Lecture 14

Lot Sizing & Multicriteria Optimization

01 Feb 2012

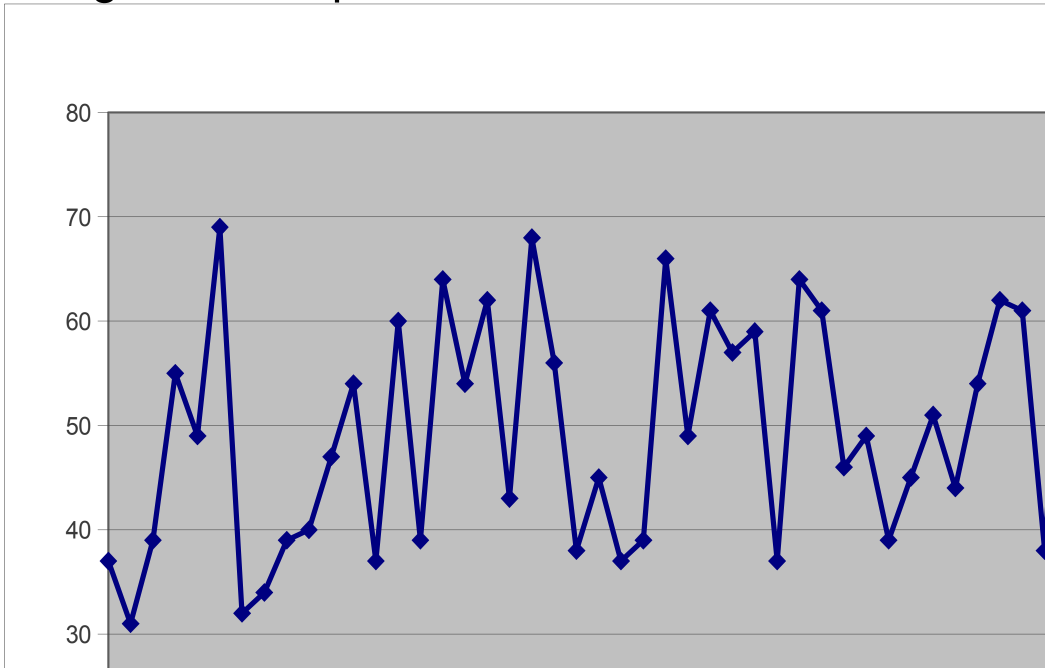
TU Berlin
Rüdiger Stephan

- **Lot-Sizing Problems**

IDEA manufactures and sells furniture

KJELT

Selling numbers per week



- Production of couches is performed in batches
- Inventory to meet demand

How large should be the batch size; When should be produced?

Matching of Batch Size and Inventory such that cost are minimized

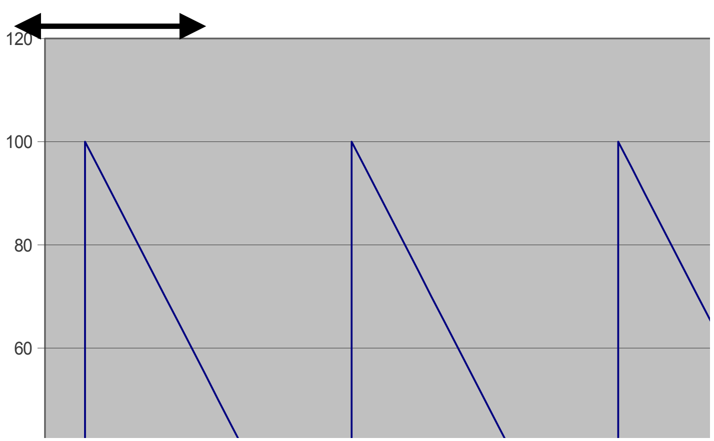
- Demand fluctuates per time unit
- Backorders are sometimes allowed
- Production involves a fixed setup cost (order cost)
- Lead time for delivery is instantaneous

Basic EOQ Model

- Demand is known with certainty and fixed at D per time unit
- Shortages are not permitted
- Lead time for delivery is instantaneous
- **Cost per order**: K
- **Unit holding cost** per time unit: h
  - Physical storage cost
  - Cost of Capital invested in inventory

Objective: Minimize average costs per time unit over an infinite time horizon

Cycle time T



Order whenever inventory hits zero

Order size: Q

T = Q / D or **Q = T · D**

Holding cost per cycle = ½hTQ

- Holding cost per cycle = ½hTQ

$$T = Q / D \quad \text{or} \quad Q = T \cdot D$$

- Cost per order = K
- Total cost per cycle = K + ½hTQ
- Average cost per time unit C(Q) = (K + ½hTQ) / T = KD / Q + ½hQ
- Find Q that minimizes average cost per time unit !

$$\frac{dC}{dQ} = 0 \Leftrightarrow \frac{-KD}{Q^2} + \frac{h}{2} = 0$$

$$Q = \sqrt{\frac{2KD}{h}}$$

$$C(Q) = \sqrt{2KDh}$$

- Holding cost per cycle = ½hTQ
- Cost per order of size Q = K + pQ

p Unit production cost
T = Q / D or Q = T · D

- Total cost per cycle = K + pQ + ½hTQ
- Average cost per time unit C(Q) = (K + pQ + ½hTQ) / T
  = KD / Q + pD + ½hQ
- Find Q that minimizes average cost per time unit !

$$\frac{dC}{dQ} = 0 \Leftrightarrow \frac{-KD}{Q^2} + \frac{h}{2} = 0$$

$$Q = \sqrt{\frac{2KD}{h}}$$

$$C(Q) = \sqrt{2KDh} + pD$$

Demand is not stationary but fluctuates over time: What to do?

Finite time horizon models

Wagner-Within Model

- Shortages are not permitted
- Starting inventory is zero
- Linear Holding cost h
- Fixed order cost K

D_t	Demand in period t
y_t	Production in period t
x_t	Inventory at the end of period t
$\delta(y)$	1 if y > 0, 0 if y = 0

$$x_t = \sum_{j=1}^{t} \left(y_j - D_j \right)$$

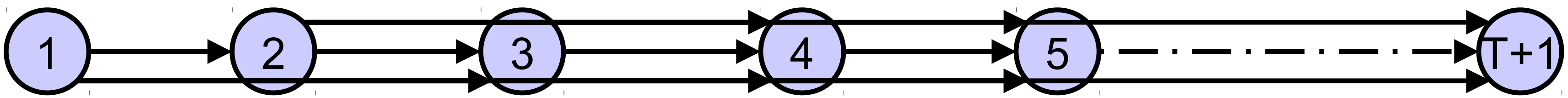
$$\min \quad \sum_{t=1}^{T} K_t \delta\left(y_t\right) + h_t x_t$$

$$s.t. \quad x_t = \sum_{j=1}^{t} \left(y_j - D_j \right) \quad \forall t = 1, \dots, T$$

$$x_t \geq 0, x_0 = 0$$

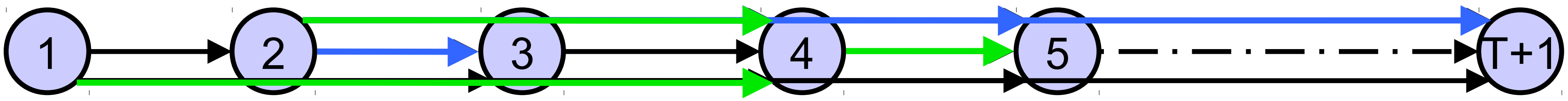
Observation: $y_t x_{t-1} = 0$ in optimal solution

- **Observation: $y_t x_{t-1} = 0$ in optimal solution**
- **$y_t > 0$ if $x_{t-1} = 0$, e.g., inventory zero**
- **$y_t \in \{0, D_t, D_t + D_{t+1}, D_t + D_{t+1} + D_{t+2}, \ldots\}$**
- **Optimal Solution can be found by shortest path computation**



- Node for every point in time 1,…,T+1
- Arc (i,j) for all i < j
- Length of arc (i,j) equals cost of ordering in period i to satisfy demand through period j-1:

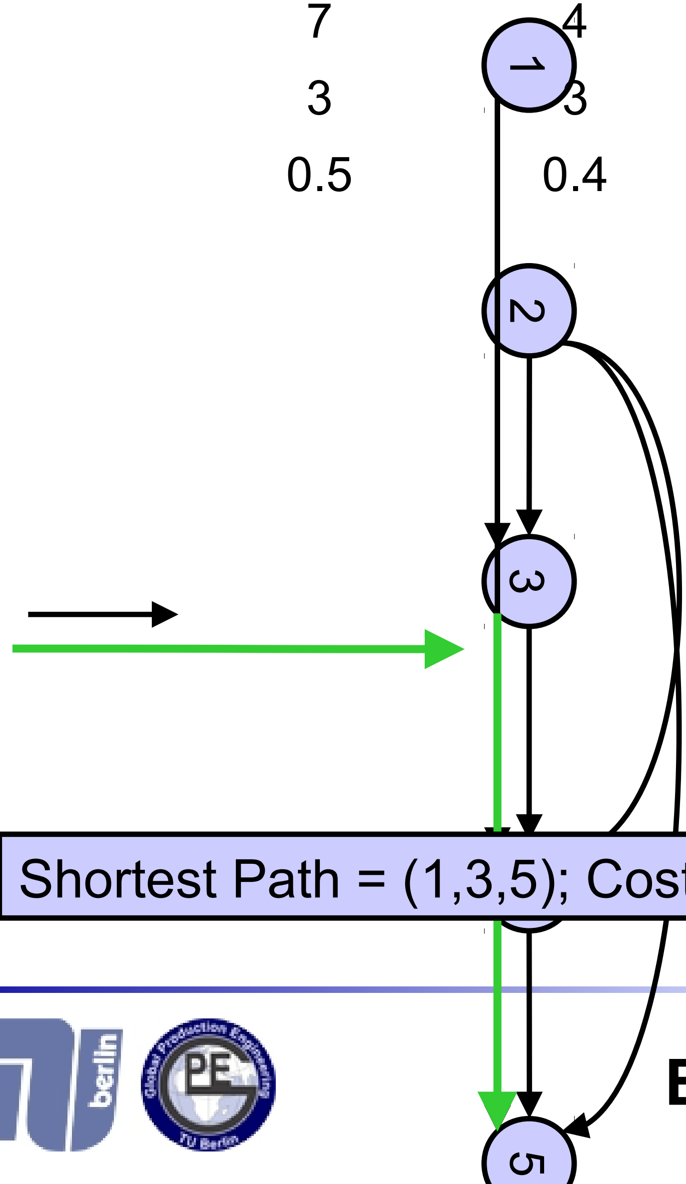
$$\underbrace{c_{ij} = K + \sum_{t=i}^{j-2} h_t \underbrace{\left(\sum_{u=t+1}^{j-1} D_u \right)}_{Inventory}}_{\text{Total Inventory Cost}}$$



- For every node i select at most one outgoing arc (i,j):
  decide that order at time i must satisfy demand until period j-1
  ("no parallel stocks")
- For every node j with $1<j<T+1$:
  (incoming arc selected $\Leftrightarrow$ stock empty at end of period j-1)
  $\Rightarrow$(order at begin of period j $\Leftrightarrow$ select outgoing arc)
- For node 1 select one outgoing arc $\Leftrightarrow$ satisfy first demand
- For node T+1 select one incoming arc $\Leftrightarrow$ satisfy last demand

period	1	2	3	4
D_t	7	4	5	6
K_t	3	3	4	5
h_t	0.5	0.4	0.3	0.4

Arc	c(i,j)	y(I)
(1,2)	3	7
(1,3)	5	11
(1,4)	9.5	16
(1,5)	16.7	22
(2,3)	3	4
(2,4)	5	9
(2,5)	9.2	15
(3,4)	4	5
(3,5)	5.8	11
(4,5)	5	6



Shortest Path = (1,3,5); Cost = 5 + 5.8 = 10.8

Example

Maximum Inventory Volume

- All arcs for which (Production – Demand of period i) is too large have to be removed from network

Maximum Batch Size

- Rule $y_t \, x_{t-1} = 0$ does not hold anymore
- New Rule: $y_t \, (B_t - y_t) \, x_{t-1} = 0$
- NP-hard problem

Fixed Batch Size

- Produce either 0 or B in each time period (all-or-nothing strategy)
- Known as Discrete Lot Sizing (DLS)

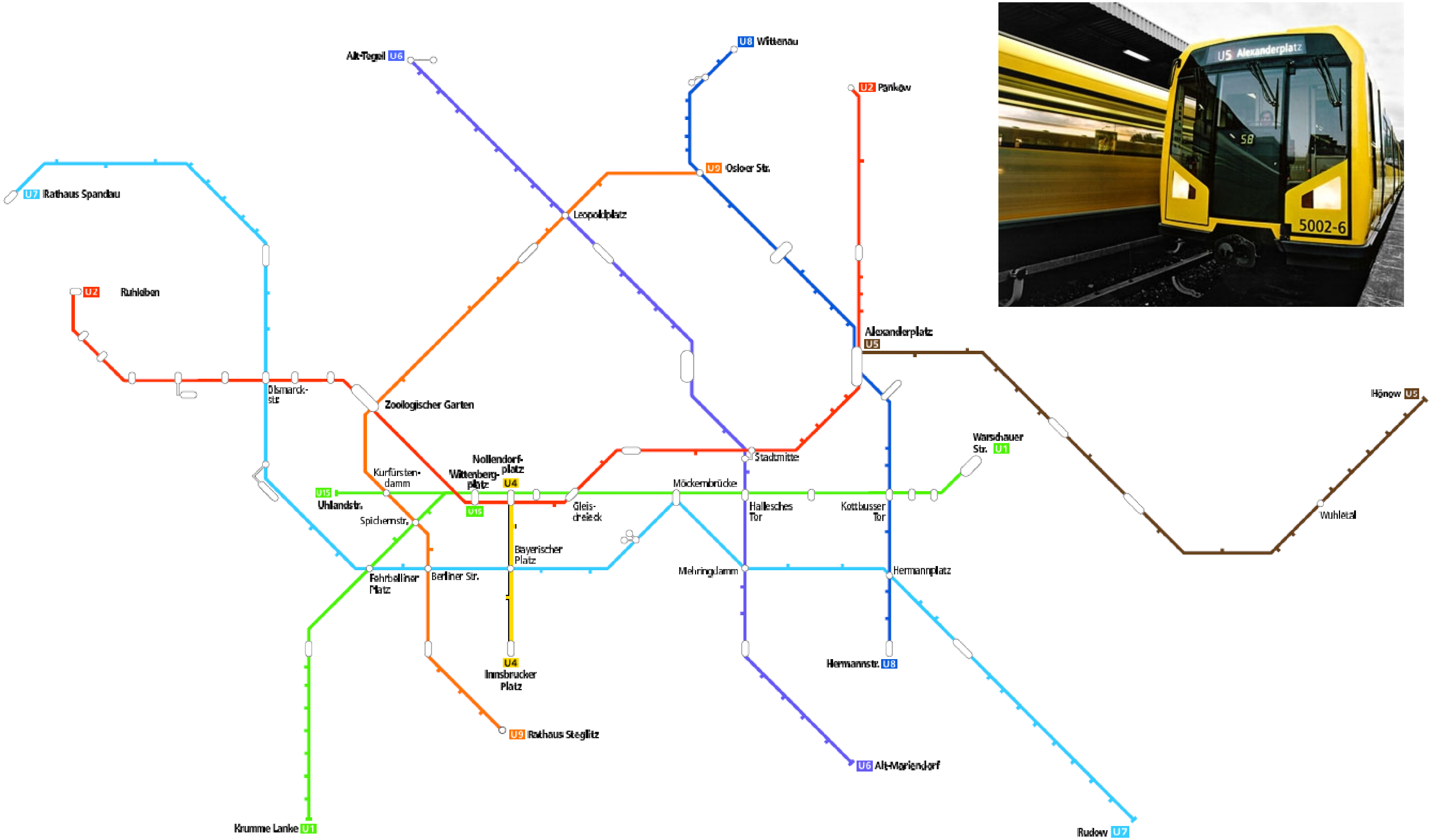
Backordering

- Generalized efficient algorithm

- Constant Demand is very easy
- Variable Demand can be solved by shortest path computation
- Production limitations makes problem NP-hard
- Demand is stochastic
- Multi-product Lot-Sizing

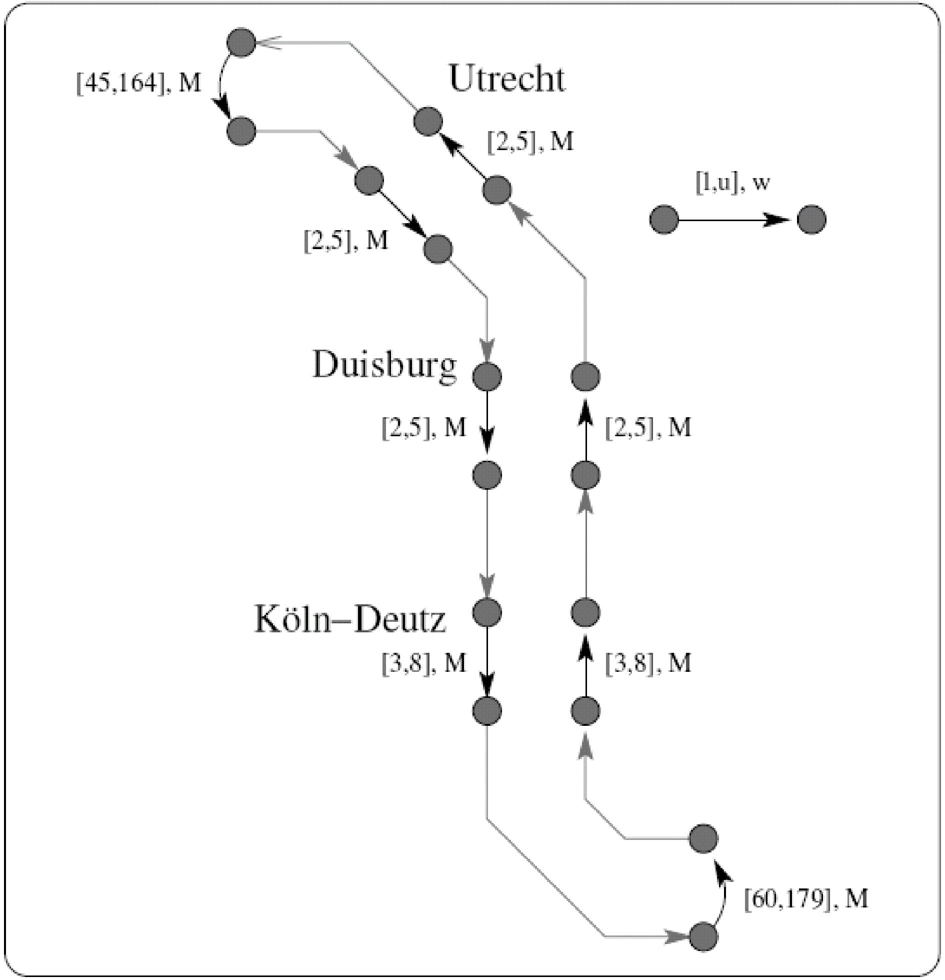
- Multi-criteria Optimization
- Multi-criteria Linear Problem
- Multi-criteria Integer Problem

Zug		RB 27945	RE 33105	IC 2571	RB 27949	IR 2285	RE 33107	EC 175	RB 27953
von			Rostock	Hamburg			Rostock	Aarhus	
Berlin-Spandau			11 15	8 11 21			13 15	13 23	
Berlin Zoologischer Garten			11 26	11 35		13 02	13 26	13 35	
Berlin Friedrichstr			11 33				13 33		
Berlin Alexanderplatz			11 36				13 36		
Berlin Ostbahnhof			11 41	11 49		13 15	13 41	13 49	
Berlin-Karlshorst			11 48				13 48		
Berlin-Schönefeld Flughafen ✈	204		12 00	12 06		13 31	14 00	14 06	
Blankenfelde (Teltow-Fläming)	205		12 08				14 08		
Dahlewitz			12 10				14 10		
Rangsdorf			12 20				14 20		
Dabendorf			12 25				14 25		
Zossen			12 28				14 28		
Wünsdorf-Waldstadt			12 34				14 34		
Neuhof (b Zossen)			12 38				14 38		
Baruth (Mark)			12 45				14 45		
Klasdorf			12 49				14 49		
Golßen (Niederlausitz)			12 53				14 53		
Drahnsdorf			12 58				14 58		
Luckau-Uckro	205		13 04				15 04		
Walddrehna			13 12				15 12		
Doberlug-Kirchhain	205.215.520 7 ○		13 23			14 19	15 23		
Doberlug-Kirchhain			13 24			14 23	15 24		



Berlin Underground Network

TU Berlin
Rüdiger Stephan



Timetables have influence on costs

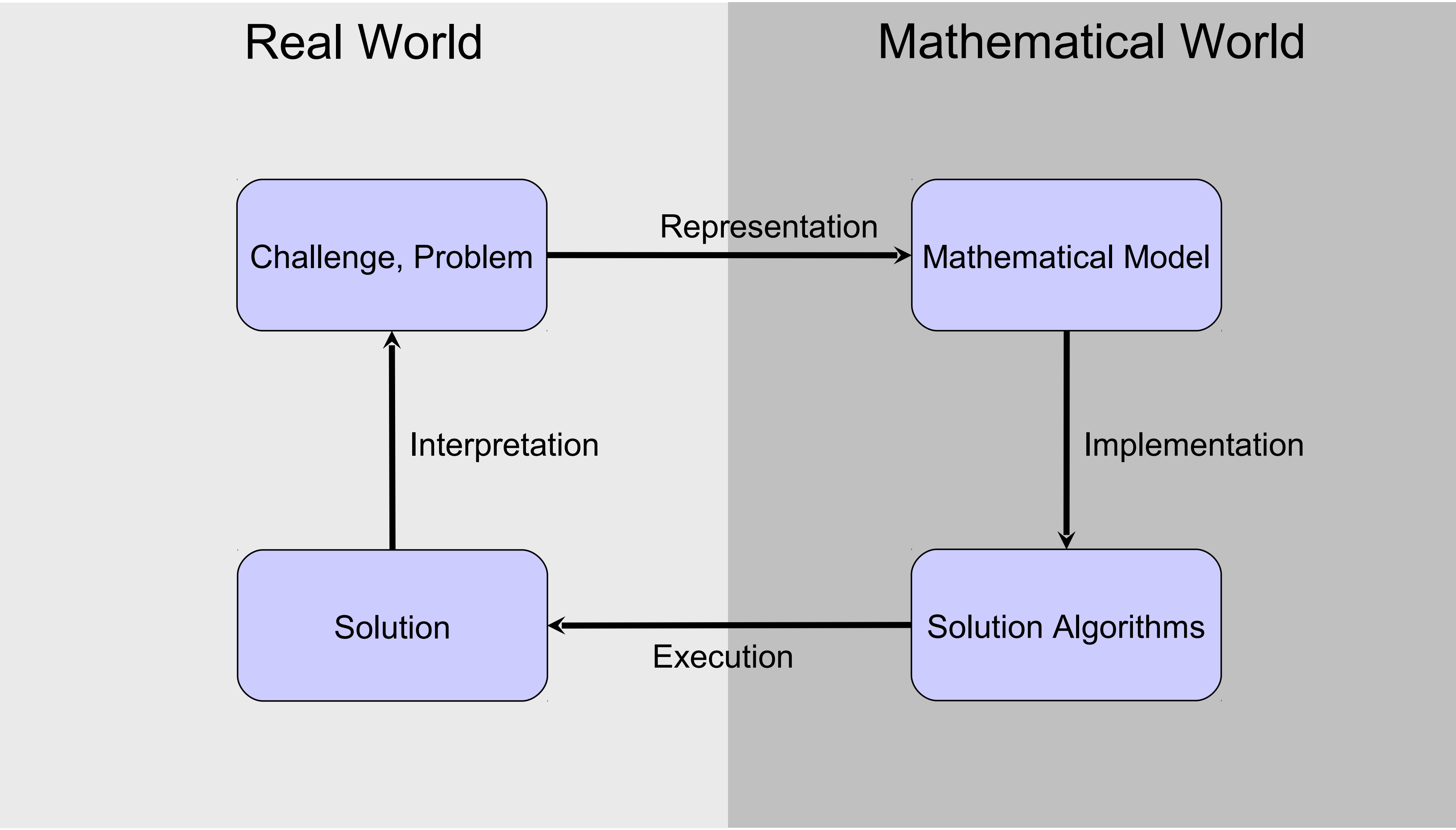
Problem: timetable has influence on passenger waiting time **AND** costs

- Short passenger waiting times at transfers are a very natural goal
- Removing timetables with long waiting times, we could miss some of small cost
- passenger waiting times should be small
- But small number of trains is sought, too
- Multi-criteria Optimization!

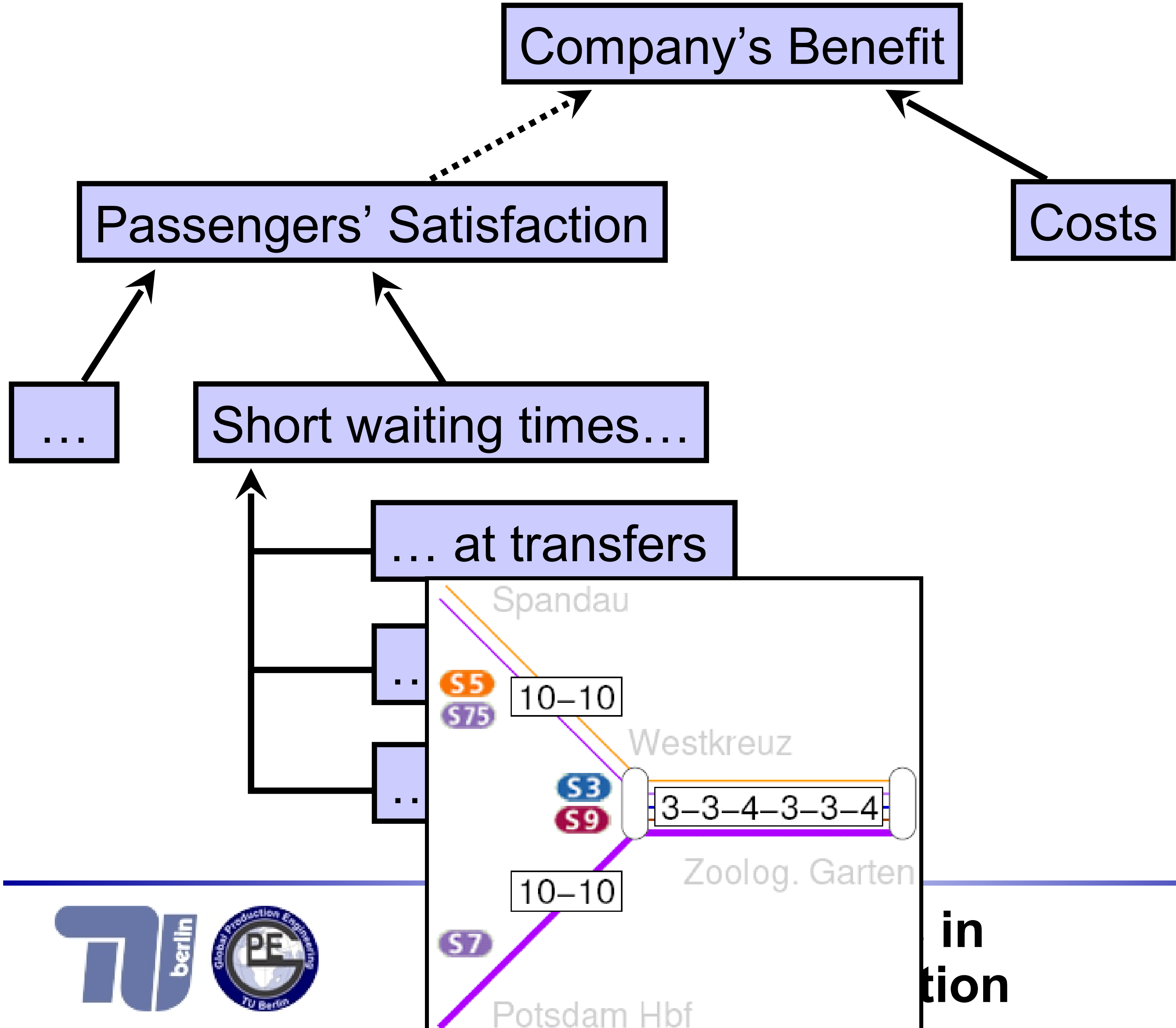
- **Minimize number of trains**
  - subject to restricted passenger waiting time

- **Minimize passenger waiting time**
  - subject to limited number of trains

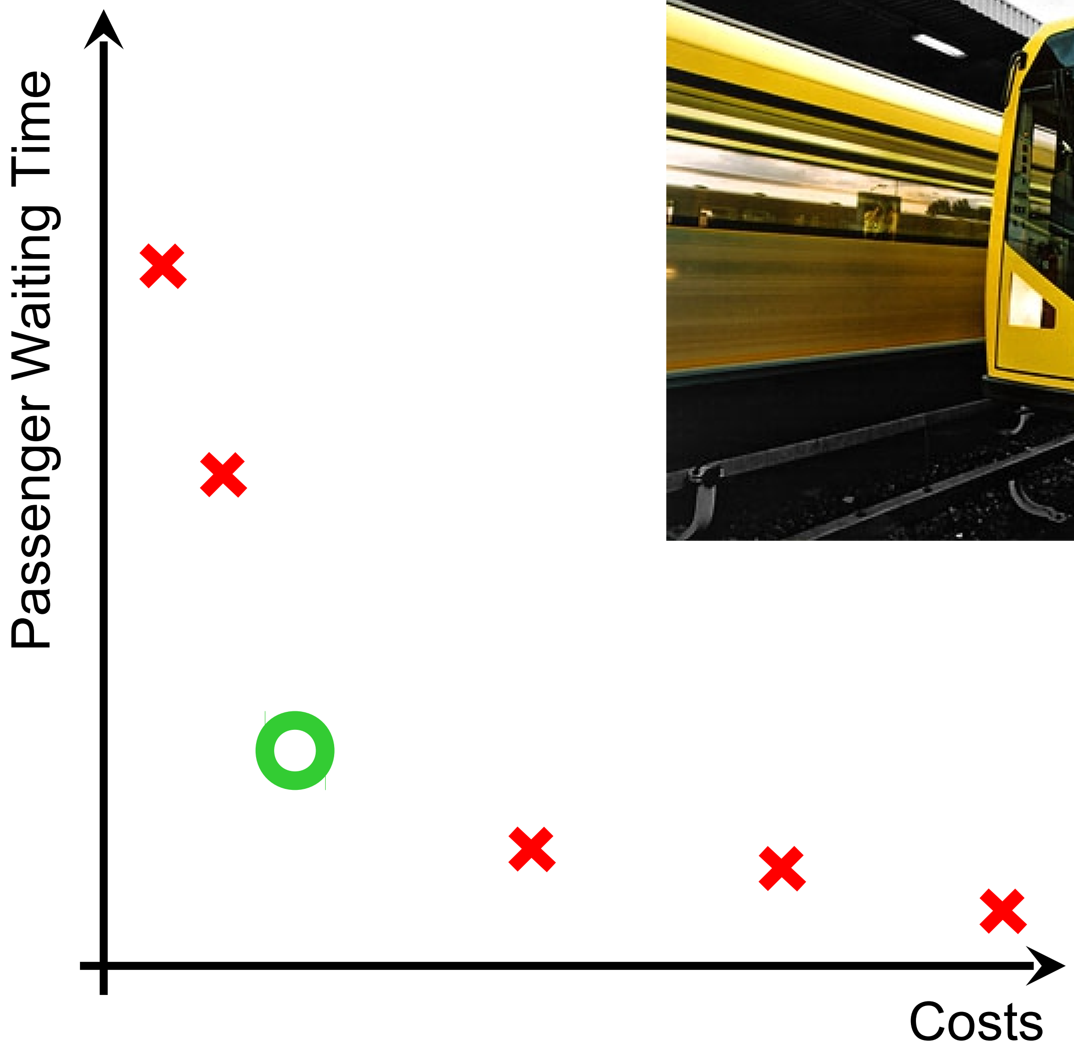
- **Minimize weighted sum of passenger waiting time and number of trains**



Modeling & Solution Cycle


Company's Benefit
Passengers' Satisfaction
Costs
…
Short waiting times…
… at transfers
..
..
Spandau
S5
S75
10–10
Westkreuz
S3
S9
3–3–4–3–3–4
Zoolog. Garten
10–10
S7
Potsdam Hbf

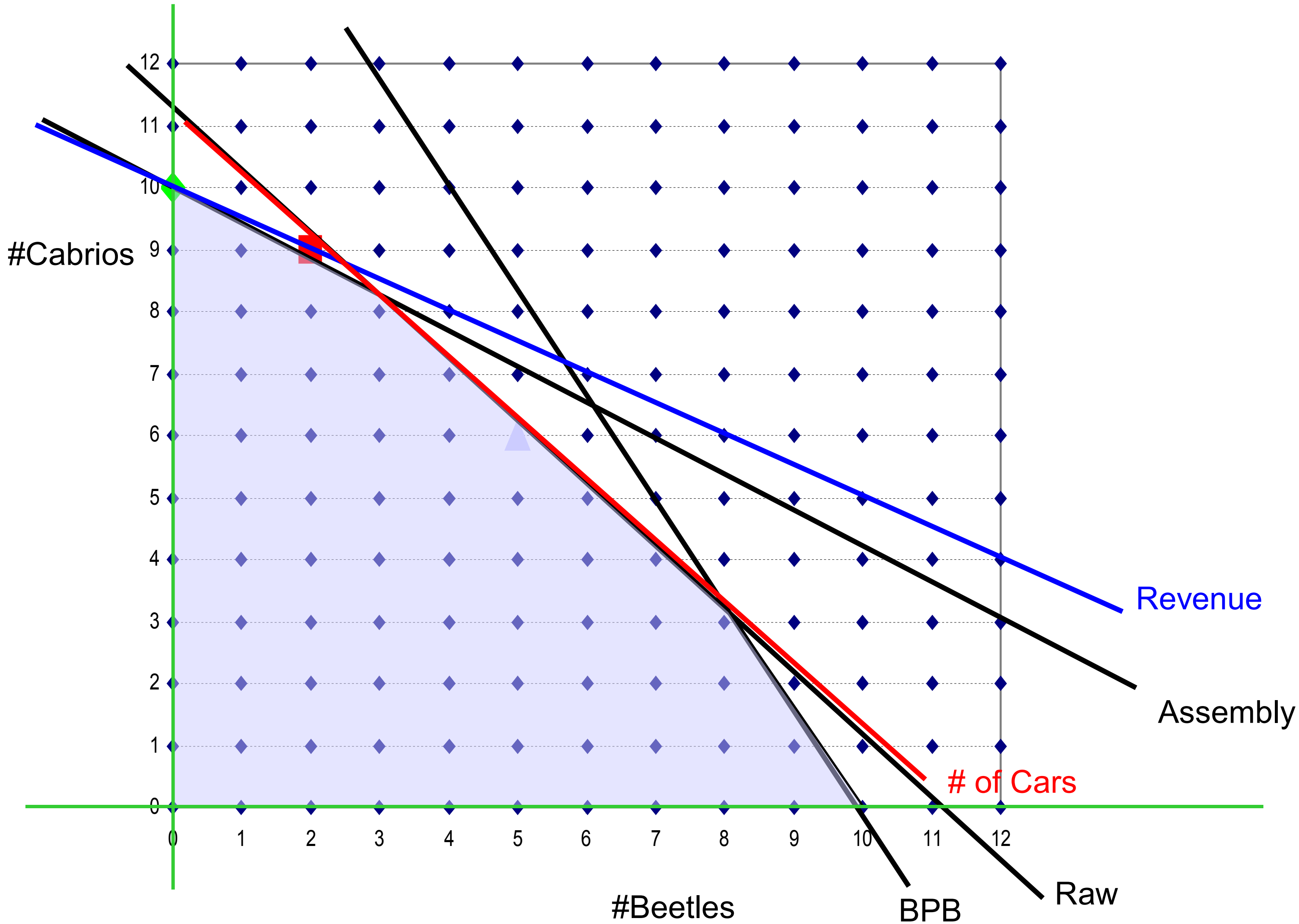
in
tion



Properties of some timetables

TU Berlin
Rüdiger Stephan

- Multi-criteria Optimization
- Multi-criteria Linear Problem
- Multi-criteria Integer Problem



Graphical Representation

TU Berlin
Rüdiger Stephan

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

Multiple Objective functions

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

$$l_j \le x_j \le u_j \qquad j = 1, \ldots, n$$

A feasible solution ***x*** is referred to as an **efficient (non-dominated) solution** if there is no feasible solution ***y*** such that

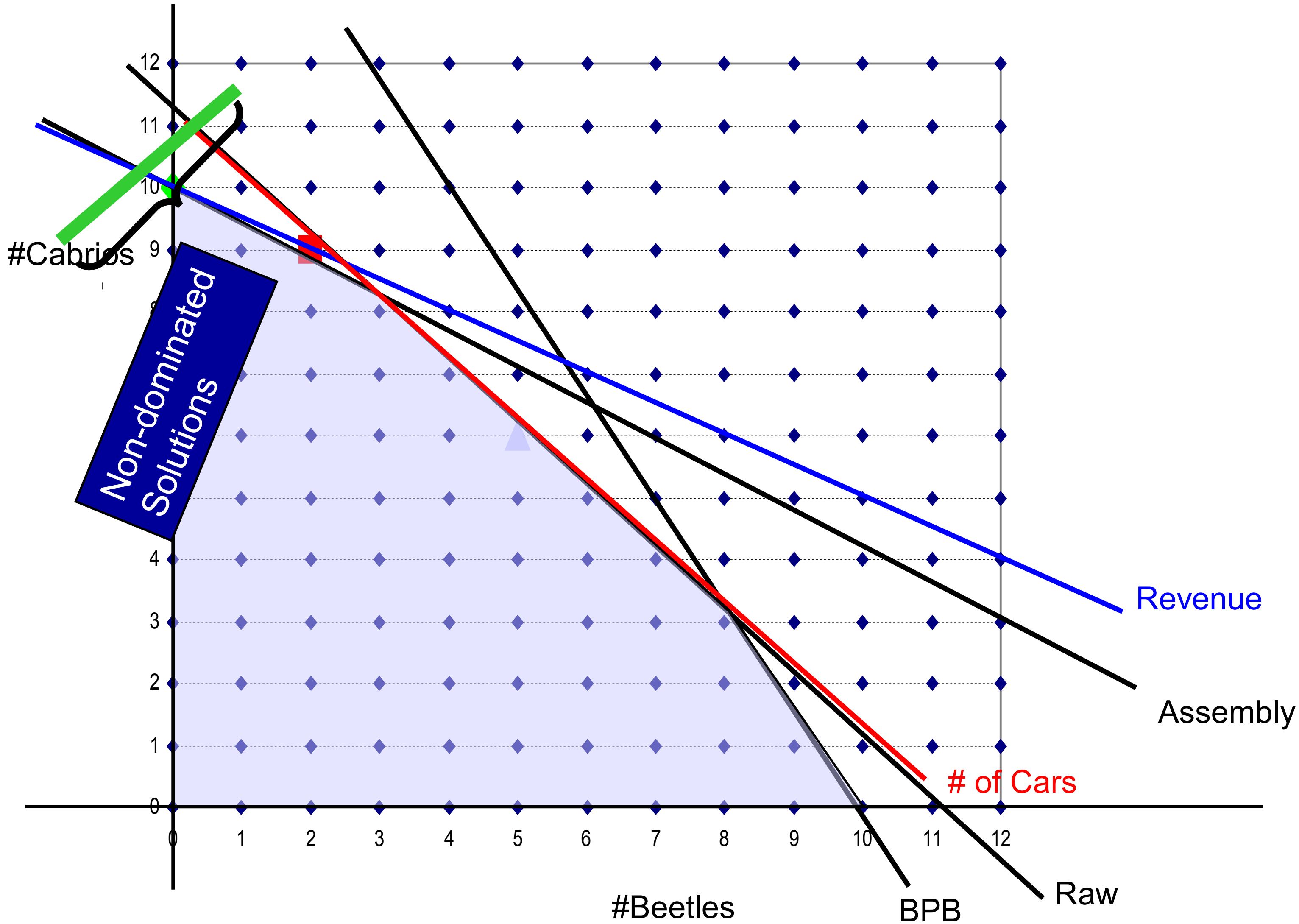
$$\sum_{j=1}^{n} c_j^p y_j \geq \sum_{j=1}^{n} c_j^p x_j \quad p = 1, \ldots, q$$

and

$$\sum_{j=1}^{n} c_j^r y_j > \sum_{j=1}^{n} c_j^r x_j \text{ for some } r, 1 \leq r \leq q$$

⇨ **No objective can be improved without reduction of one of the other objectives**

Efficient (Non-dominated) Solutions



Non-dominated Solutions

TU Berlin
Rüdiger Stephan

Combine all objectives into one objective by taking a linear combination

$$\max \sum_{p=1}^{q} \lambda^p \left[\sum_{j=1}^{n} c_j^p x_j \right]$$

$$\lambda^1, \ldots, \lambda^q > 0$$

Single objective linear program

$$\max \sum_{j=1}^{n} \bar{c}_j x_j \quad \text{with} \quad \bar{c}_j = \sum_{p=1}^{q} \lambda^p c_j^p$$

x is efficient if and only if there exists λ_p such that x is optimal for single obj. LP

Idea: Maximize one objective subject to bounds on all other objectives

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

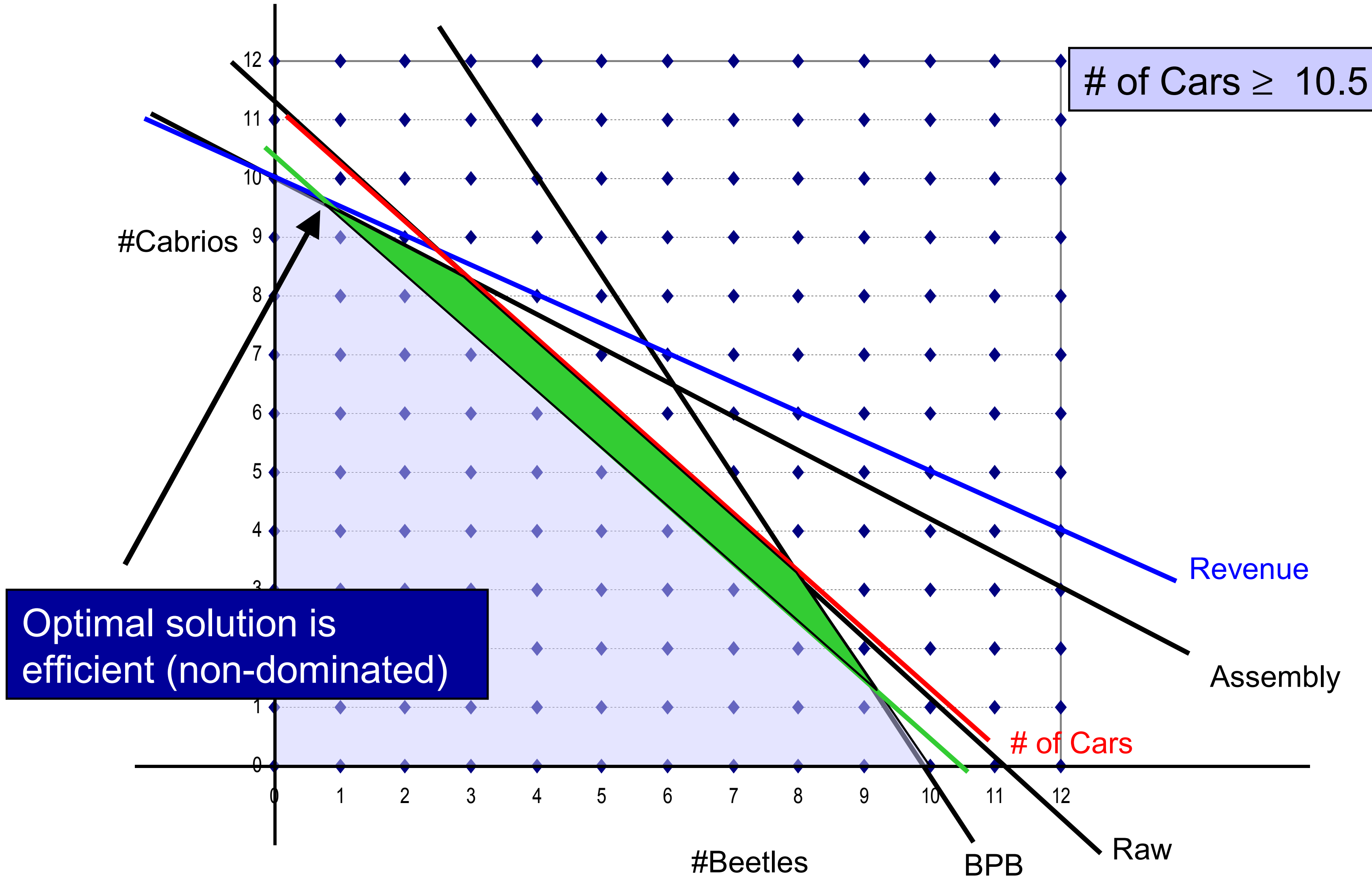
r-th objective optimized

$$\text{s.t.} \quad \sum_{j=1}^{n} c_j^p x_j \geq L^p \qquad p = 1, \ldots, q, p \neq r$$

Lower bound for p-th objective

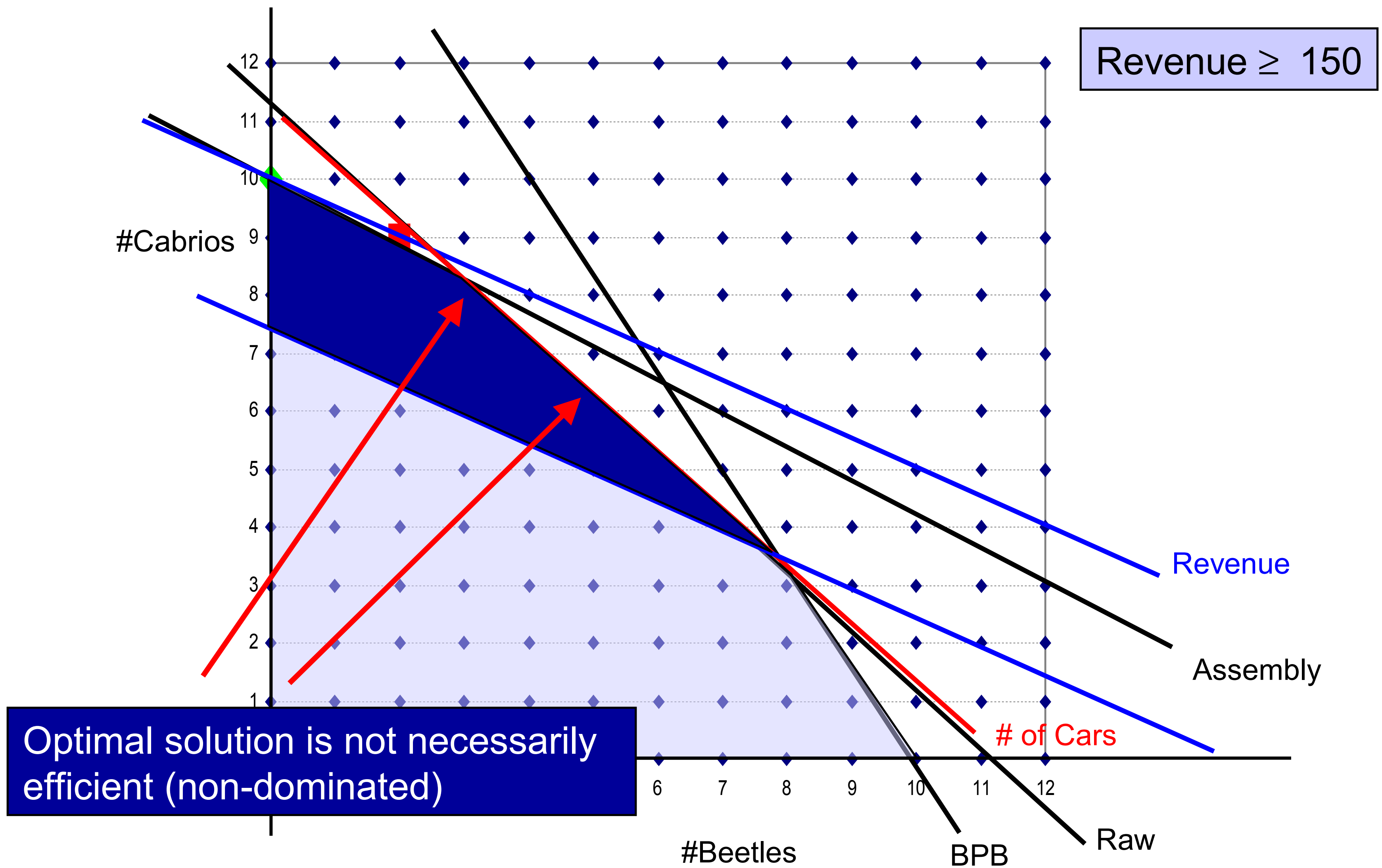
$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1, \ldots, m$$

$$l_j \leq x_j \leq u_j \qquad j = 1, \ldots, n$$



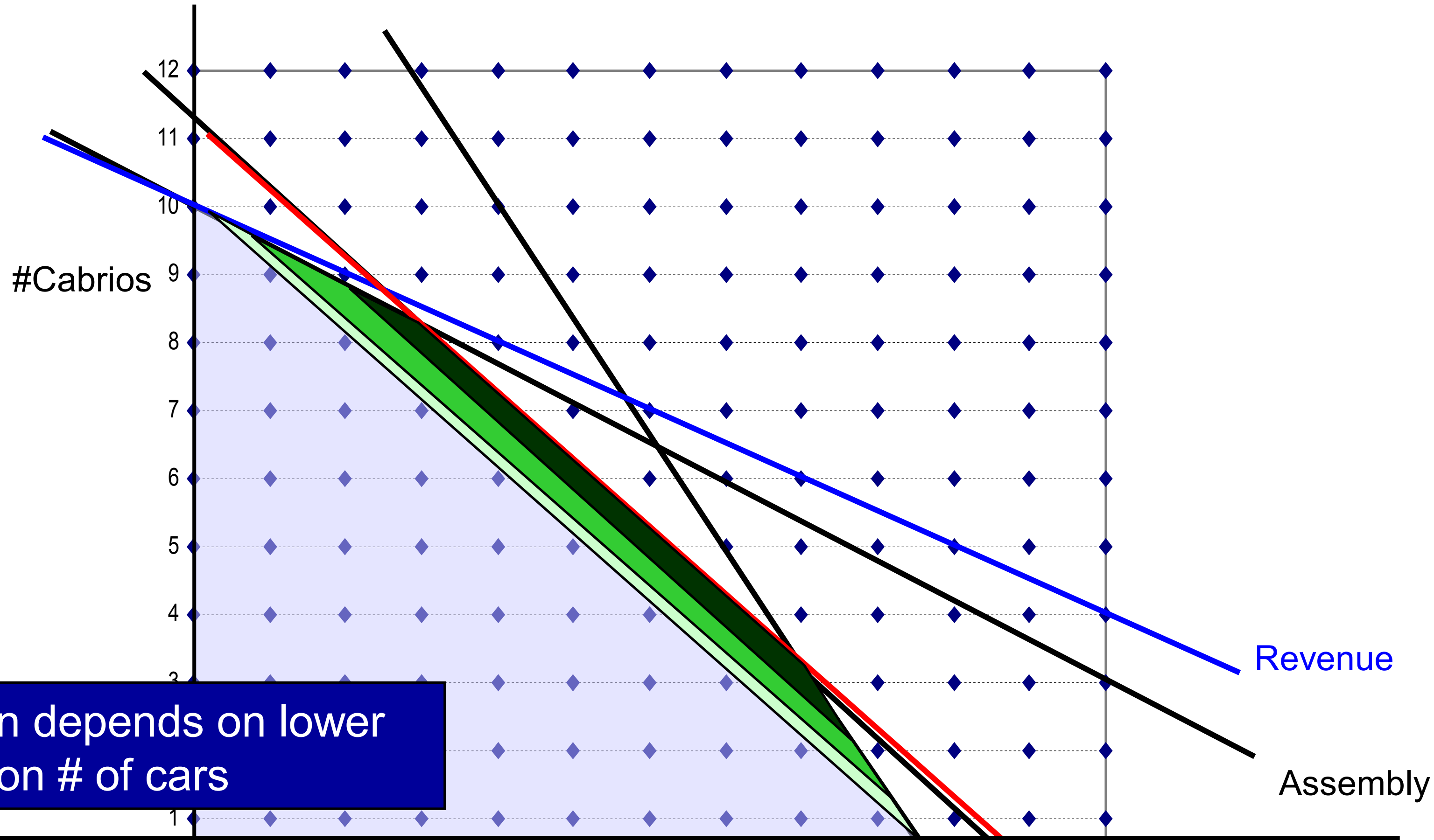
Maximize Revenue subject to Minimum Production value

TU Berlin
Rüdiger Stephan



Maximum Production subject to Minimum Revenue Value

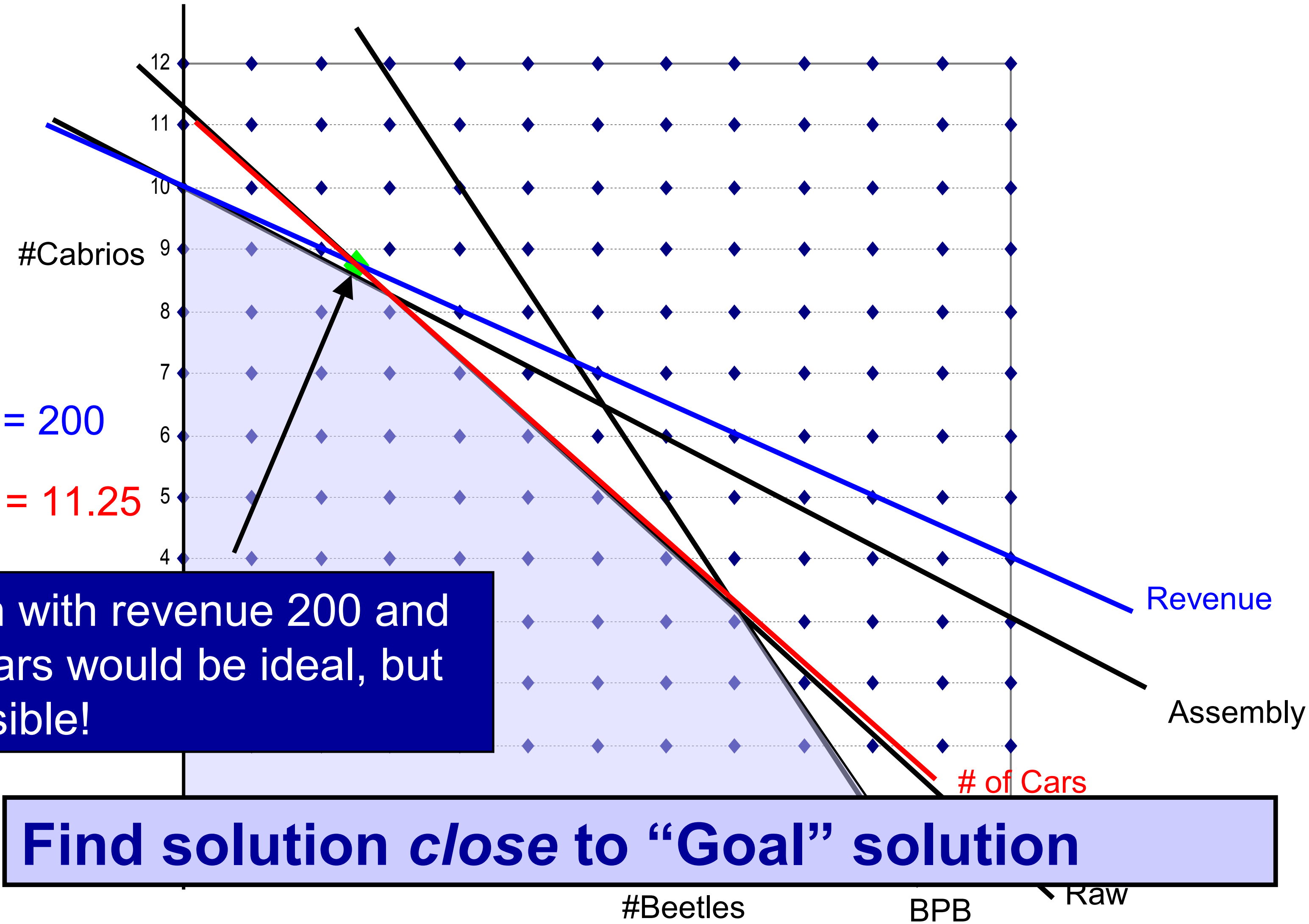
TU Berlin
Rüdiger Stephan



Solution depends on lower bound on # of cars

Trade-off between two/more objectives: decision support tool

Trade-Off between objective and lower bounds



Optimum = 200

Optimum = 11.25

Solution with revenue 200 and 11.25 cars would be ideal, but is infeasible!

Find solution *close* to "Goal" solution

Goal Programming

- **Efficient (non-dominated) solutions**
- **Method 1: Linear Combination of Objectives**
- **Method 2: Single Objective with Constraints for other objectives**
- **Method 3: Goal Programming**

TU Berlin
Rüdiger Stephan

- **Multi-criteria Optimization**
- **Multi-criteria Linear Problem**
- **Multi-criteria Integer Problem**

- Method 1: Linear Combination of Objectives
- Method 2: Single Objective with Constraints for other objectives
- Method 3: Goal Programming

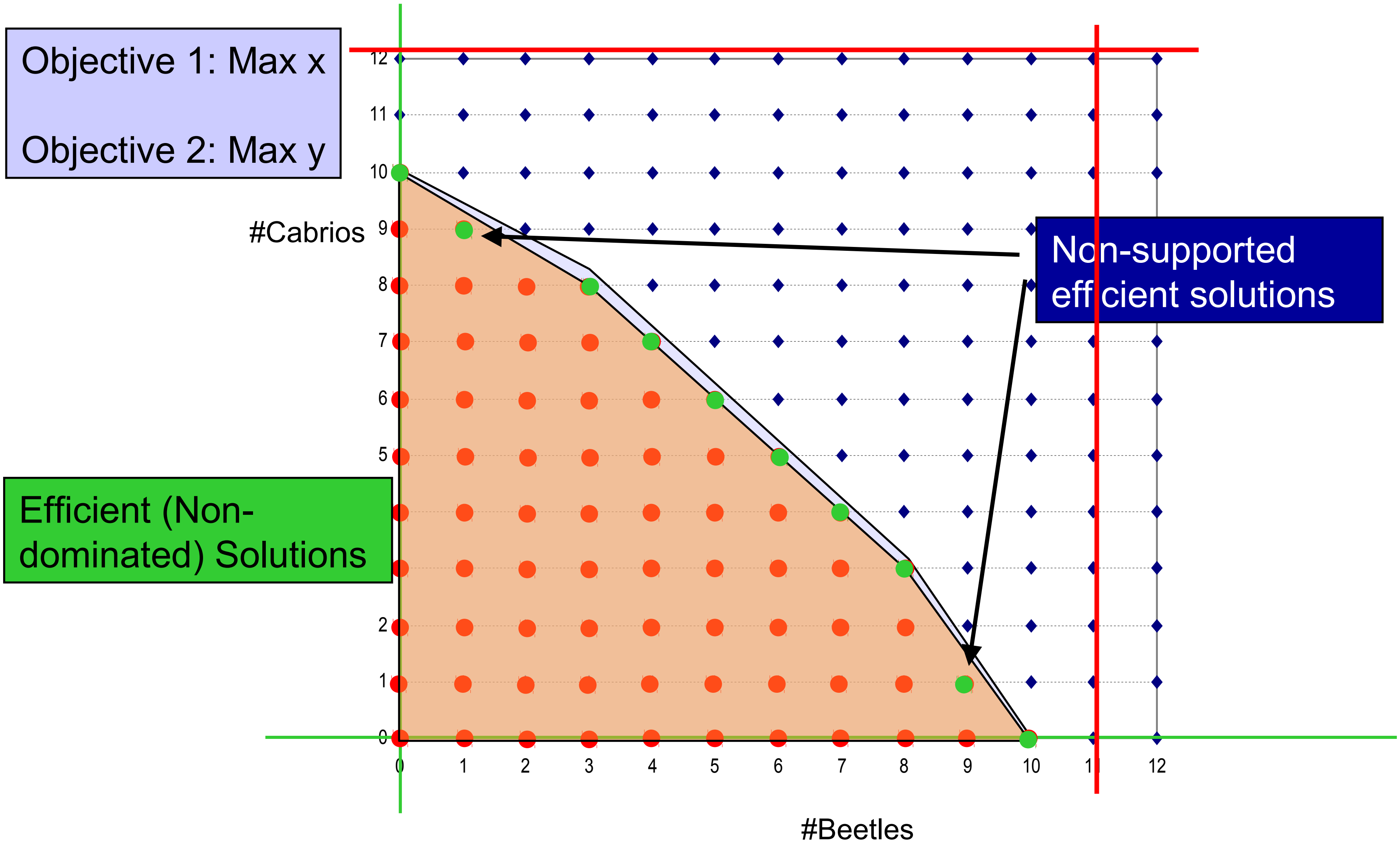
Linear Programming

x is efficient if and only if there exists λ_p such that x is optimal for single obj. LP

Integer Programming

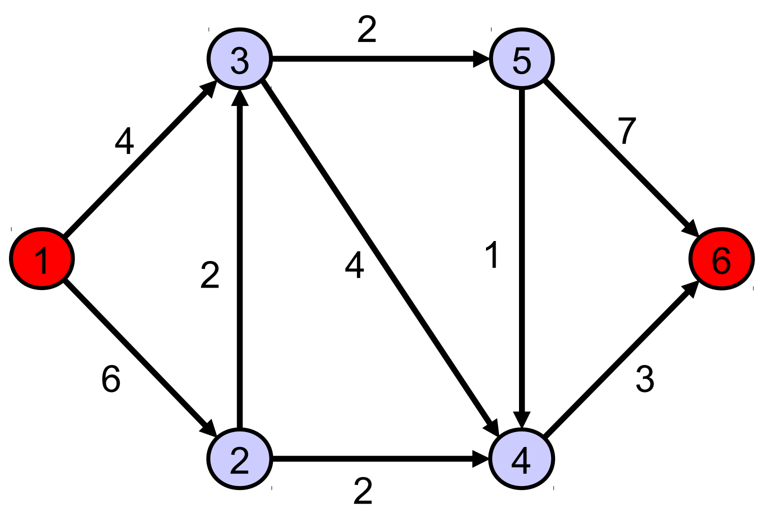
There exist efficient solutions that are not optimal for any linear combination

- **Supported efficient (SE)** solutions: efficient solutions that are optimal for a linear combination of objectives
- **Non-supported efficient (NE)** solutions: efficient solutions that are not optimal for any linear combination of objectives

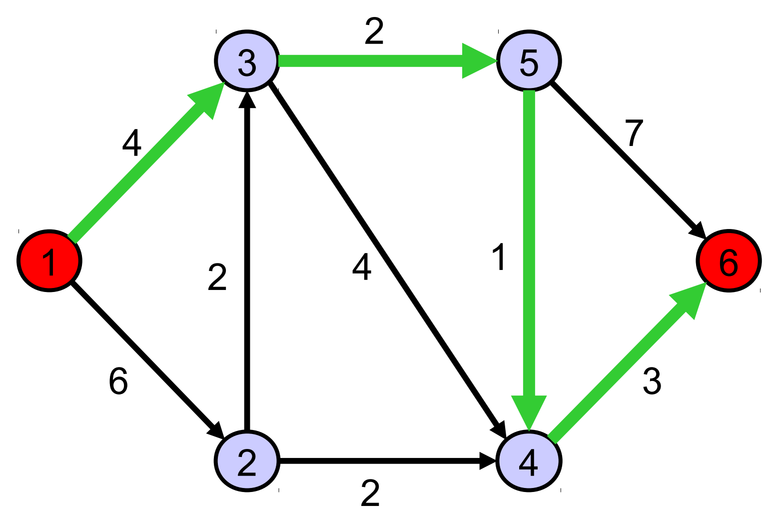


Non-supported Efficient Solutions

Shortest path from 1 to 6 ?



Solution:



- Minimize Length of path from 1 to 6
- Minimize # of hops (= # of nodes on path)

- Min Length = 10
- Min Hops = 4

Constrained Shortest Path Problem

NP-hard in general; polynomial for #hops

TU Berlin
Rüdiger Stephan

- Existence of non-supporting efficient solutions
- Extra Constraint(s) make(s) easy combinatorial problems often NP-hard
- Many more techniques, in particular heuristics