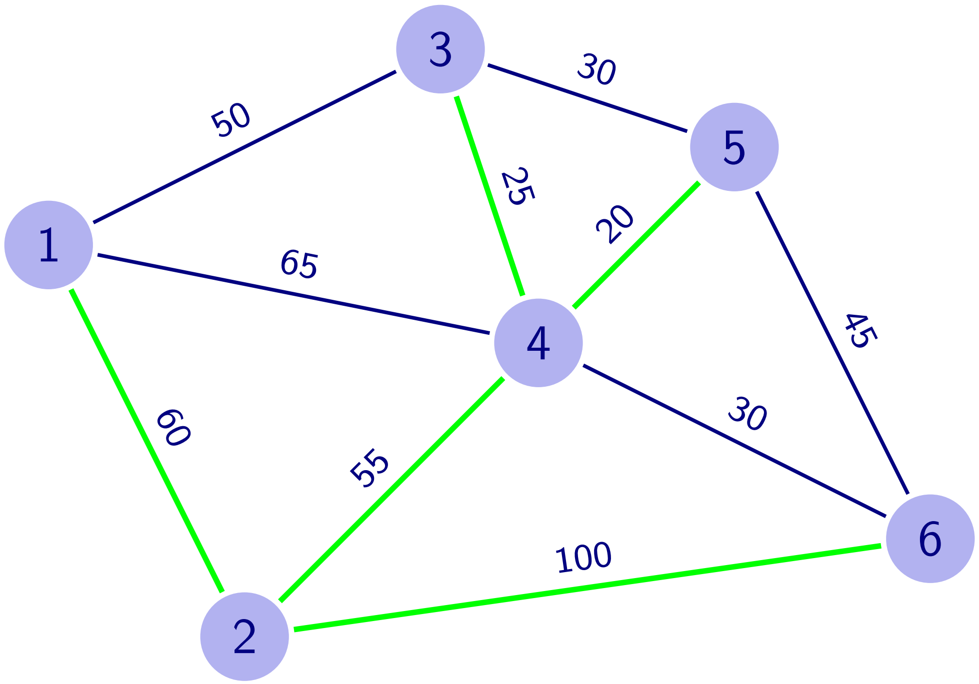


➡ total weight: 290

▷ Given a graph $G = (V, E)$ with non-negative edge-weights w_e for all $e \in E$...

...find a minimum spanning tree for G, that is: a subset E' of the edges such that

- the edges in E' form a tree (connected and no cycles)
- all vertices of G are in the tree
- the total weight of the tree edges is minimal

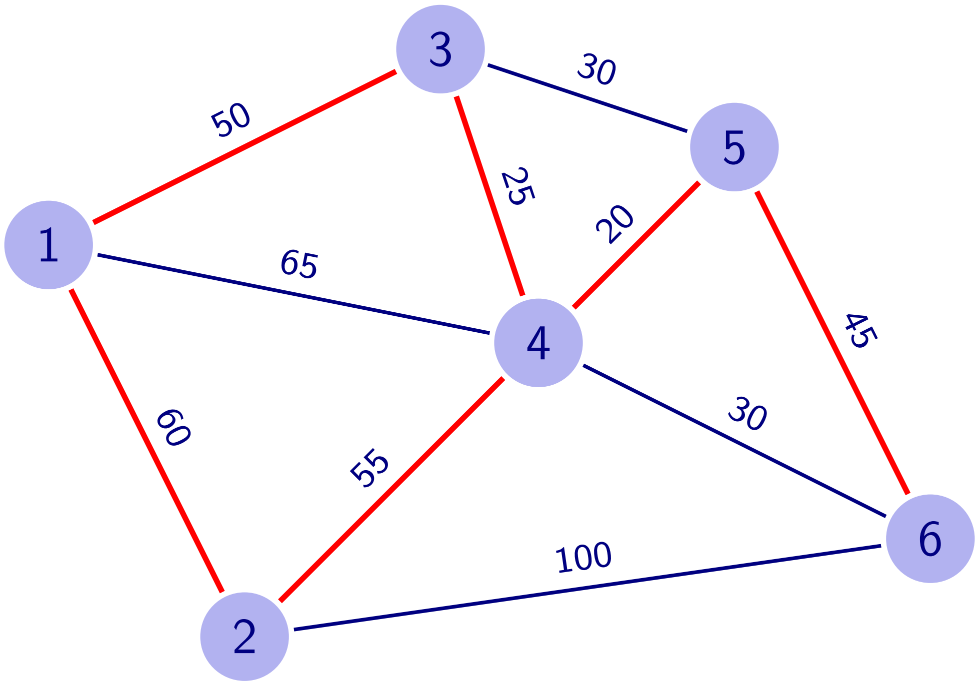


➡ total weight: 260

▷ Given a graph $G = (V, E)$ with non-negative edge-weights w_e for all $e \in E$...

...find a minimum spanning tree for G, that is: a subset E' of the edges such that

- the edges in E' form a tree (connected and no cycles)
- all vertices of G are in the tree
- the total weight of the tree edges is minimal

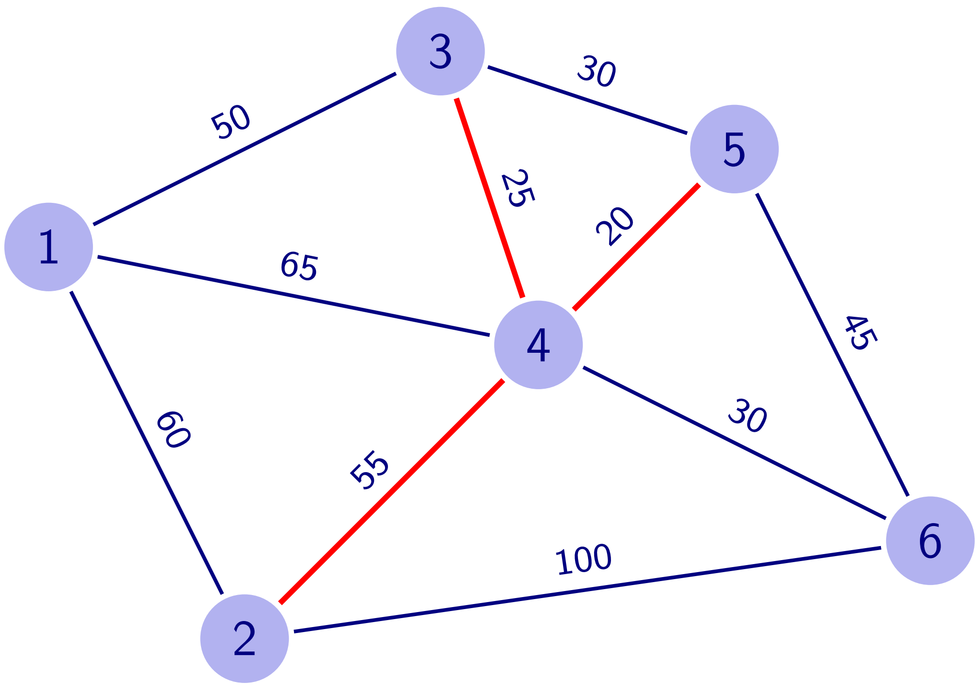


➡ not allowed: not a tree!

▷ Given a graph $G = (V, E)$ with non-negative edge-weights w_e for all $e \in E$...

...find a minimum spanning tree for G, that is: a subset E' of the edges such that

- the edges in E' form a tree (connected and no cycles)
- all vertices of G are in the tree
- the total weight of the tree edges is minimal

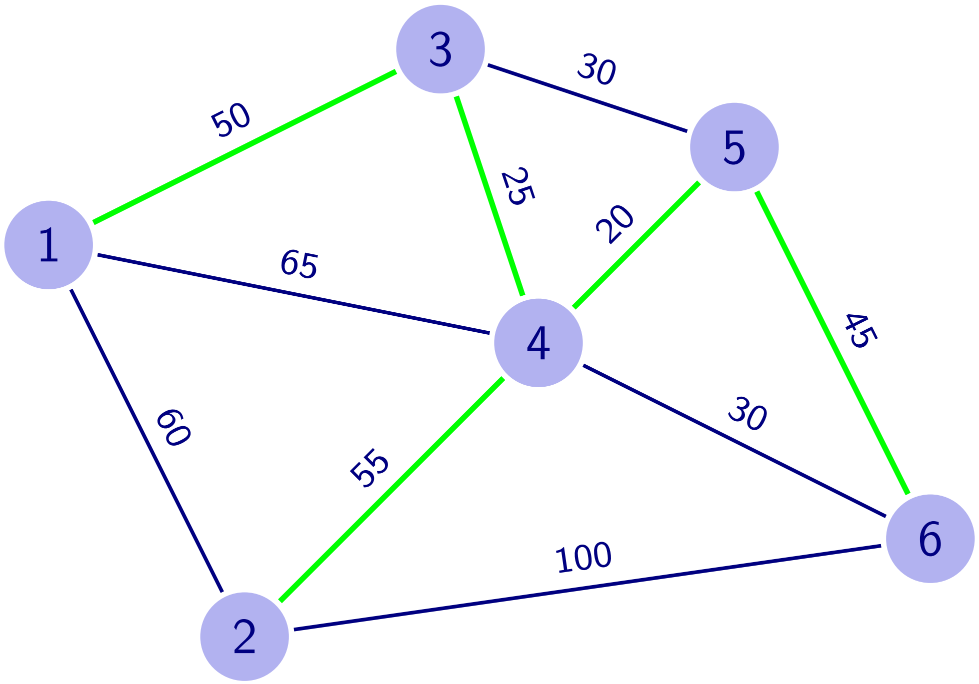


➡ not allowed: misses vertices!

▷ Given a graph $G = (V, E)$ with non-negative edge-weights w_e for all $e \in E$...

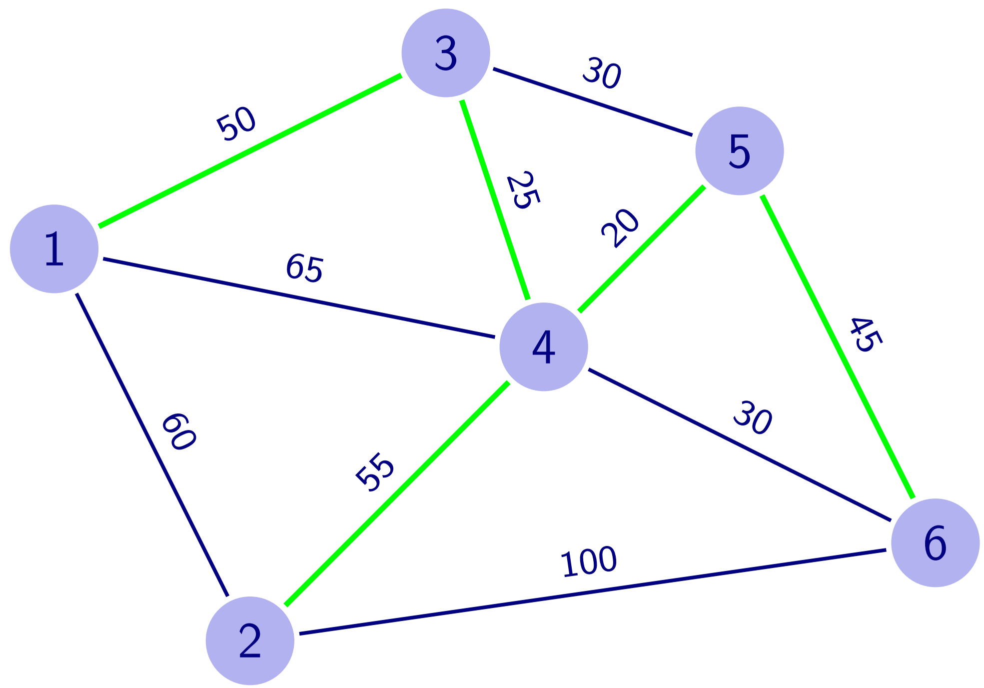
...find a minimum spanning tree for G, that is: a subset E' of the edges such that

- the edges in E' form a tree (connected and no cycles)
- all vertices of G are in the tree
- the total weight of the tree edges is minimal



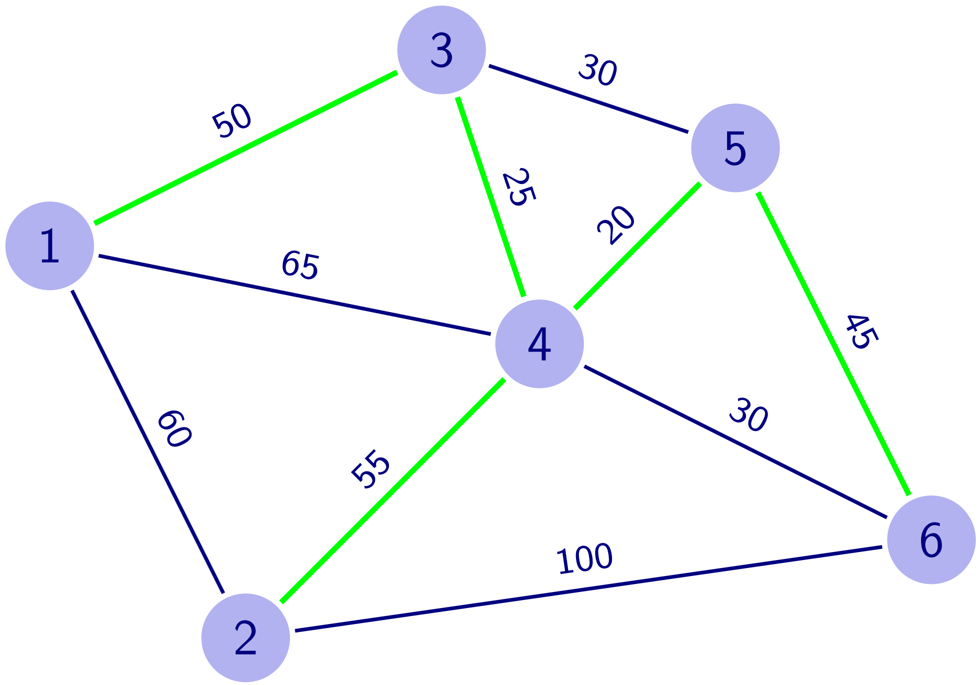
➡ total weight: 195

▷ Idea: at every step select the next cheap edge, as long as it doesn't result in a cycle



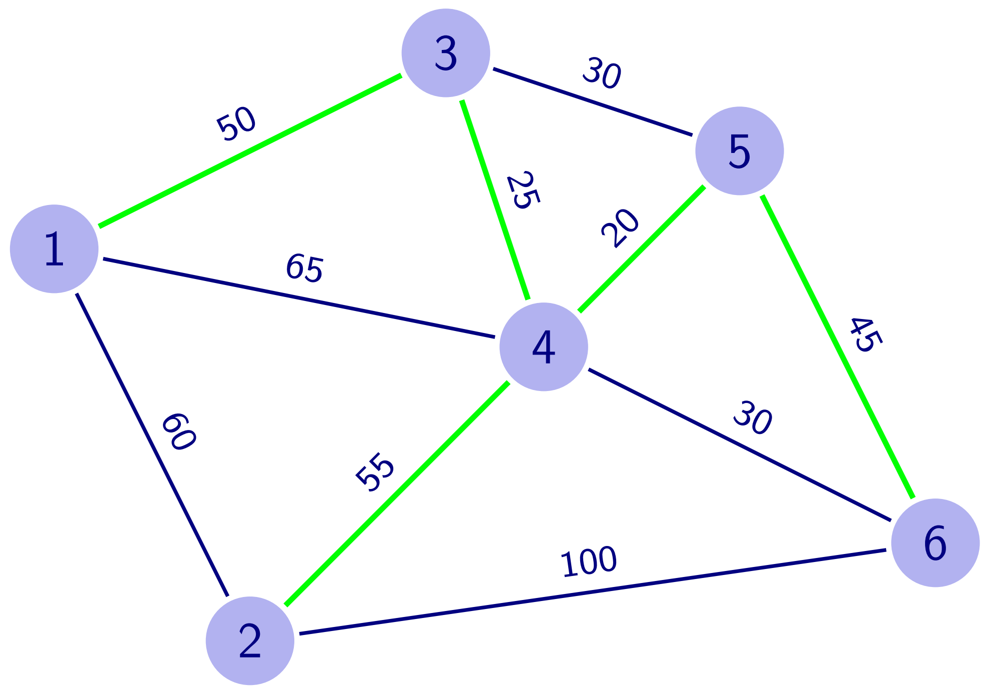
- Idea: at every step select the next cheap edge, as long as it doesn't result in a cycle

  ➡ Greedy algorithm

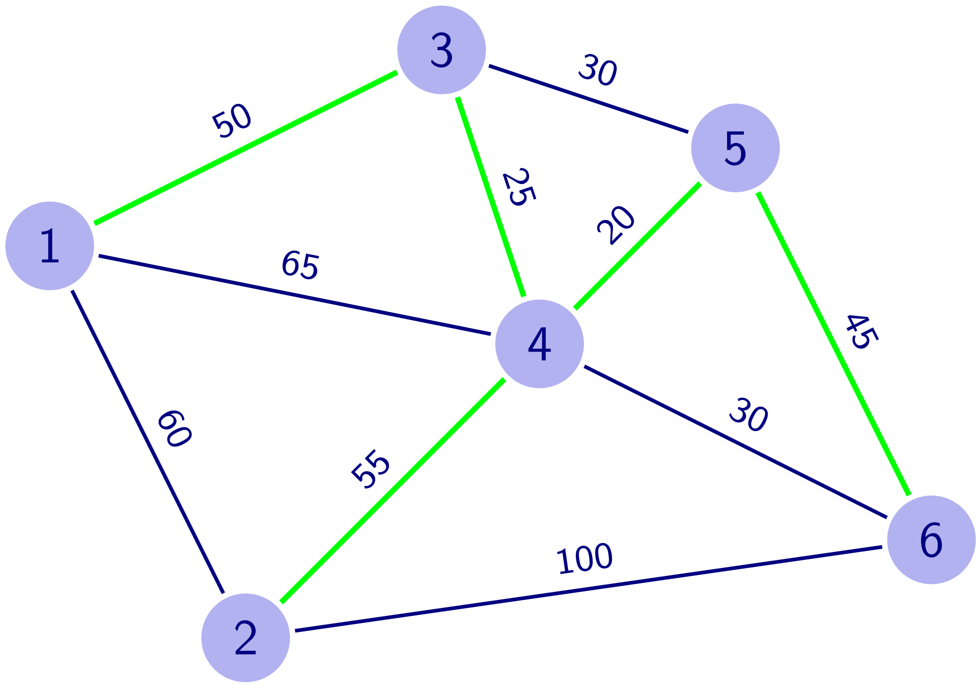


▷ Idea: at every step select the next cheap edge, as long as it doesn't result in a cycle

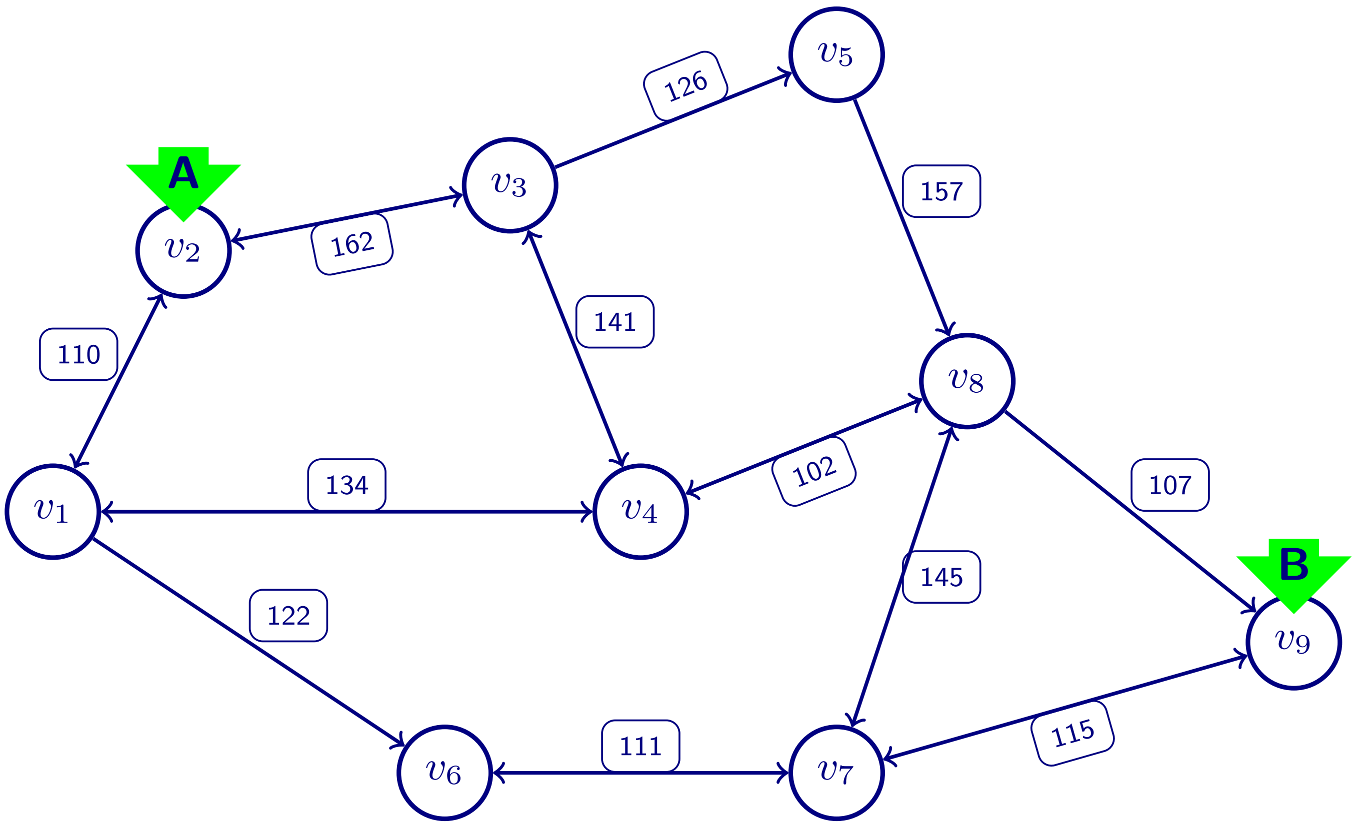
- ➡ Greedy algorithm
- ➡ Polynomial runtime
  - ➡ efficient algorithm



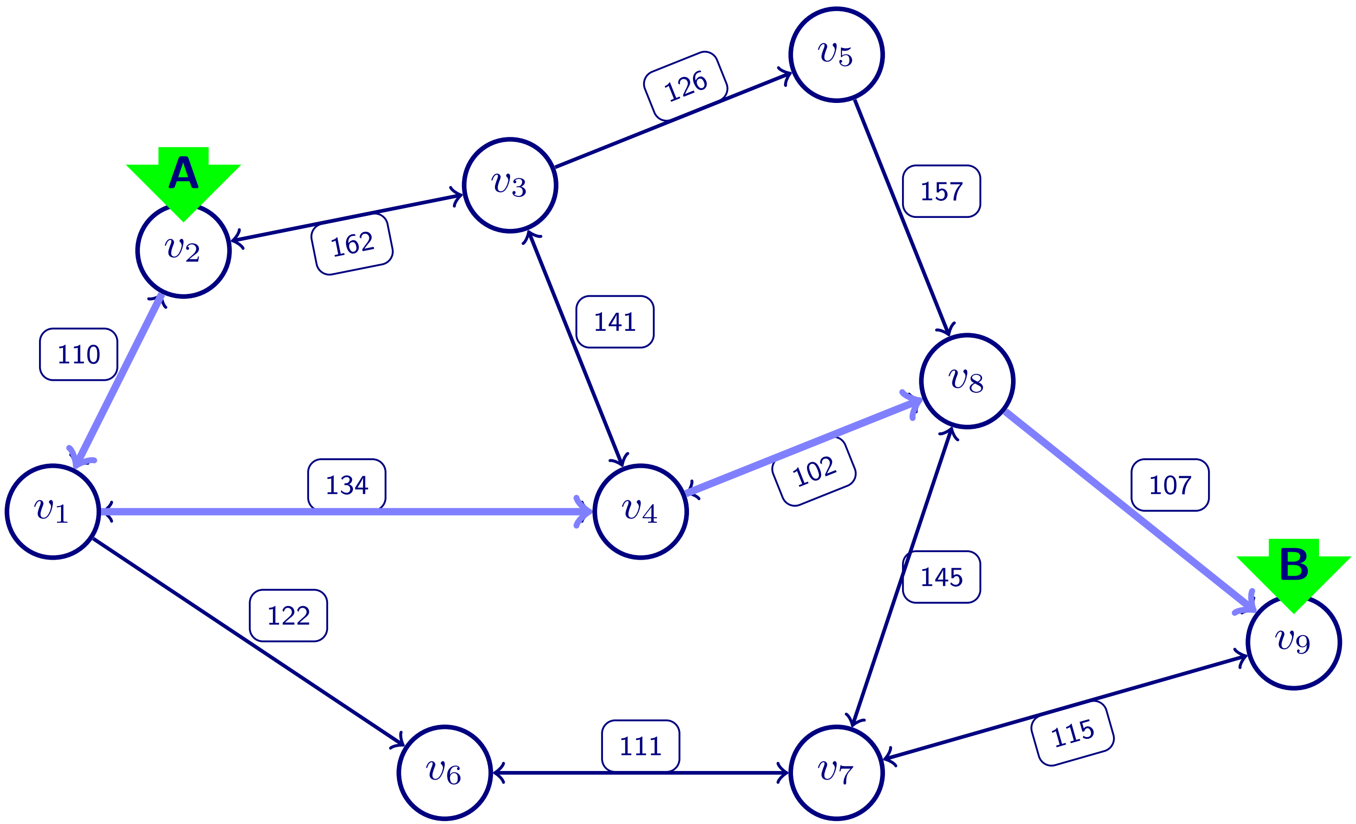
- Idea: at every step select the next cheap edge, as long as it doesn't result in a cycle
  - ➡ Greedy algorithm
  - ➡ Polynomial runtime
    - ➡ efficient algorithm
  - ➡ Yields an optimal solution for every input graph (proof!)
    - ➡ exact algorithm



▷ Given a network – i.e. a directed graph – with a length for each arc, a start node A and a destination B...



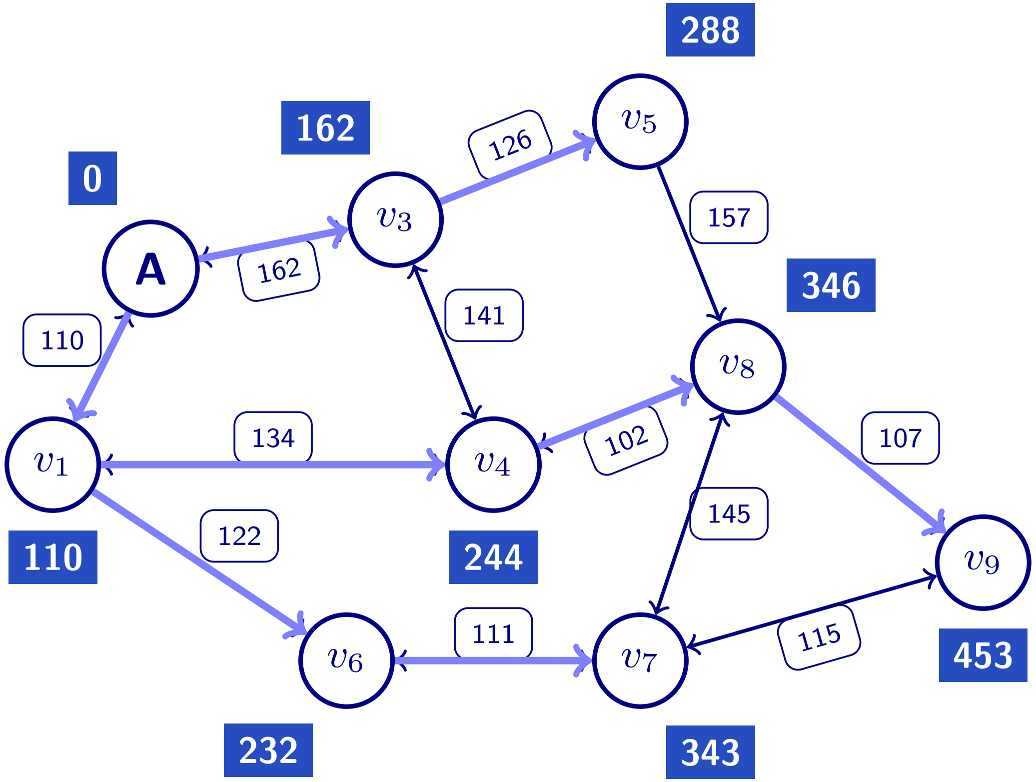
- Given a network – i.e. a directed graph – with a length for each arc, a start node A and a destination B...
- ...compute a shortest path through the network from A to B



- Computes a complete shortest path tree from start node A to all other nodes

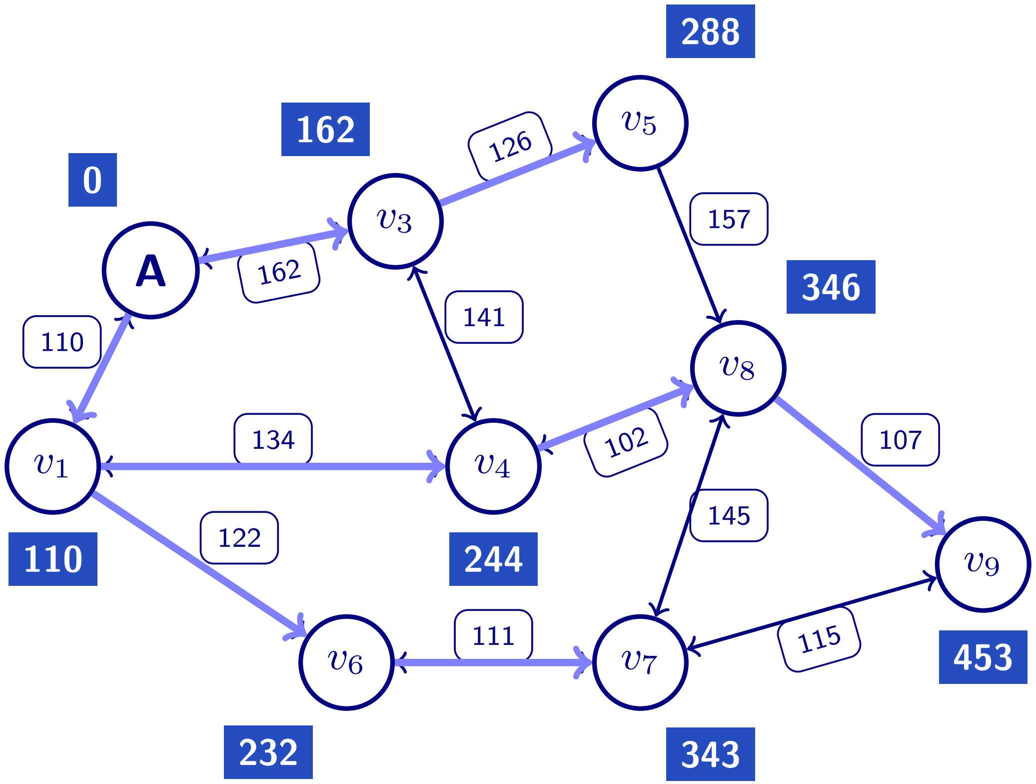
Edsger Wybe Dijkstra
(1930–2002)

▷ Computes a complete shortest path tree from start node A to all other nodes



Edsger Wybe Dijkstra
(1930–2002)

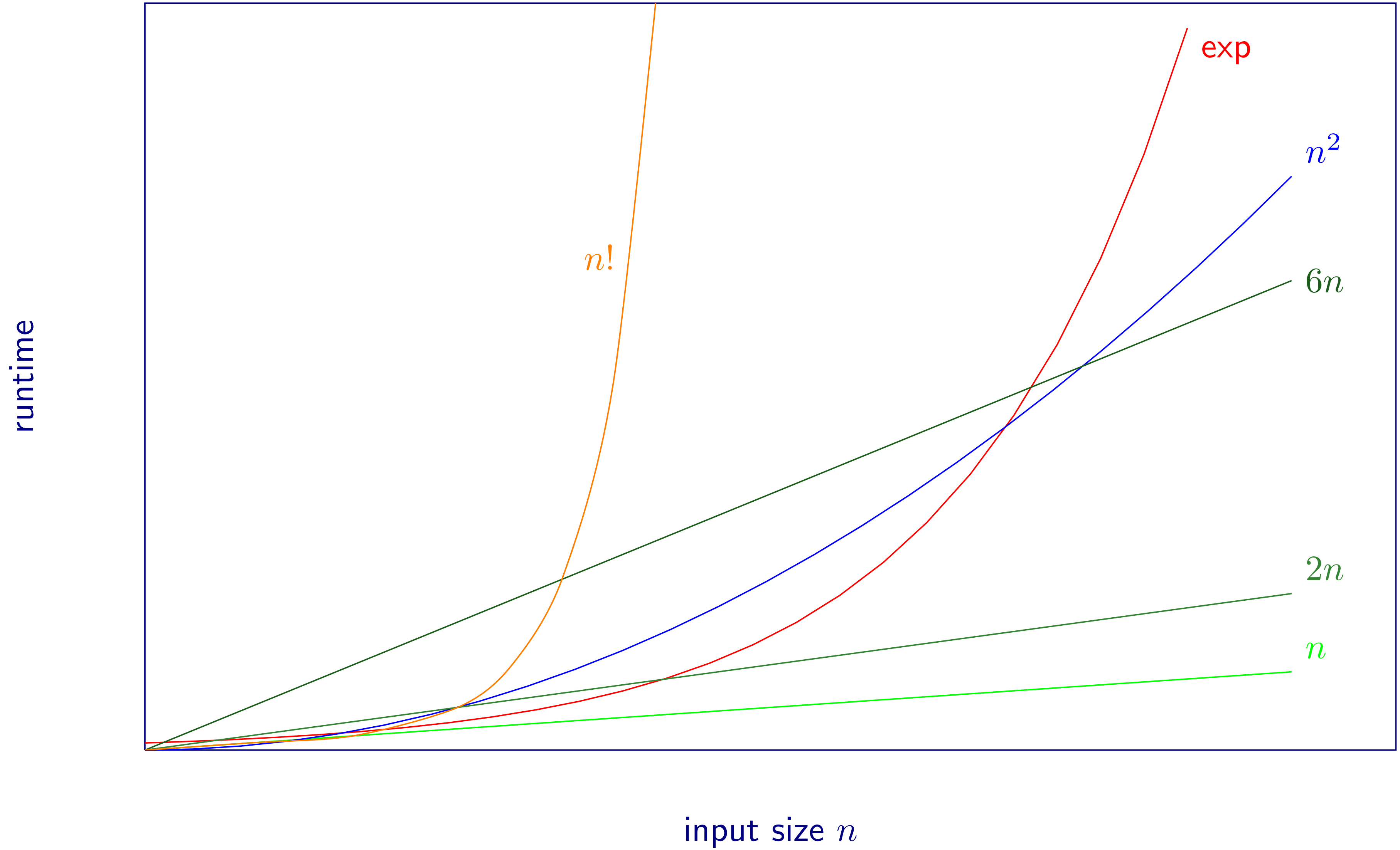
▷ Computes a complete shortest path tree from start node A to all other nodes



Edsger Wybe Dijkstra
(1930–2002)

▷ Polynomial runtime ➡ efficient algorithm

▷ Always yields an optimal solution ➡ exact algorithm



linear — polynomial — exponential

An algorithm is called efficient if it has polynomial runtime (i.e. its runtime can be bounded by a polynomial in the input size).

An algorithm is called efficient if it has polynomial runtime (i.e. its runtime can be bounded by a polynomial in the input size).

An algorithm is called exact if it guarantees to return an optimal solution (i.e. it can be proved that it always (i.e. for every input!) returns a solution with best possible objective).

An algorithm is called efficient if it has polynomial runtime (i.e. its runtime can be bounded by a polynomial in the input size).

An algorithm is called exact if it guarantees to return an optimal solution (i.e. it can be proved that it always (i.e. for every input!) returns a solution with best possible objective).

	efficient	not efficient
exact	Dijkstra's algorithm Kruskal's algorithm Ellipsoid method	Simplex algorithm (?) Branch & bound Complete enumeration
not exact	TSP heuristic using MST approximation algorithms	

- Models, Data and Instances
- Linear Optimization
  - Modelling as a linear program
  - Solving a linear program (graphically, and in princple by the simplex algorithm)
  - Sensitivity analysis
- (Mixed) Integer Programming
  - Modelling as a (mixed) integer program
  - How to solve a (mixed) integer program (in principle)
- Combinatorial Optimization
  - Exemplary problems, algorithms, and runtimes
- Nonlinear Optimization
  - Local and global optima, convex optimization
- Scheduling
- Lot Sizing
- Multicriteria Optimization

▷ Model (non-linear program):

Objective maximize $10x_{\mathsf{b}} + 150\sqrt{x_{\mathsf{p}}} - 20x_{\mathsf{p}}$

C subject to

$$\begin{aligned} 5x_{\mathsf{b}} + 3x_{\mathsf{p}} &\leq 50 && \text{(total land usage)} \\ 4x_{\mathsf{b}} + 7x_{\mathsf{p}} &\leq 70 && \text{(total working time)} \\ 4x_{\mathsf{b}} + 4x_{\mathsf{p}} &\leq 45 && \text{(total water consumption)} \end{aligned}$$

V

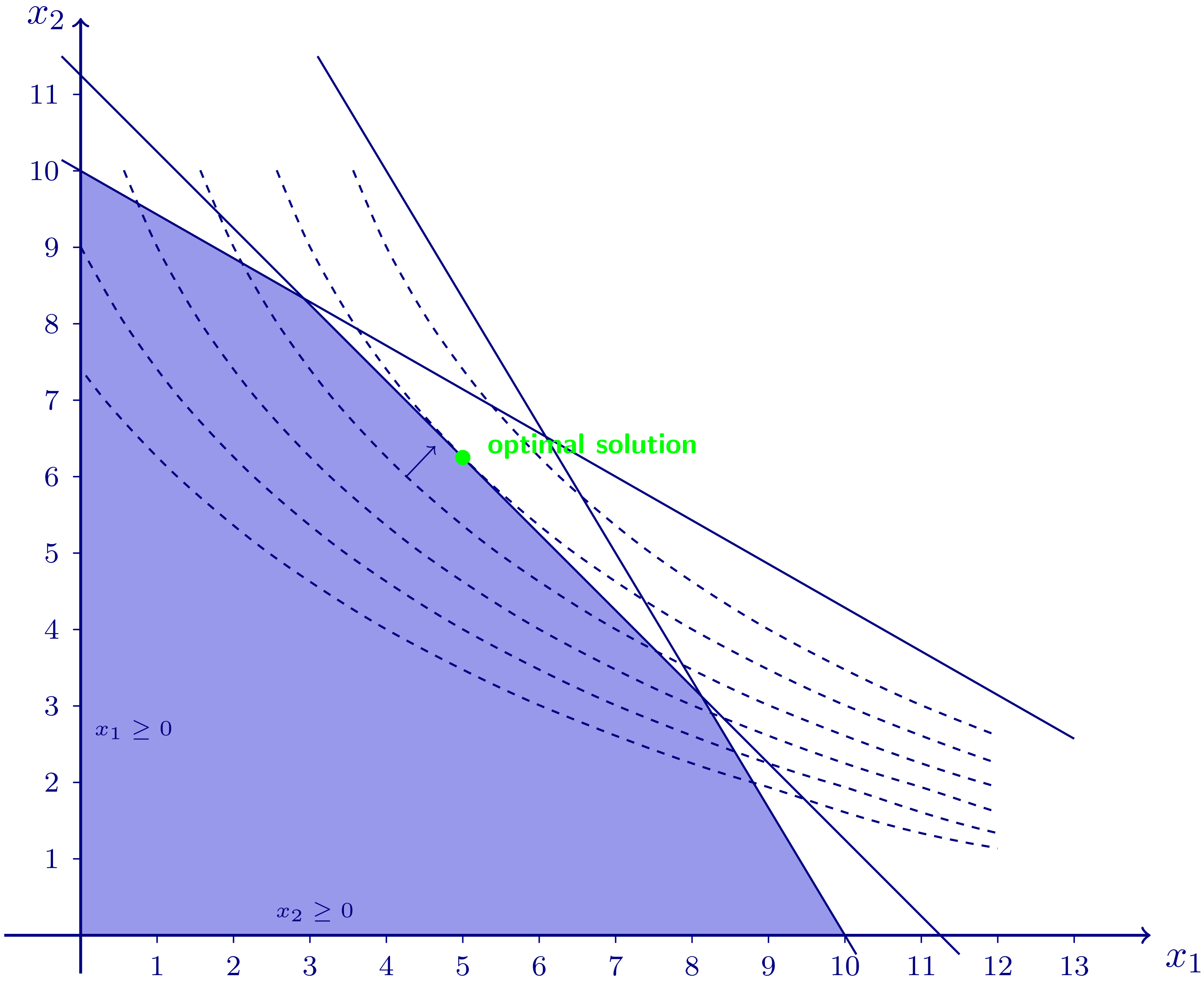
$$x_{\mathsf{b}}, x_{\mathsf{p}} \geq 0 \quad \text{(non-negativity)}$$

▷ Model (non-linear program):

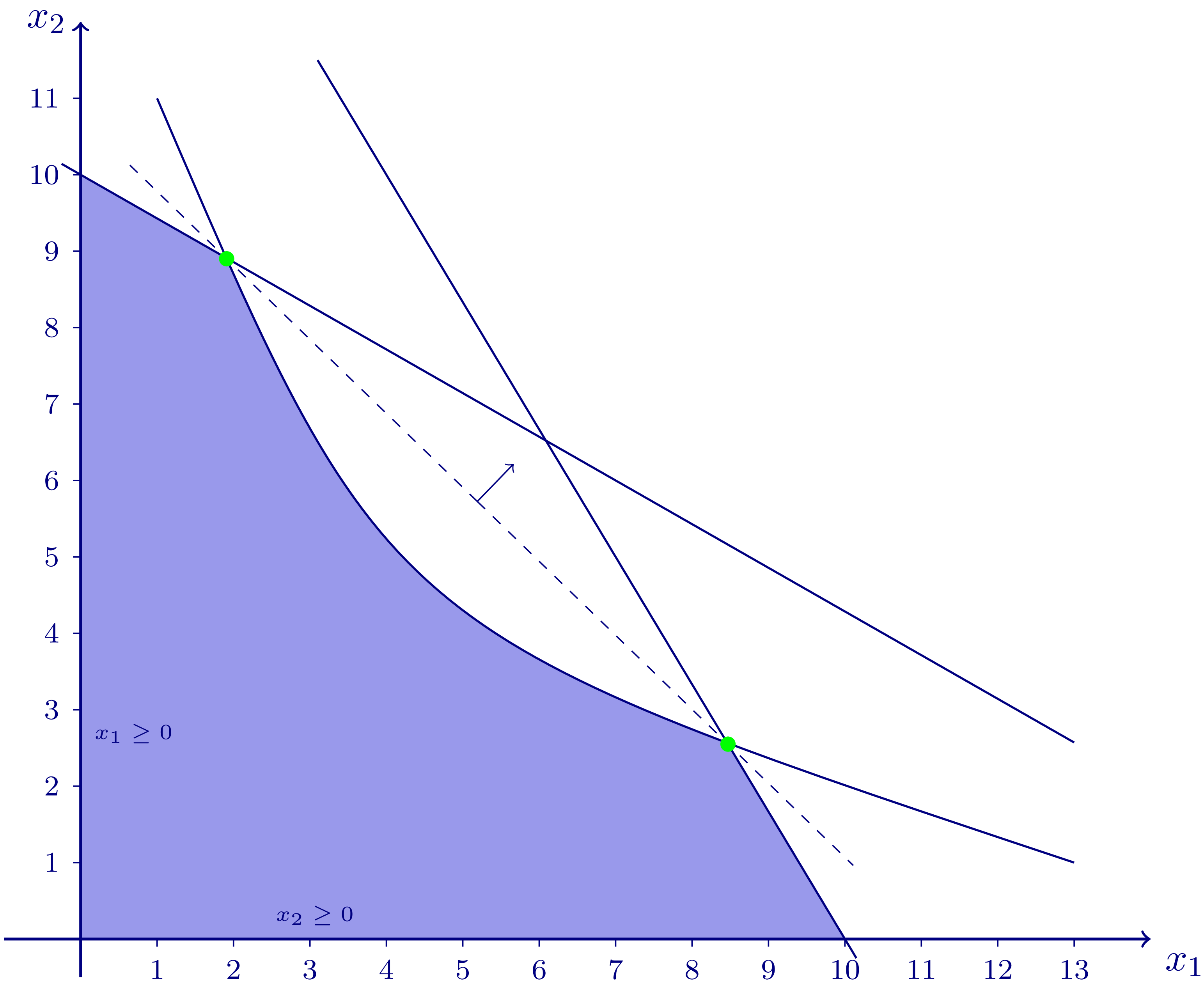
Objective	maximize	$10x_{\mathsf{b}} + 150\sqrt{x_{\mathsf{p}}} - 20x_{\mathsf{p}}$			
	subject to	$5x_{\mathsf{b}} + 3x_{\mathsf{p}}$	$\leq$	50	(total land usage)
C		$4x_{\mathsf{b}} + 7x_{\mathsf{p}}$	$\leq$	70	(total working time)
		$4x_{\mathsf{b}} + 4x_{\mathsf{p}}$	$\leq$	45	(total water consumption)
V		$x_{\mathsf{b}}, x_{\mathsf{p}}$	$\geq$	0	(non-negativity)

▷ Examples of non-linear terms:

- Products of variables: $x_i \cdot x_j$ } quadratic expressions
- Squares of variables: x_i^2 } quadratic expressions
- Higher-order terms of variables: $x_i \cdot x_j \cdot x_k$, $x_j^5 \cdot x_j$
- Absolute values or maxima/minima: $|x_i|$, $\max x_j$
- Terms including elementary functions: $\sin x_i$, $2^{x_i \cdot x_j}$, $\frac{1}{\sqrt{x_i}}$, $\log(x_i + x_j^{x_k})$



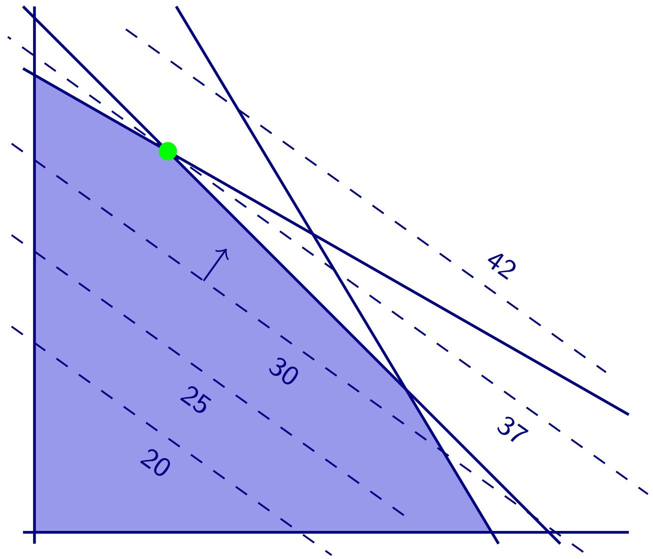
x_2
x_1
11
10
9
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
9
10
11
12
13
optimal solution
$x_1 \geq 0$
$x_2 \geq 0$



x_2
x_1
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13
$x_1 \geq 0$
$x_2 \geq 0$

▷ **Linear models**

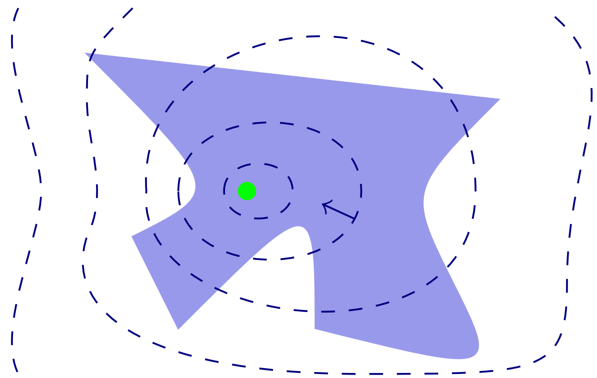
- Linear objective
  - ➡ Level sets are straight lines (in higher dimension: hyperplanes)
- Linear constraints
  - ➡ Feasible region is a polygon (in higher dimension: polyhedron)



➡ Optimal solutions can always be found in vertices

▷ **Non-linear models**

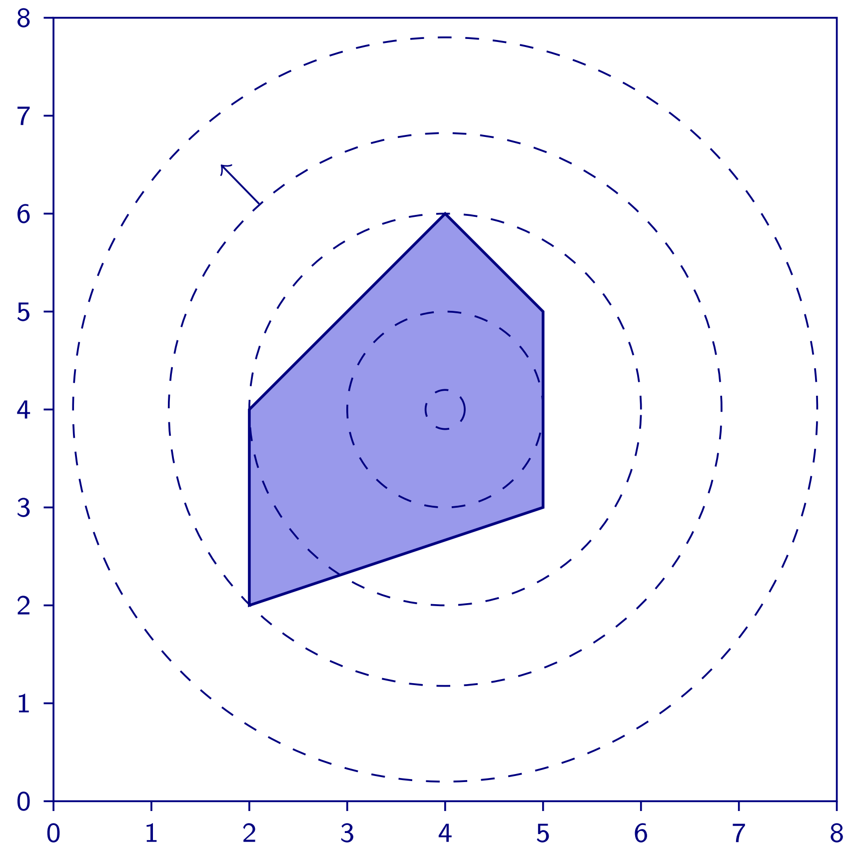
- Non-linear objective
  - ➡ Level sets can be complicated curves
- Non-linear constraints
  - ➡ Feasible region can be complicated



➡ Finding optimal solution can be difficult

- Example:

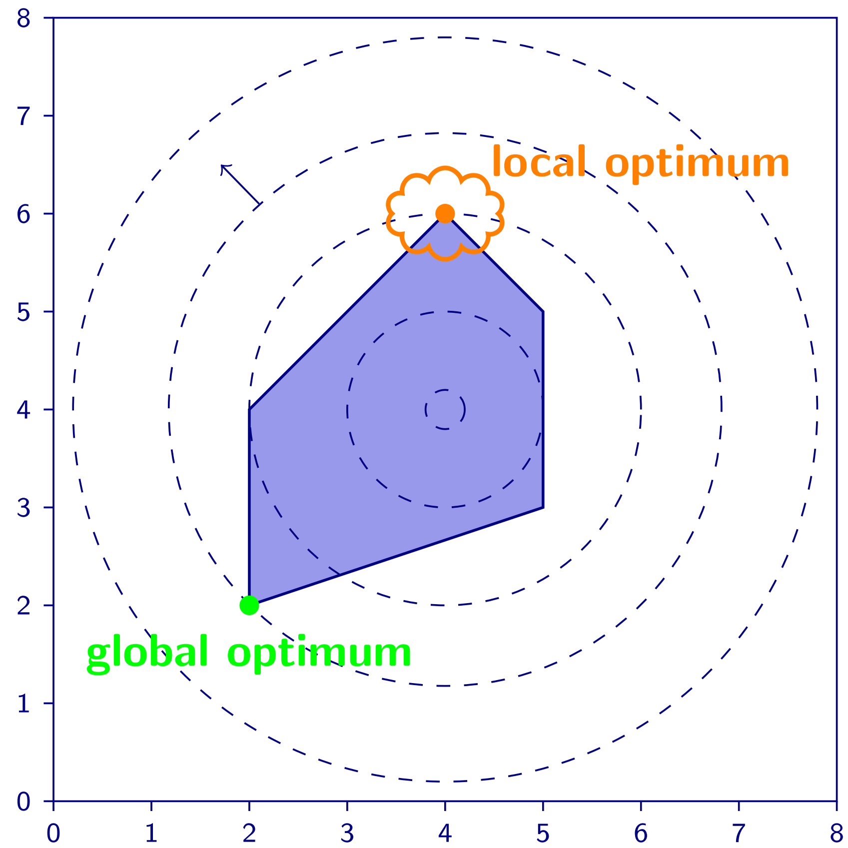
$$
\begin{aligned}
\max \quad & \sqrt{(x-4)^2+(y-4)^2} \\
\text{s.t.} \quad x &\geq 2 \\
x &\leq 5 \\
-x+y &\leq 2 \\
x+y &\leq 10 \\
x-3y &\leq -4
\end{aligned}
$$



Local and global optima

▷ Example:

$$\begin{aligned} \max \quad & \sqrt{(x-4)^2 + (y-4)^2} \\ \text{s.t.} \quad x &\geq 2 \\ x &\leq 5 \\ -x + y &\leq 2 \\ x + y &\leq 10 \\ x - 3y &\leq -4 \end{aligned}$$



➡ (4,6) is a local (but not a global) optimum

➡ (2,2) is a (local and) global optimum

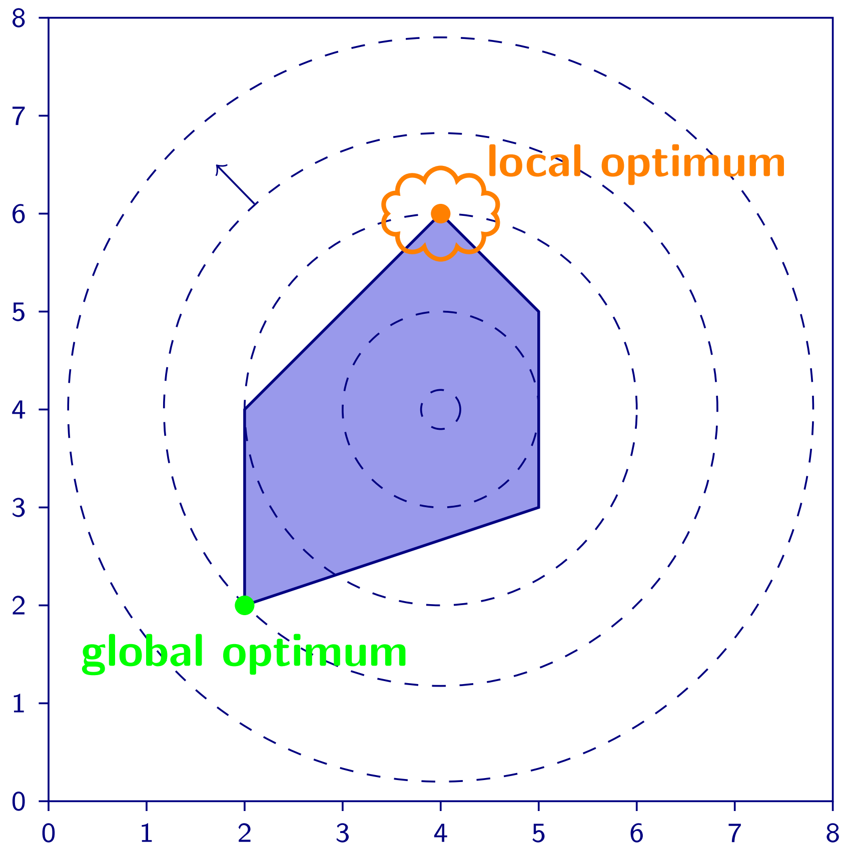
Local and global optima

- ▷ Example:

$$
\begin{aligned}
\max \quad & \sqrt{(x-4)^2+(y-4)^2} \\
\text{s.t.} \quad & x \geq 2 \\
& x \leq 5 \\
& -x+y \leq 2 \\
& x+y \leq 10 \\
& x-3y \leq -4
\end{aligned}
$$

➡ (4,6) is a local (but not a global) optimum

➡ (2,2) is a (local and) global optimum



A feasible solution is called locally optimal if there is no nearby feasible solution with a better objective function value

A feasible solution is called globally optimal if there is no feasible solution at all with a better objective function value

▷ Usual strategy of solvers for non-linear models:

- Find a point somewhere in the feasible region
- Follow steps to find a local optimum

- Usual strategy of solvers for non-linear models:
  - Find a point somewhere in the feasible region
  - Follow steps to find a local optimum
- Problem: usually, the solution is not a global optimum!

▷ Usual strategy of solvers for non-linear models:

- Find a point somewhere in the feasible region
- Follow steps to find a local optimum

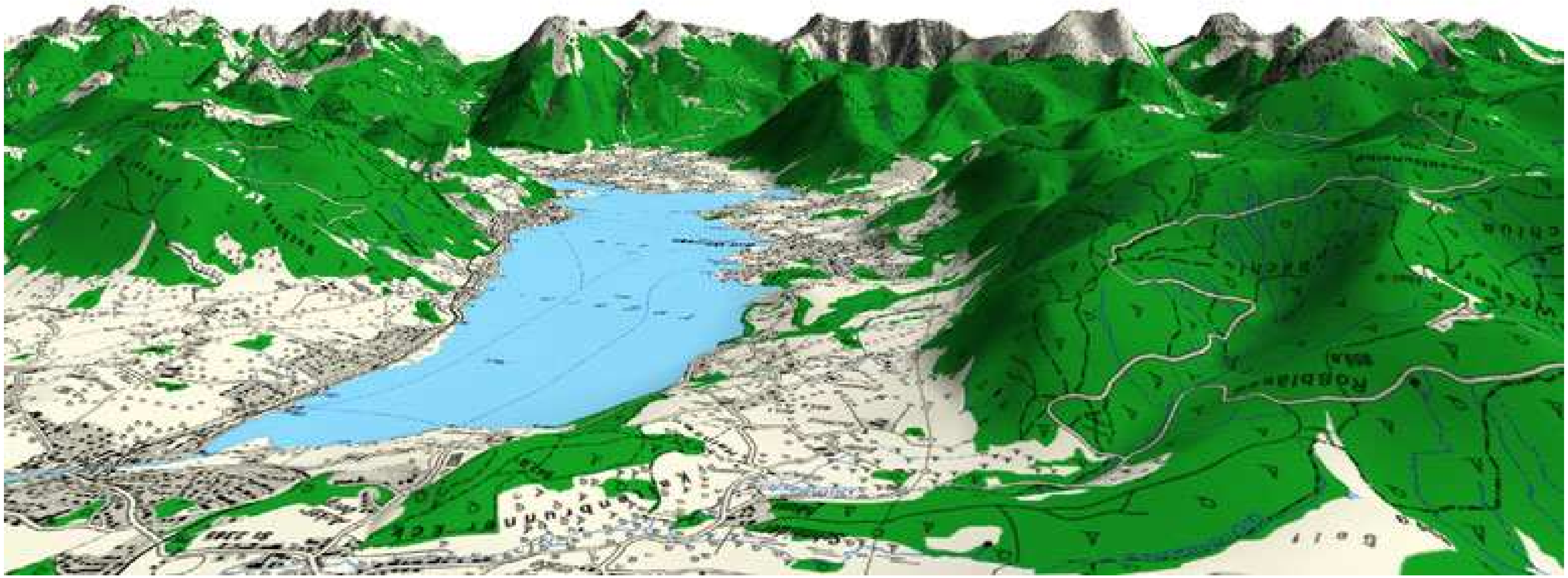
▷ Problem: usually, the solution is not a global optimum!

▷ In special cases, this works nonetheless:

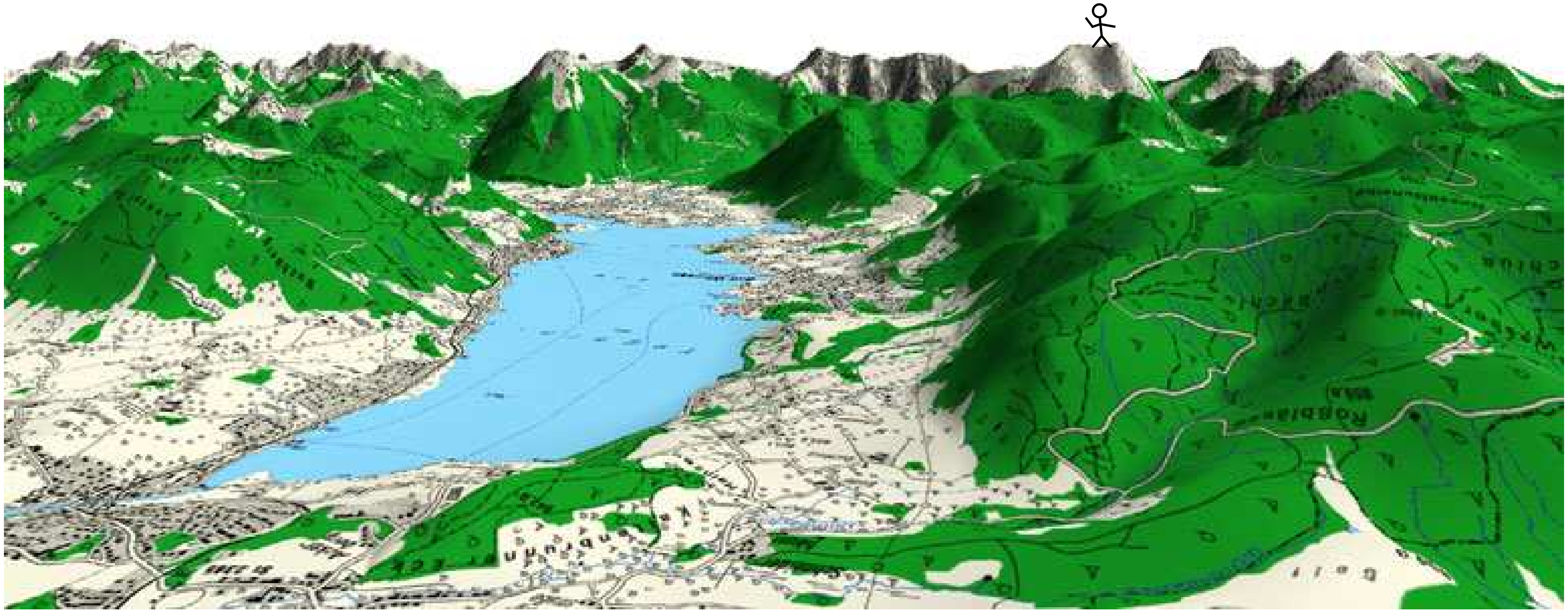
- If a concave function is maximized over a convex feasible set
- If a convex function is minimized over a convex feasible set
- If the problem is linear

- Usual strategy of solvers for non-linear models:
  - Find a point somewhere in the feasible region
  - Follow steps to find a local optimum
- Problem: usually, the solution is not a global optimum!
- In special cases, this works nonetheless:
  - If a concave function is maximized over a convex feasible set
  - If a convex function is minimized over a convex feasible set
  - If the problem is linear
- Possibilities otherwise:
  - Reformulate or approximate as a linear model
  - Rely on heuristic strategies and luck...

- Non-linear optimization is like mountain-climbing in the fog



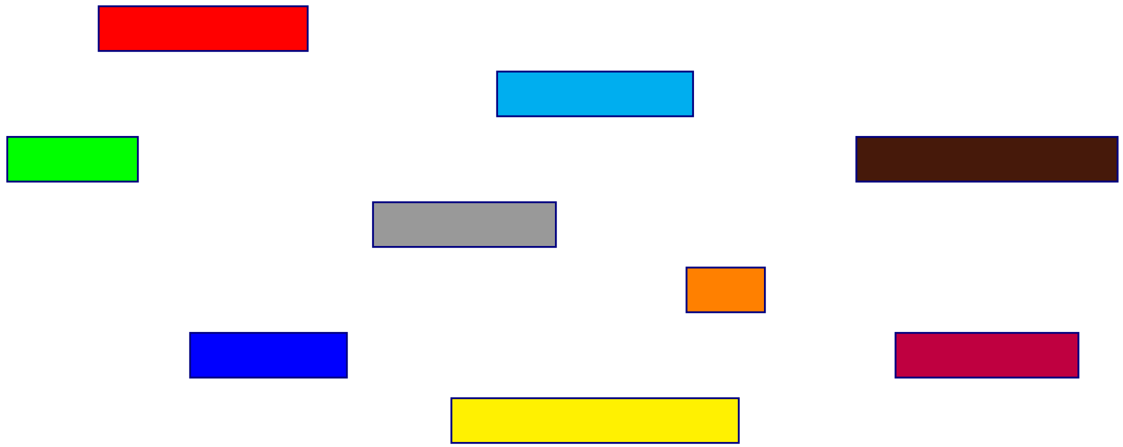
- ▷ Non-linear optimization is like mountain-climbing in the fog



- ▷ How do you know that you're on the highest mountain if you can't see the other peaks?

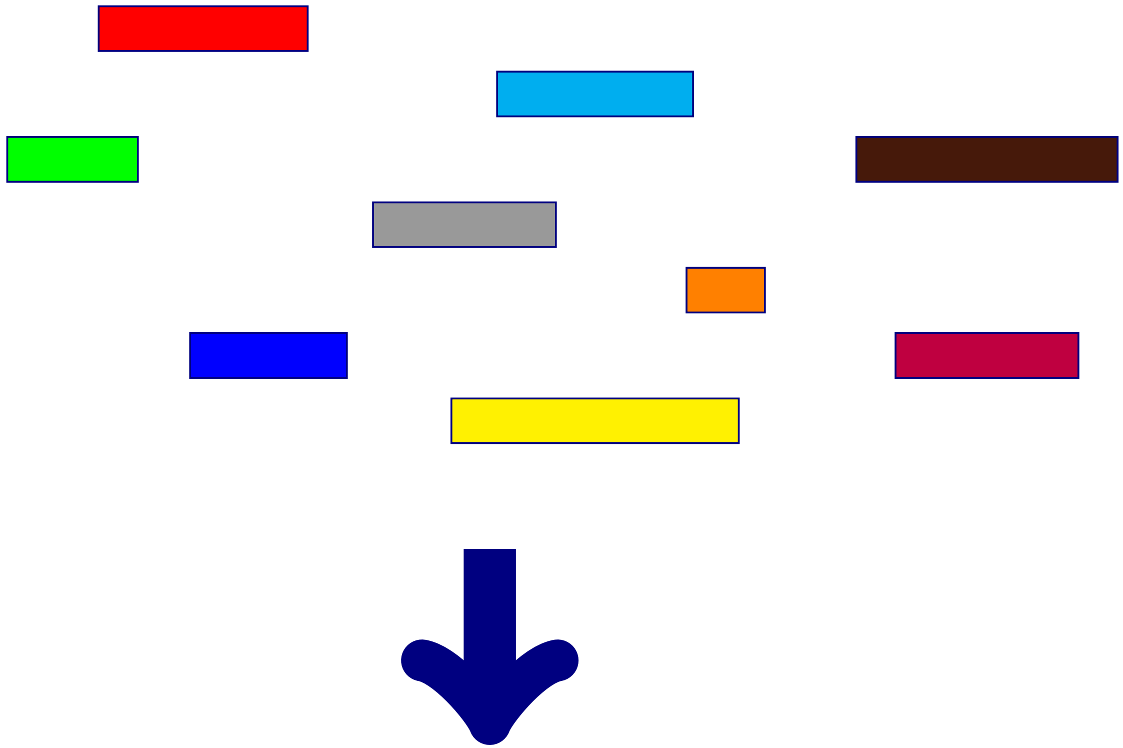
- ▷ Models, Data and Instances
- ▷ Linear Optimization
  - ➡ Modelling as a linear program
  - ➡ Solving a linear program (graphically, and in princple by the simplex algorithm)
  - ➡ Sensitivity analysis
- ▷ (Mixed) Integer Programming
  - ➡ Modelling as a (mixed) integer program
  - ➡ How to solve a (mixed) integer program (in principle)
- ▷ Combinatorial Optimization
  - ➡ Exemplary problems, algorithms, and runtimes
- ▷ Nonlinear Optimization
  - ➡ Local and global optima, convex optimization
- ▷ Scheduling
- ▷ Lot Sizing
- ▷ Multicriteria Optimization

- Jobs:

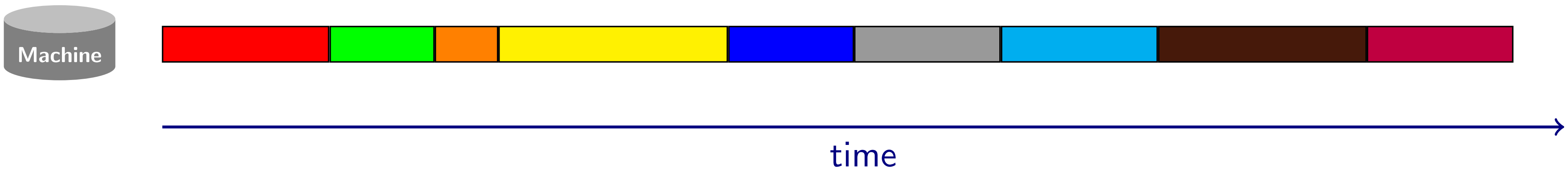


General principle

- Jobs:



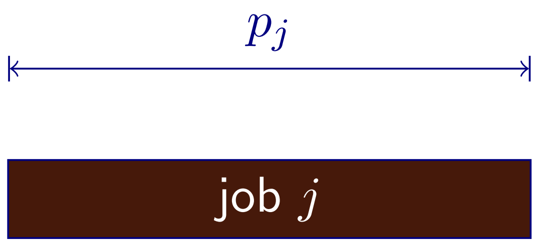
- Schedule (Gantt chart): ➡ optimal with respect to an objective to specify!



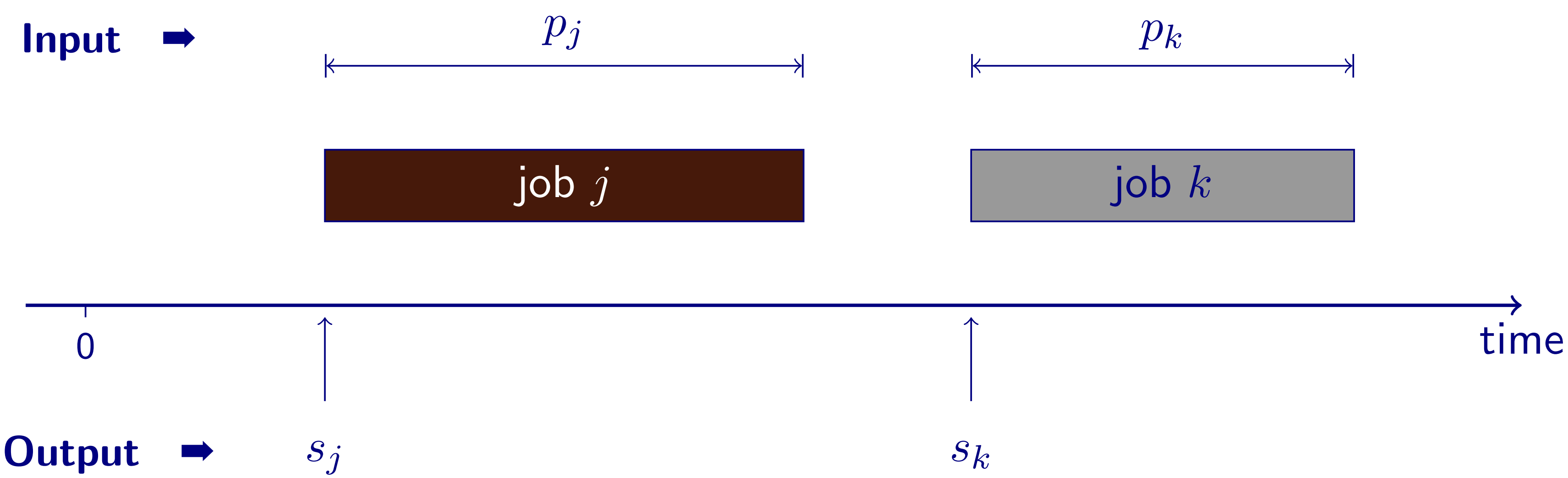
Basic terminology

- Jobs usually have: a processing time p_j

Input ➡

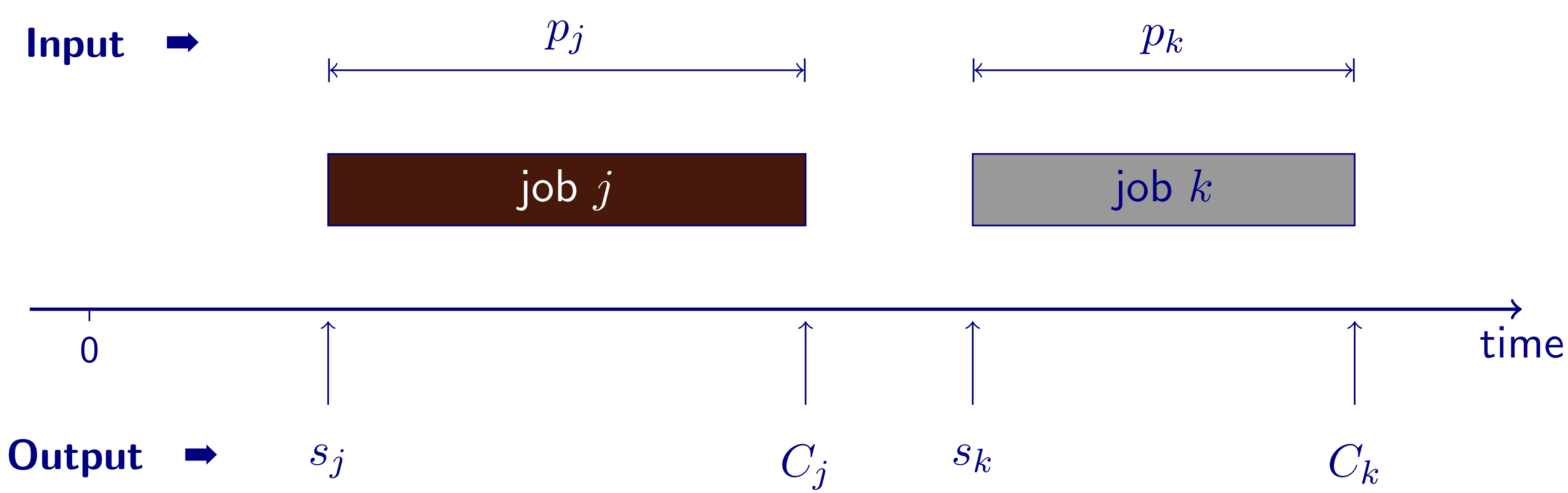


- Jobs usually have: a **processing time** p_j
- A schedule has to provide: a **start time** s_j, such that different jobs do not overlap



- ▷ Jobs usually have: a processing time p_j
- ▷ A schedule has to provide: a start time s_j, such that different jobs do not overlap

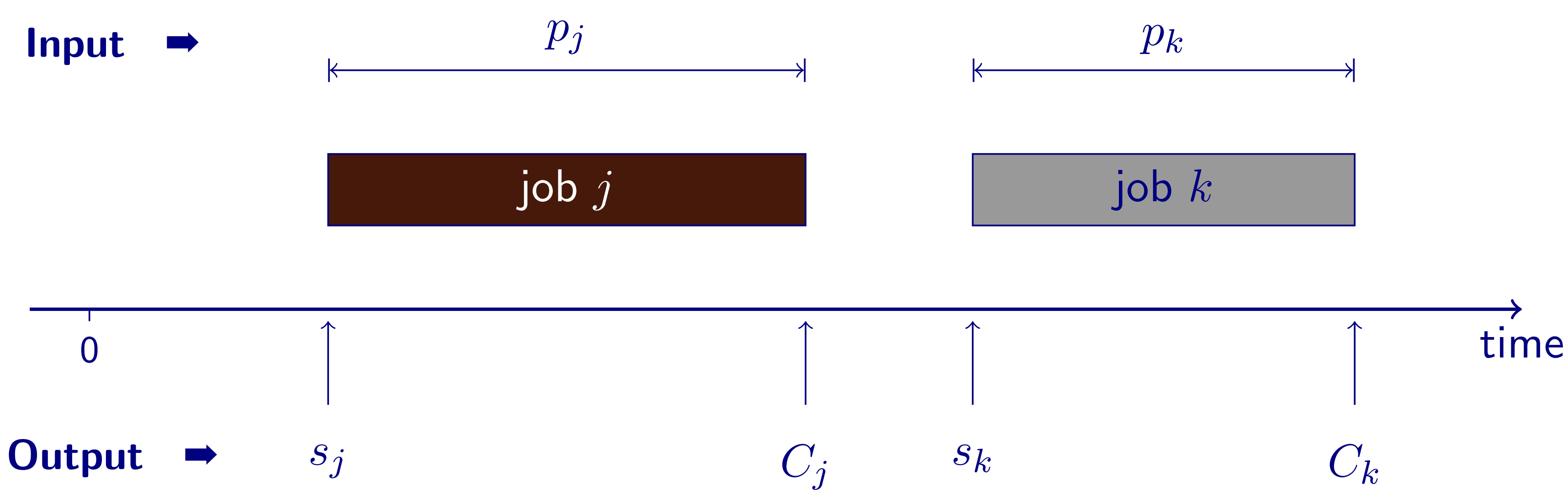
➡ Completion time $C_j := s_j + p_j$



- Jobs usually have: a **processing time** p_j
- A schedule has to provide: a **start time** s_j, such that different jobs do not overlap

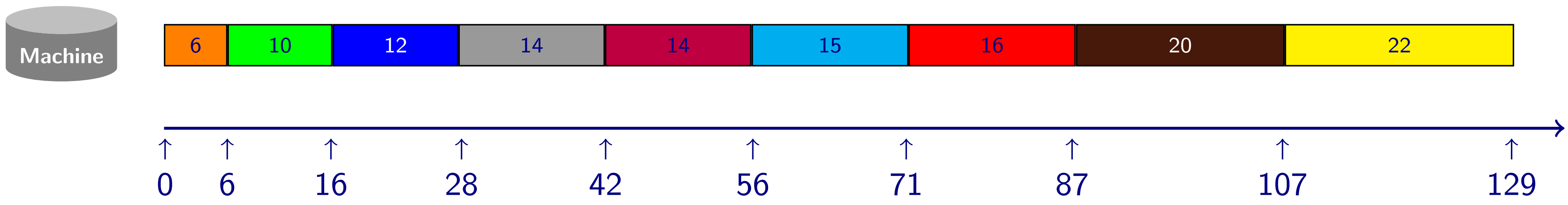
➡ **Completion time** $C_j := s_j + p_j$

- Possible objective functions: (to minimize)
  - ➡ **Sum of completion times** $\sum_{j=1}^{n} C_j$
  - ➡ **Makespan** $\max_{j=1,\ldots,n} C_j$



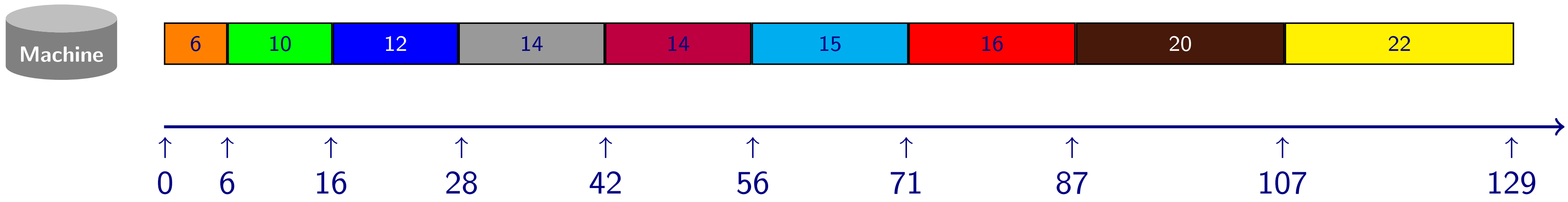
▷ Single Machine, minimize sum of completion times

➡ easy (greedy algorithm)



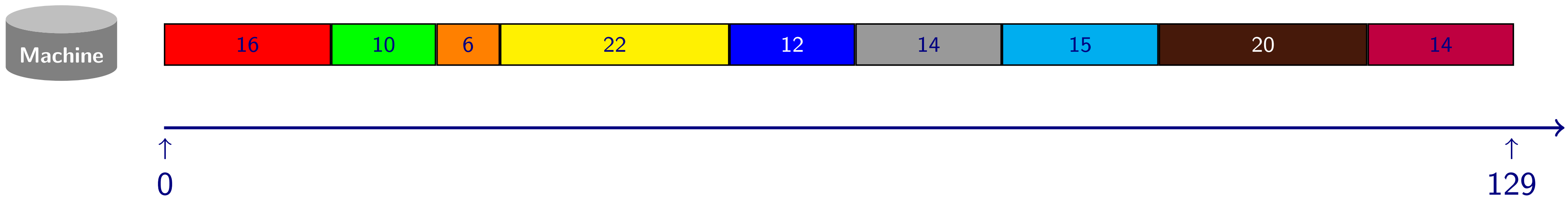
▷ Single Machine, minimize sum of completion times

➡ easy (greedy algorithm)



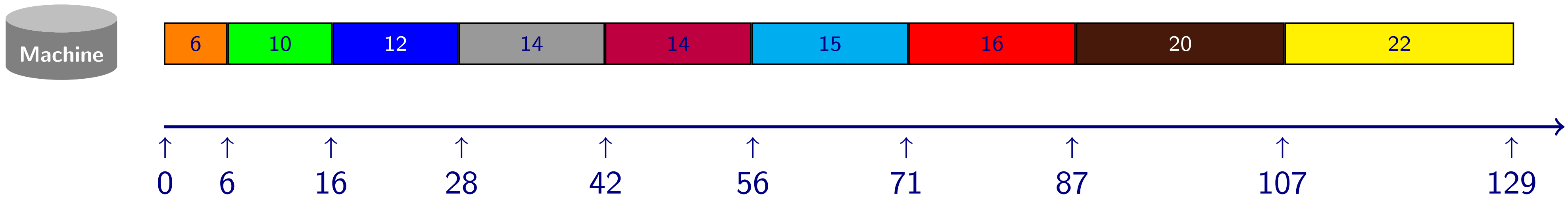
▷ Single Machine, minimize makespan

➡ trivial (always the same)

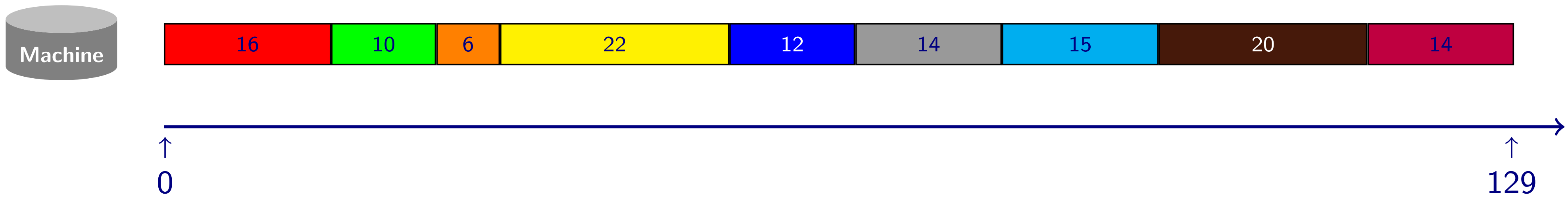


Scheduling – executive summary I

- ▷ Single Machine, minimize sum of completion times
  - ➡ easy (greedy algorithm)



- ▷ Single Machine, minimize makespan
  - ➡ trivial (always the same)



- ▷ Single Machine, jobs with release dates, minimize sum of completion times
  - ➡ similarly easy (greedy algorithm)

▷ Jobs with precedence constraints (project scheduling)

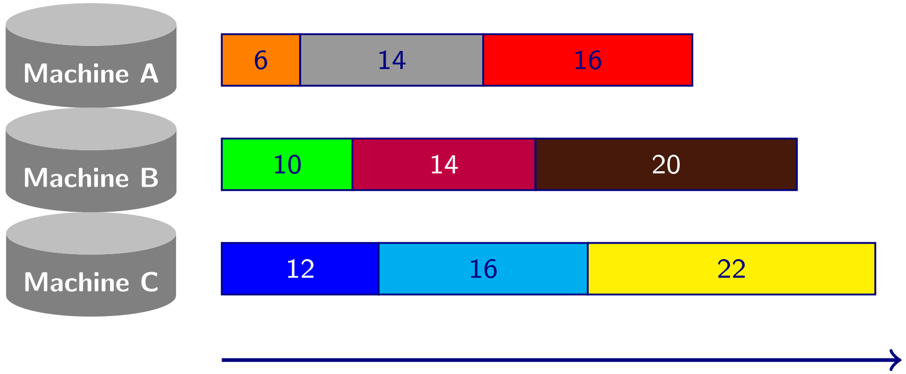
- Single machine ➡ easy (greedy)
- Multiple machines ➡ hard
- Unlimited number of machines ➡ easy again (critical path method)

▷ Jobs with precedence constraints (project scheduling)

- Single machine ➡ easy (greedy)
- Multiple machines ➡ hard
- Unlimited number of machines ➡ easy again (critical path method)

▷ Multiple machines, minimize sum of completion times

➡ easy (greedy)

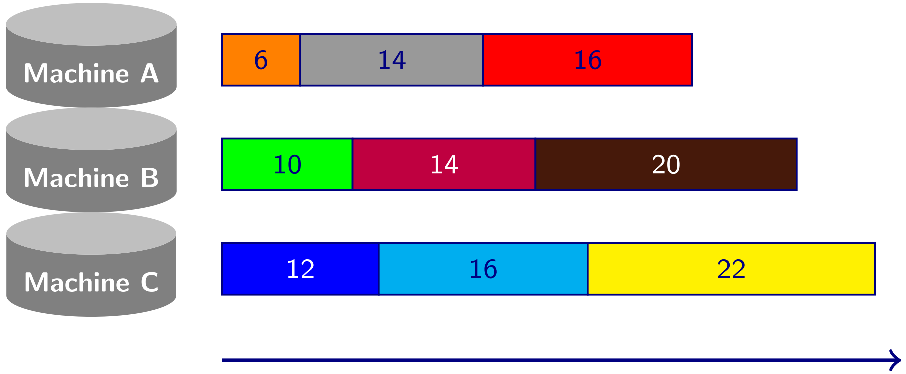


▷ Jobs with precedence constraints (project scheduling)

- Single machine ➡ easy (greedy)
- Multiple machines ➡ hard
- Unlimited number of machines ➡ easy again (critical path method)

▷ Multiple machines, minimize sum of completion times

➡ easy (greedy)



▷ Multiple machines, minimize makespan

➡ hard (partitioning problem)

- ▷ Models, Data and Instances
- ▷ Linear Optimization
  - ➡ Modelling as a linear program
  - ➡ Solving a linear program (graphically, and in princple by the simplex algorithm)
  - ➡ Sensitivity analysis
- ▷ (Mixed) Integer Programming
  - ➡ Modelling as a (mixed) integer program
  - ➡ How to solve a (mixed) integer program (in principle)
- ▷ Combinatorial Optimization
  - ➡ Exemplary problems, algorithms, and runtimes
- ▷ Nonlinear Optimization
  - ➡ Local and global optima, convex optimization
- ▷ Scheduling
- ▷ Lot Sizing
- ▷ Multicriteria Optimization

Data:

- d_t Demand in period t
- f_t fixed (start-up) costs in period t
- c_t unit production costs in period t
- h_t unit holding costs in period t
- C_t available capacity in period t

Variables:

- x_t production in period t
- y_t installation of capacity in period t
- s_t inventory at the end of period t

$$\min \sum_t c_t x_t + f_t y_t + h_t s_t$$

$$\begin{aligned}
s_{t-1} + x_t &= d_t + s_t && t = 1, \ldots, n \\
x_t &\leq C_t y_t && t = 1, \ldots, n \\
x_t, s_t &\geq 0 && t = 1, \ldots, n \\
s_0 &= 0 && \\
y_t &\in \{0, 1\} && t = 1, \ldots, n
\end{aligned}$$

- NP-hard problem
- poly-solvable cases:
  - ▷ Wagner-Whitin: $C_t = \infty$ for all periods t. In practice, $C_t = M$ with M very large value.
  - ▷ constant capacity: $C_t = C$ for all periods t.
  - ▷ Discrete lot sizing: constant capacity $C_t = C$ and $x_t = C_t y_t$ for all periods t
  - ▷ capacity in each period an integer multiple of constant batch size: $C_t = C y_t$ with $y_t \in \mathbb{Z}_+$ for all periods t.

- ▷ Models, Data and Instances
- ▷ Linear Optimization
  - ➡ Modelling as a linear program
  - ➡ Solving a linear program (graphically, and in princple by the simplex algorithm)
  - ➡ Sensitivity analysis
- ▷ (Mixed) Integer Programming
  - ➡ Modelling as a (mixed) integer program
  - ➡ How to solve a (mixed) integer program (in principle)
- ▷ Combinatorial Optimization
  - ➡ Exemplary problems, algorithms, and runtimes
- ▷ Nonlinear Optimization
  - ➡ Local and global optima, convex optimization
- ▷ Scheduling
- ▷ Lot Sizing
- ▷ Multicriteria Optimization

Multicriteria MIP model

$$\max \sum_{j=1}^{n} c_j^1 x_j, \sum_{j=1}^{n} c_j^2 x_j, \ldots, \sum_{j=1}^{n} c_j^q x_j$$

$$\begin{aligned} \text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i & \qquad i = 1, \ldots, m \\ \ell_j \le x_j \le u_j & \qquad j = 1, \ldots, n \\ x_j \in \mathbb{Z}_+ & \qquad j = 1, \ldots, k \end{aligned}$$

Multicriteria MIP model

$$
\begin{aligned}
\max \sum_{j=1}^{n} c_j^1 x_j, \sum_{j=1}^{n} c_j^2 x_j, \ldots, \sum_{j=1}^{n} c_j^q x_j & \\
\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \le b_i & \qquad i = 1, \ldots, m \\
\ell_j \le x_j \le u_j & \qquad j = 1, \ldots, n \\
x_j \in \mathbb{Z}_+ & \qquad j = 1, \ldots, k
\end{aligned}
$$

Ideas:

- find efficient (non-dominated) solutions (A solution is efficient or non-dominated if no objective value can be improved without reducing the other objective values)
- combine objective functions to weighted linear combination
- maximize one objective subject to bounds on all other objectives
- goal programming: solver get's numerical requirements $\tilde{c}_j$ that have to be achieved as much as possible

Oral Exam takes place on Wed, 15 Feb, 10:15 a.m. - 1:45 p.m.
in PTZ 307

GOOD LUCK!