Bus Scheduling and Multicommodity Flows

Ralf Borndörfer

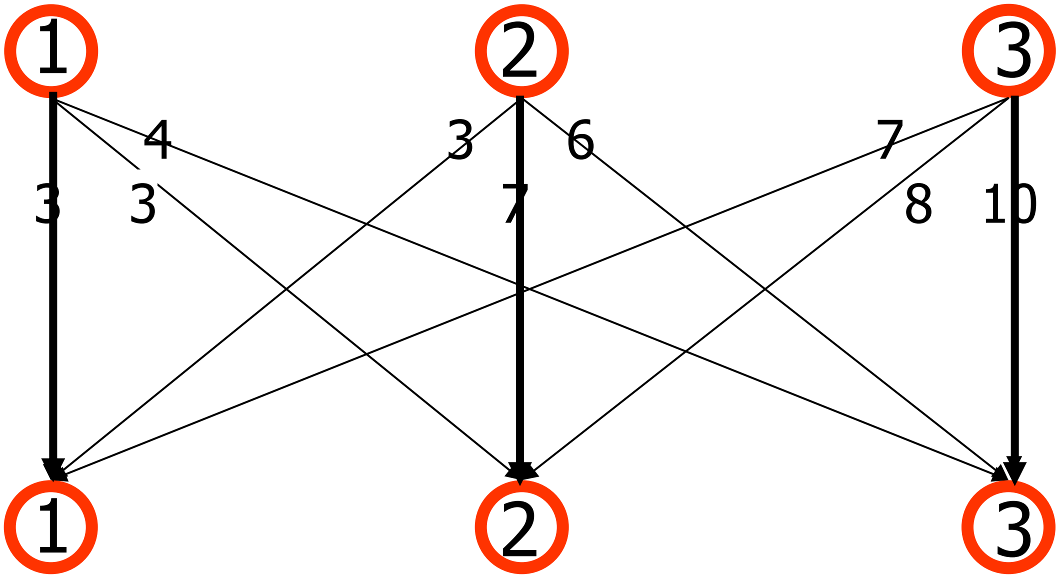
2015 Workshop on
Combinatorial Optimization with Applications in Transportation and Logistics

Beijing, 28.07.2015

Outline

- Optimal Assignments
- Single Depot Vehicle Scheduling
- Multiple Depot Vehicle Scheduling
- Lagrangean Relaxation
- Multicriteria Optimization

The Assignment Problem

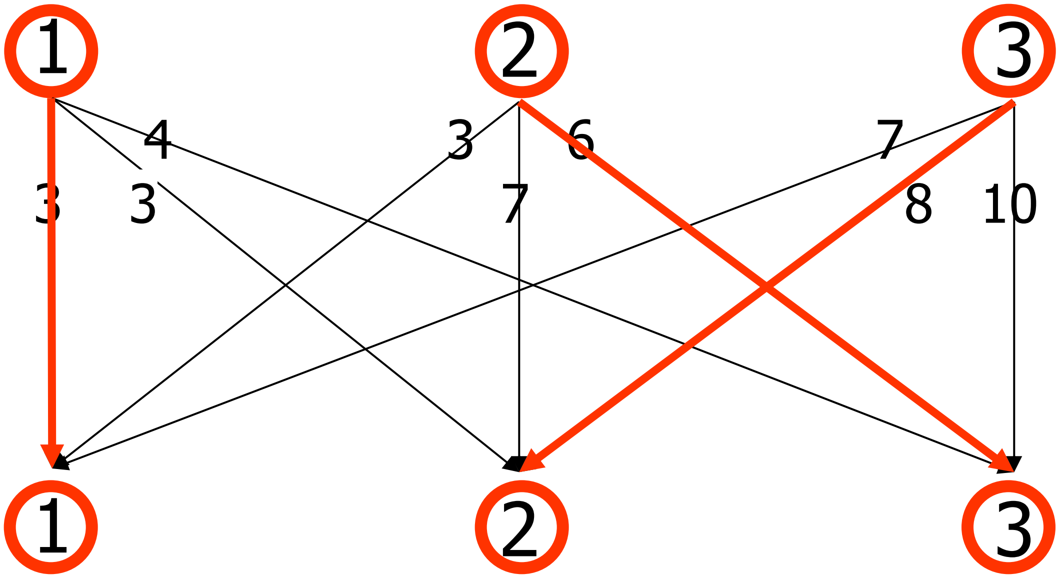


solution
cost = 20

The Problem

- Input: 3 sources, 3 destinations, costs
- Output: cost minimal assignment

A Heuristic

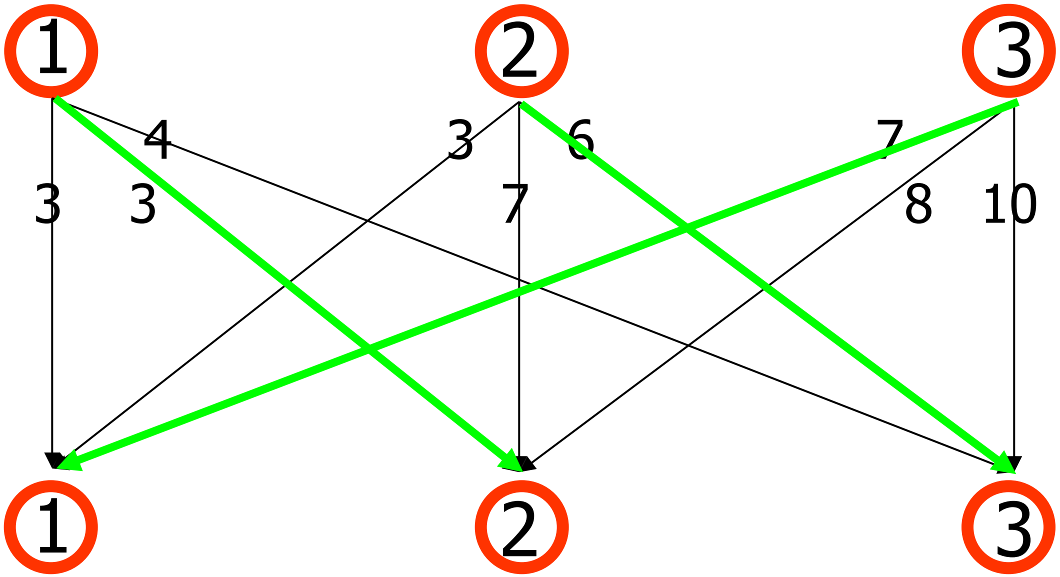


solution
cost = 17

The Greedy Heuristic

- heuretikos (gr.): inventive
  heuriskein (gr.): to find

A Heuristic

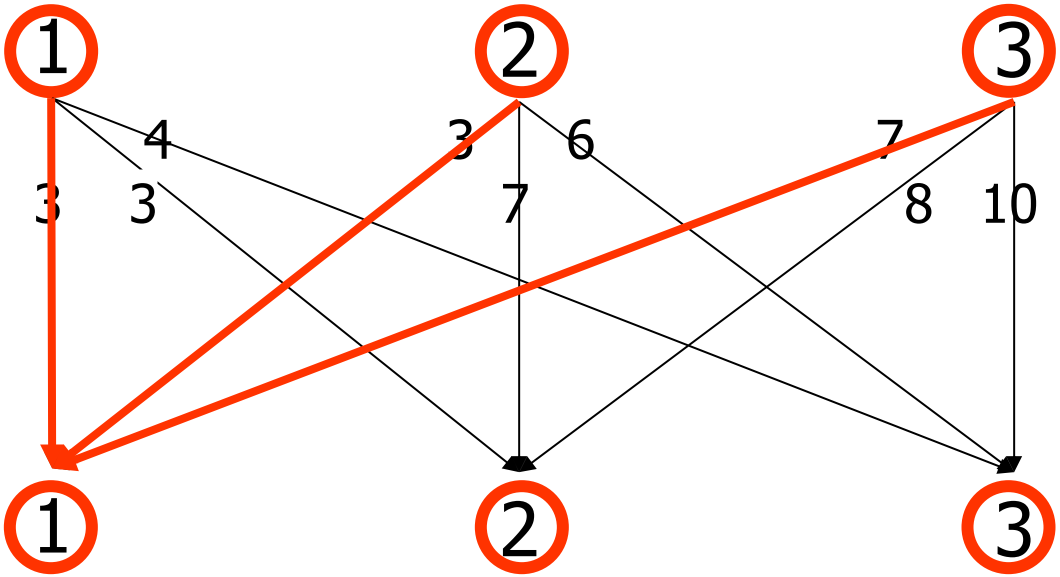


solution
cost = 16

The Greedy Heuristic

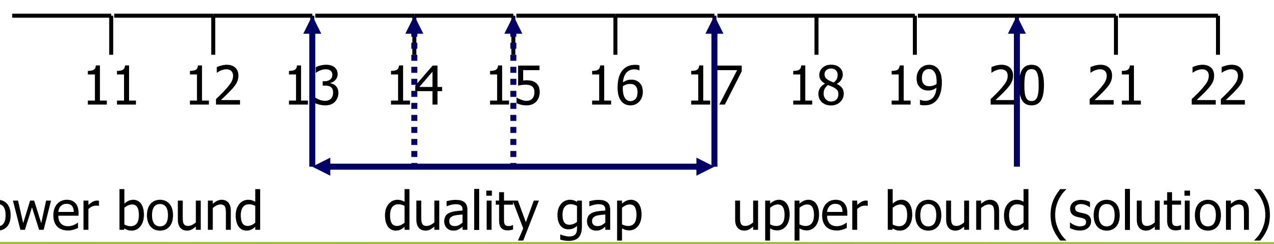
- heuretikos (gr.): inventive
  heuriskein (gr.): to find

A Lower Bound

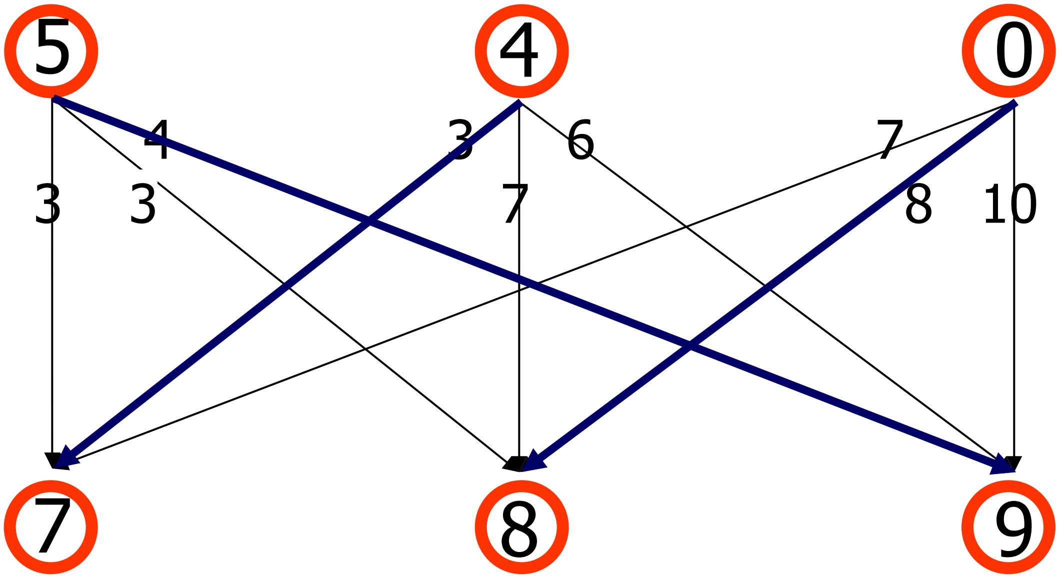


bound
cost = 13
solution
cost = 17
guarantee
4/17=23%
4/13=30%

A "Relaxation"



A Proof of Optimality



optimum
cost = 15

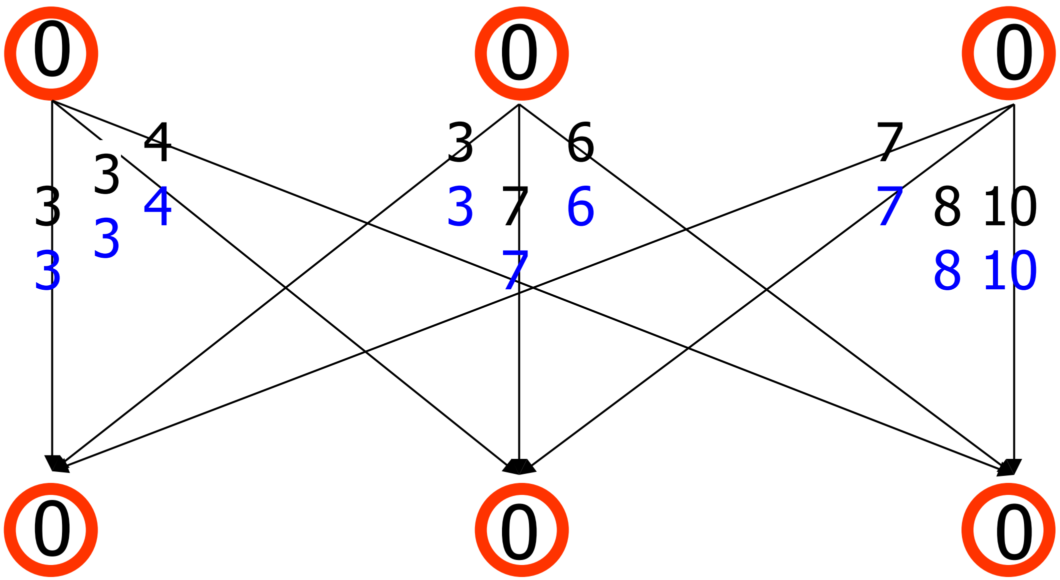
The "primal problem"

- Minimum cost assignment

The "dual problem"

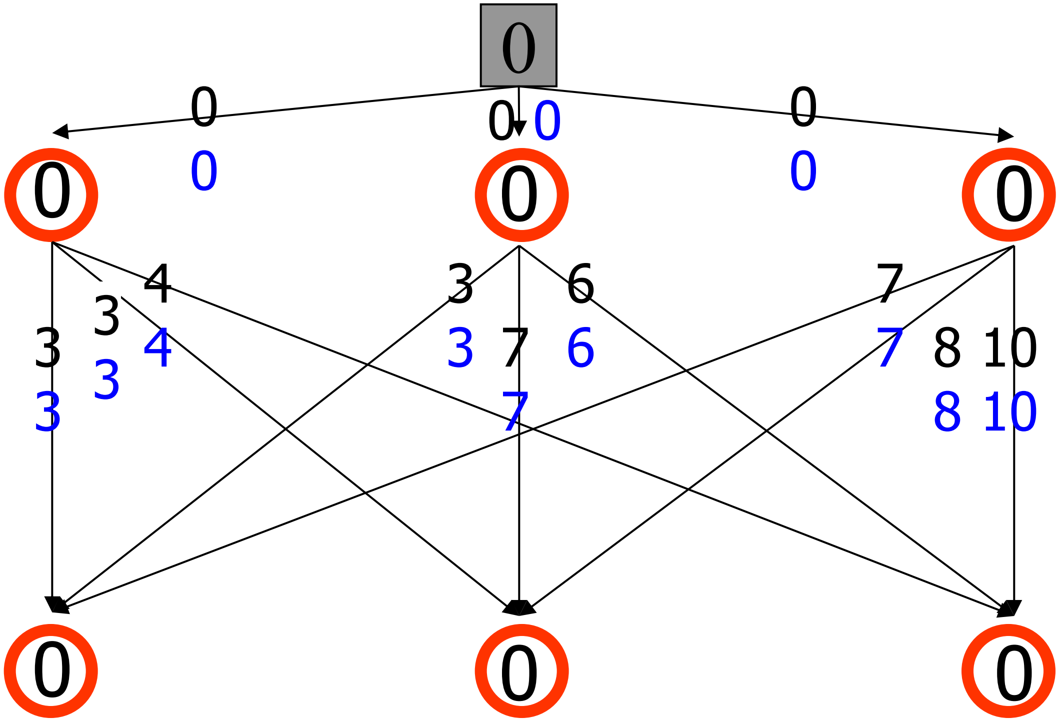
- Maximum profit sales

An Exact Algorithm



The "successive shortest path" algorithm

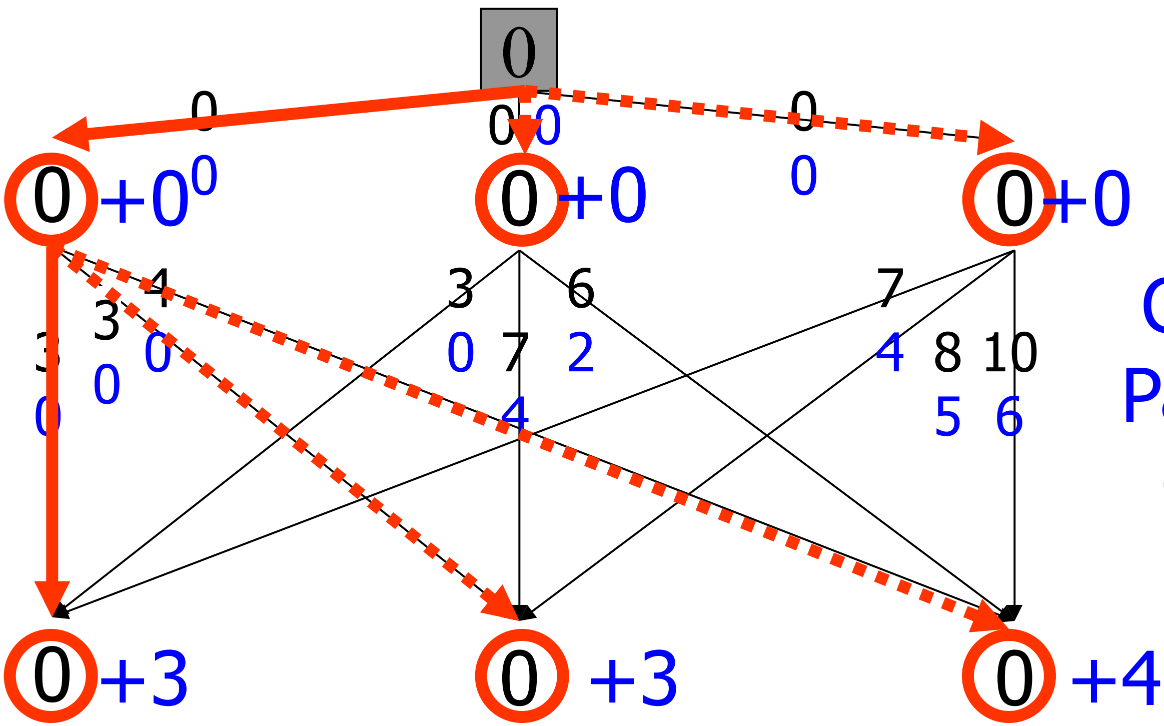
An Exact Algorithm



Bound
Costs: 0
Partial Sol.
Costs: 0

- The "successive shortest path" algorithm

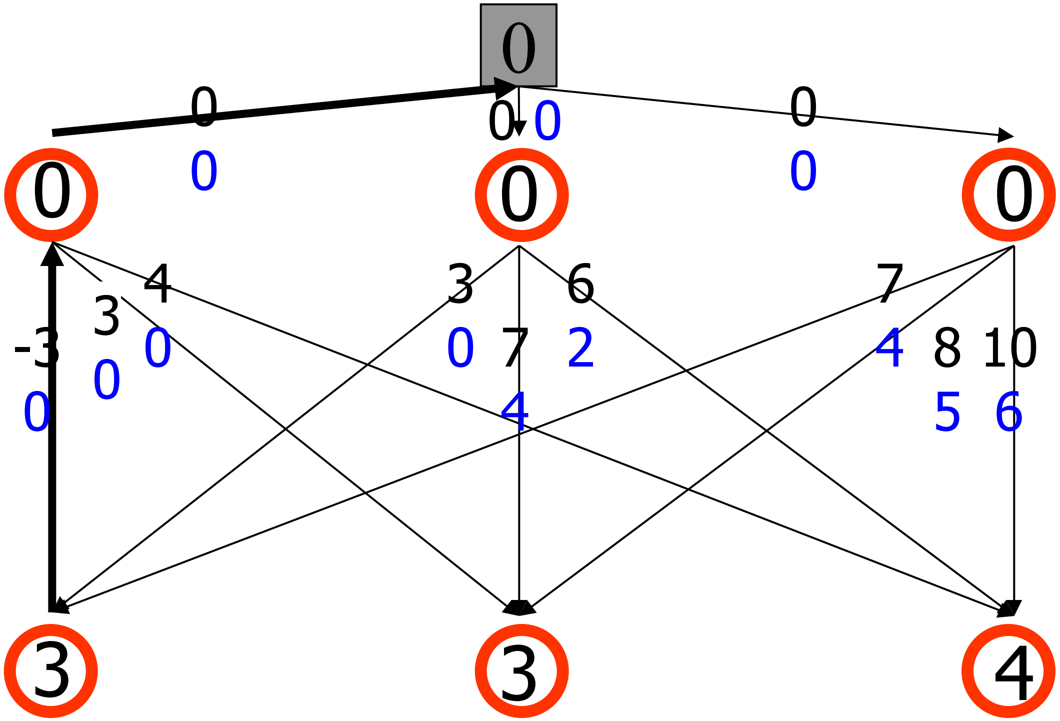
An Exact Algorithm



Bound
Costs: 10
Partial Sol.
Costs: 3

- The "successive shortest path" algorithm

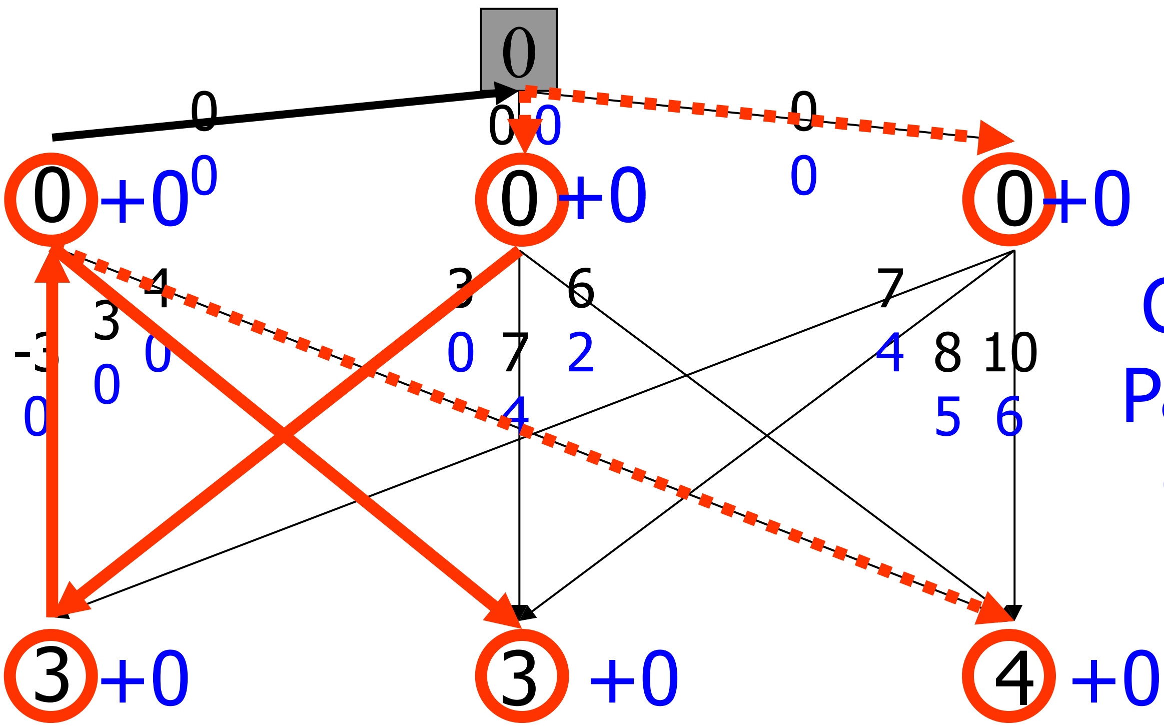
An Exact Algorithm



Bound
Costs: 10
Partial Sol.
Costs: 3

- The "successive shortest path" algorithm

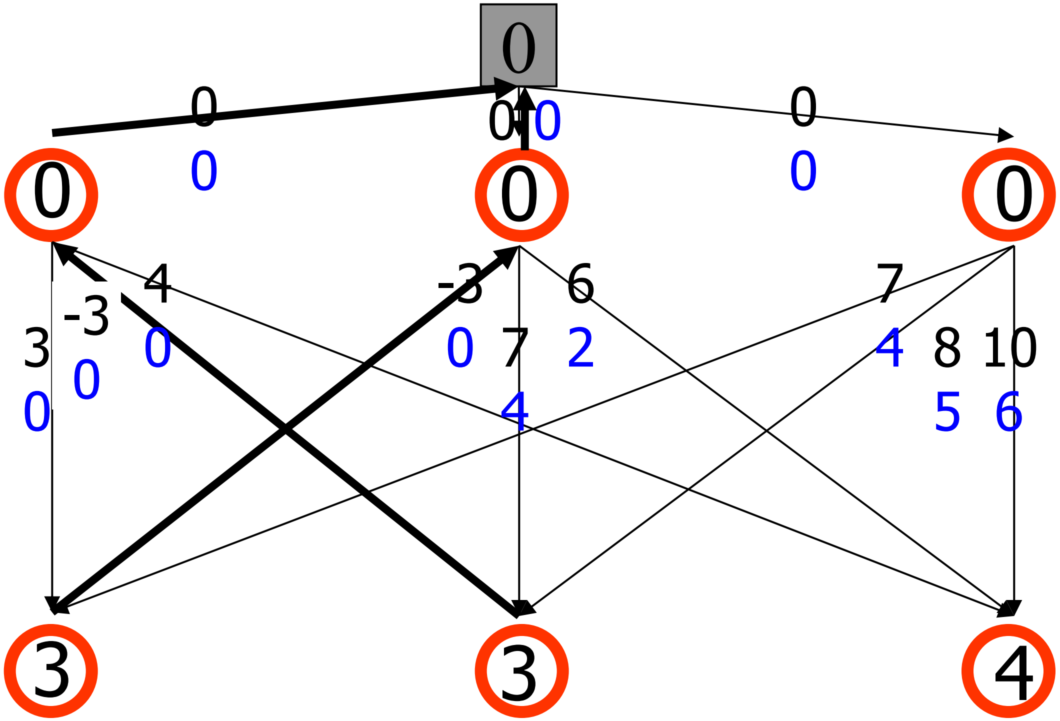
An Exact Algorithm



Bound
Costs: 10
Partial Sol.
Costs: 6

- The "successive shortest path" algorithm

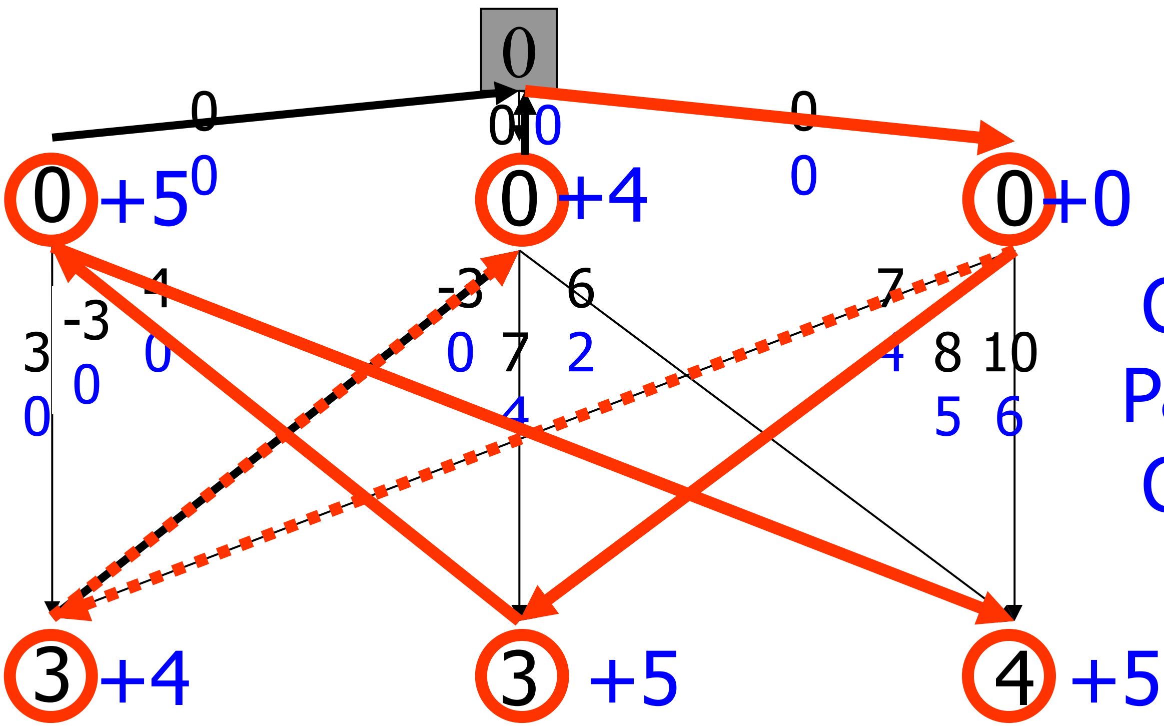
An Exact Algorithm



Bound
Costs: 10
Partial Sol.
Costs: 6

- The "successive shortest path" algorithm

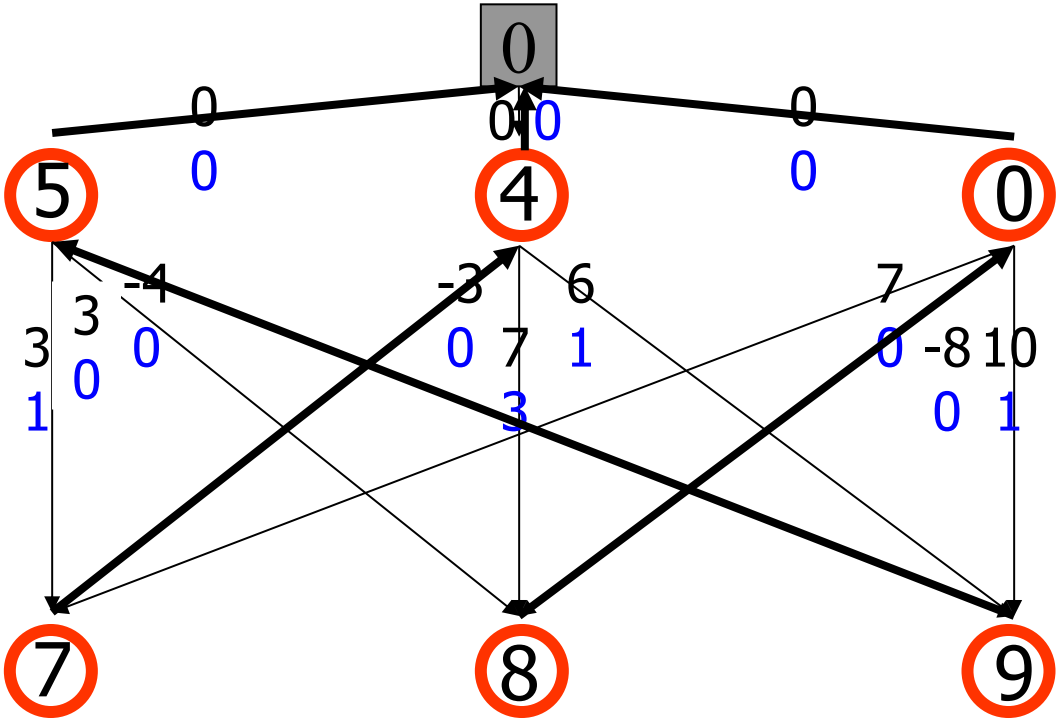
An Exact Algorithm



Bound
Costs: 15
Partial Sol.
Costs: 15

- The "successive shortest path" algorithm

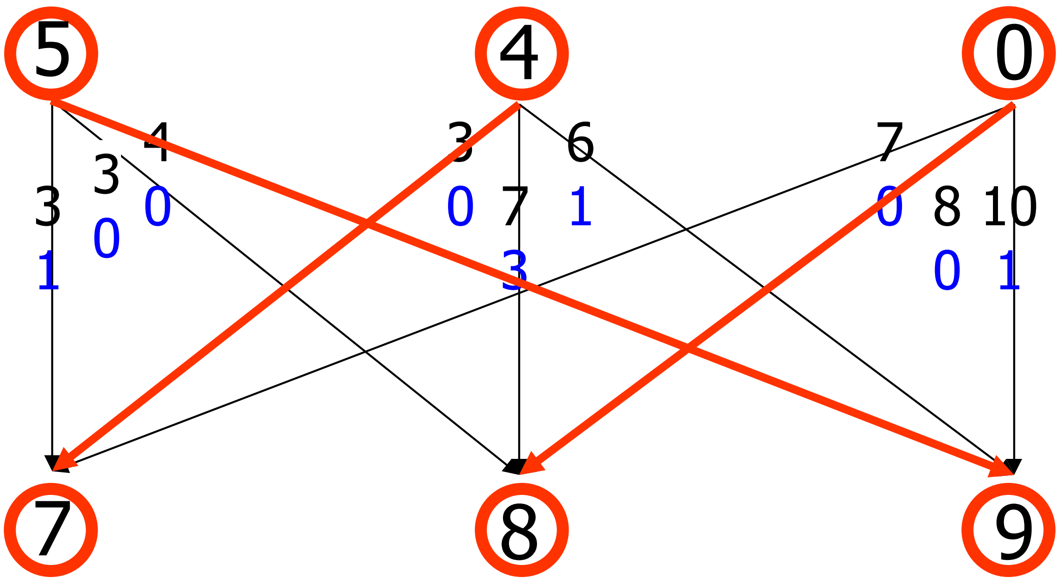
An Exact Algorithm



Bound
Costs: 15
Partial Sol.
Costs: 15

- The "successive shortest path" algorithm

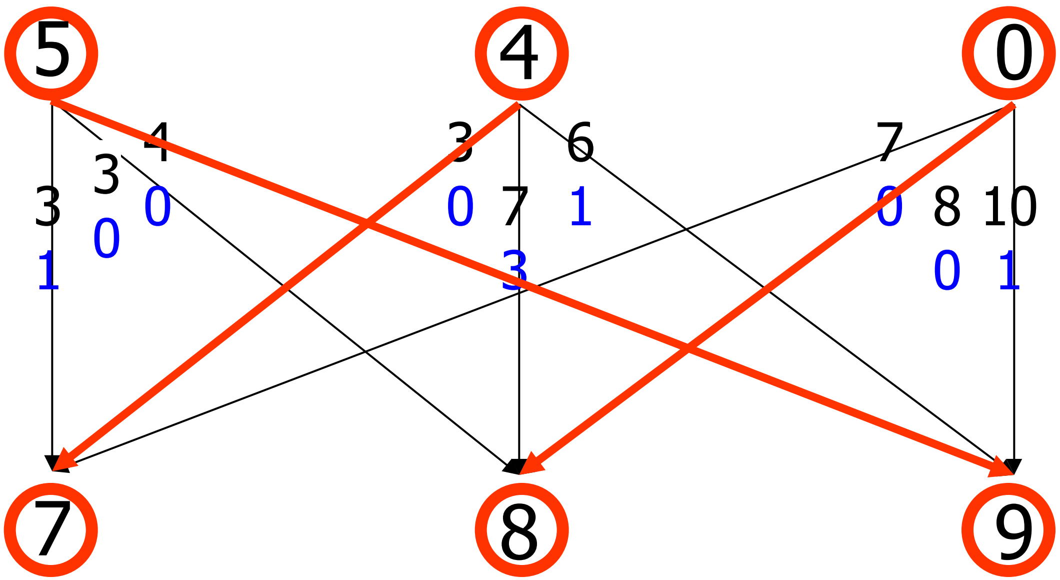
An Exact Algorithm



Bound
Costs: 15
Solution
Costs: 15
Guaranty: 0%
(Optimal)

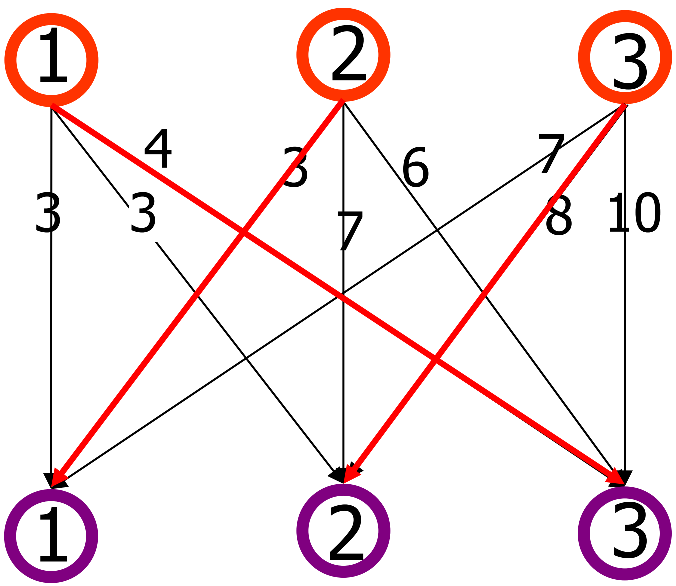
- The "successive shortest path" algorithm computes a shortest path for every source node, i.e., does n shortest path calculations.

An Exact Algorithm



Bound
Costs: 15
Solution
Costs: 15
Guaranty: 0%
(Optimal)

- **Theorem:** The assignment problem can be solved in polynomial time of $O(n^3)$.



Graphen theoretic model

$$
\begin{array}{llllr}
\min & 3x_{11} & +3x_{12} & +4x_{13} & \\
 & +3x_{21} & +7x_{22} & +6x_{23} & \\
 & +7x_{31} & +8x_{32} & +10x_{33} & \\
\text{s.t.} & x_{11} & +x_{12} & +x_{13} & =1 \\
 & x_{21} & +x_{22} & +x_{23} & =1 \\
 & x_{31} & +x_{32} & +x_{33} & =1 \\
 & x_{11} & +x_{21} & +x_{31} & =1 \\
 & x_{12} & +x_{22} & +x_{32} & =1 \\
 & x_{13} & +x_{23} & +x_{33} & =1 \\
 & x_{11} & ,\ldots & x_{33} & \geq 0 \\
 & x_{11} & ,\ldots & x_{33} & \in \{0,1\}
\end{array}
$$

Algebraic Model
"Integer Program"

LP-Relaxation

Freie Universität Berlin

$$
\begin{array}{llllr}
\min & 3x_{11} & +3x_{12} & +4x_{13} & \\
 & +3x_{21} & +7x_{22} & +6x_{23} & \\
 & +7x_{31} & +8x_{32} & +10x_{33} & \\
\text{s.t.} & x_{11} & +x_{12} & +x_{13} & =1 \\
 & x_{21} & +x_{22} & +x_{23} & =1 \\
 & x_{31} & +x_{32} & +x_{33} & =1 \\
 & x_{11} & +x_{21} & +x_{31} & =1 \\
 & x_{12} & +x_{22} & +x_{32} & =1 \\
 & x_{13} & +x_{23} & +x_{33} & =1 \\
 & x_{11} & ,\ldots & x_{33} & \geq 0 \\
 & x_{11} & ,\ldots & x_{33} & \in \{0,1\}
\end{array}
$$

Integer Program

$$
\begin{array}{llllr}
\min & 3x_{11} & +3x_{12} & +4x_{13} & \\
 & +3x_{21} & +7x_{22} & +6x_{23} & \\
 & +7x_{31} & +8x_{32} & +10x_{33} & \\
\text{s.t.} & x_{11} & +x_{12} & +x_{13} & =1 \\
 & x_{21} & +x_{22} & +x_{23} & =1 \\
 & x_{31} & +x_{32} & +x_{33} & =1 \\
 & x_{11} & +x_{21} & +x_{31} & =1 \\
 & x_{12} & +x_{22} & +x_{32} & =1 \\
 & x_{13} & +x_{23} & +x_{33} & =1 \\
 & x_{11} & ,\ldots & x_{33} & \geq 0 \\
 & x_{11} & ,\ldots & x_{33} & \leq 1
\end{array}
$$

Linear Program
"LP-Relaxation" (here: integer)

LP-Relaxation

$$
\begin{array}{llllll}
\min & 3x_{11} & +3x_{12} & +4x_{13} & \\
 & +3x_{21} & +7x_{22} & +6x_{23} & \\
 & +7x_{31} & +8x_{32} & +10x_{33} & \\
\text{s.t.} & x_{11} & +x_{12} & +x_{13} & =1 \\
 & x_{21} & +x_{22} & +x_{23} & =1 \\
 & x_{31} & +x_{32} & +x_{33} & =1 \\
 & x_{11} & +x_{21} & +x_{31} & =1 \\
 & x_{12} & +x_{22} & +x_{32} & =1 \\
 & x_{13} & +x_{23} & +x_{33} & =1 \\
 & x_{11} & ,\ldots & x_{33} & \geq 0 \\
 & x_{11} & ,\ldots & x_{33} & \leq 1
\end{array}
$$

Linear Program
"LP-Relaxation"

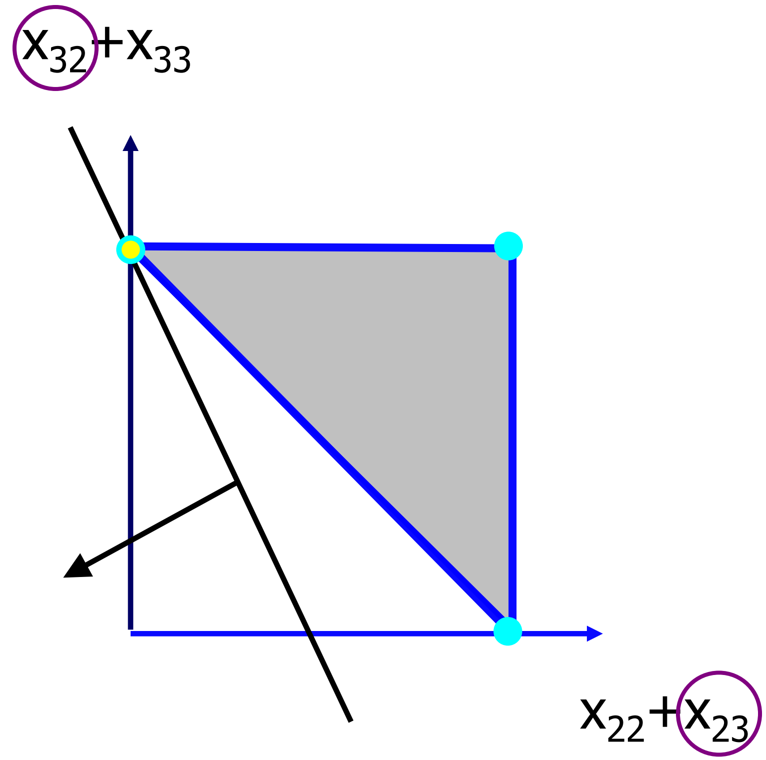
$$
\begin{array}{llllll}
\min & 3(1-x_{12}-x_{13}) & +3x_{12} & +4x_{13} & \\
 & +3x_{21} & +7x_{22} & +6x_{23} & \\
 & +7x_{31} & +8x_{32} & +10x_{33} & \\
\text{s.t.} & \cancel{1} & \cancel{-x_{12}} & \cancel{-x_{13}} & \cancel{=x_{11}} \\
 & x_{21} & +x_{22} & +x_{23} & =1 \\
 & x_{31} & +x_{32} & +x_{33} & =1 \\
 & 1-x_{12}-x_{13} & +x_{21} & +x_{31} & =1 \\
 & x_{12} & +x_{22} & +x_{32} & =1 \\
 & x_{13} & +x_{23} & +x_{33} & =1 \\
 & 1-x_{12}-x_{13} & ,\ldots & x_{33} & \geq 0 \\
 & \cancel{1-x_{12}-x_{13}} & ,\ldots & x_{33} & \leq 1
\end{array}
$$

Eliminate x_{11}

Eliminate x_{12}, x_{13}, x_{21}, x_{31}

LP-Relaxation

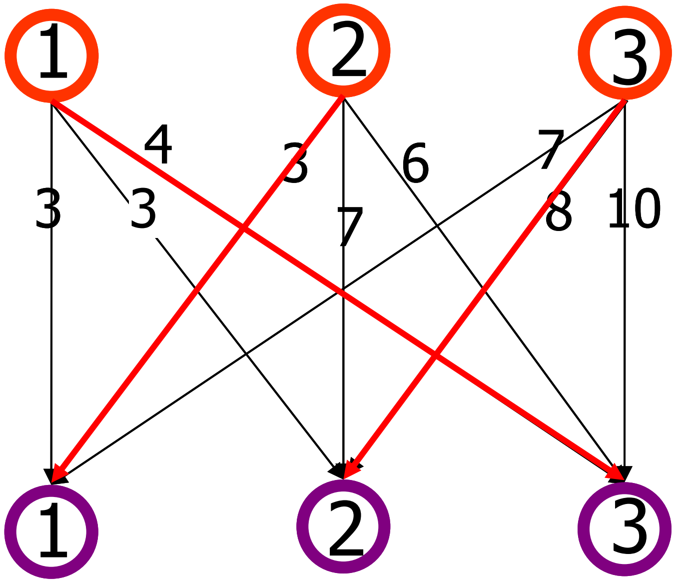
$$
\begin{array}{llllll}
\min & 4x_{22} & +2x_{23} & +1x_{32} & +2x_{33} & +14 \\
\text{s.t.} & x_{22} & +x_{23} & +x_{32} & +x_{33} & \geq 1 \\
 & x_{22} & +x_{23} & & & \leq 1 \\
 & & & x_{32} & +x_{33} & \leq 1 \\
 & x_{22}, & x_{23}, & x_{32}, & x_{33} & \geq 0
\end{array}
$$



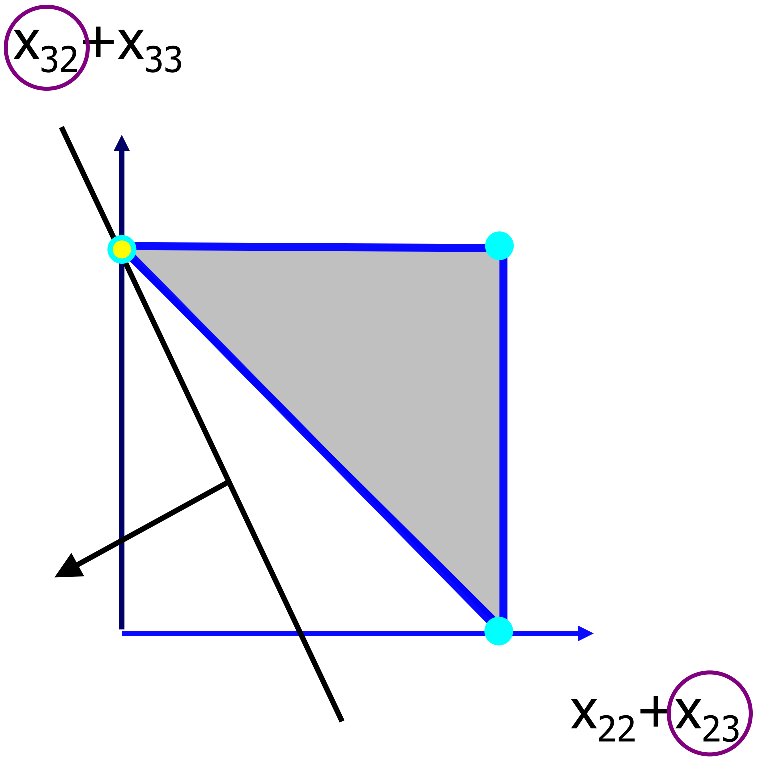
Linear Program
"LP-Relaxation"

"Polyhedron"

Mathematical Models



$x_{32}=1$
$x_{21}=1$
$x_{13}=1$



"Polyhedron"

Linear Programming

Min x1 + 2x2

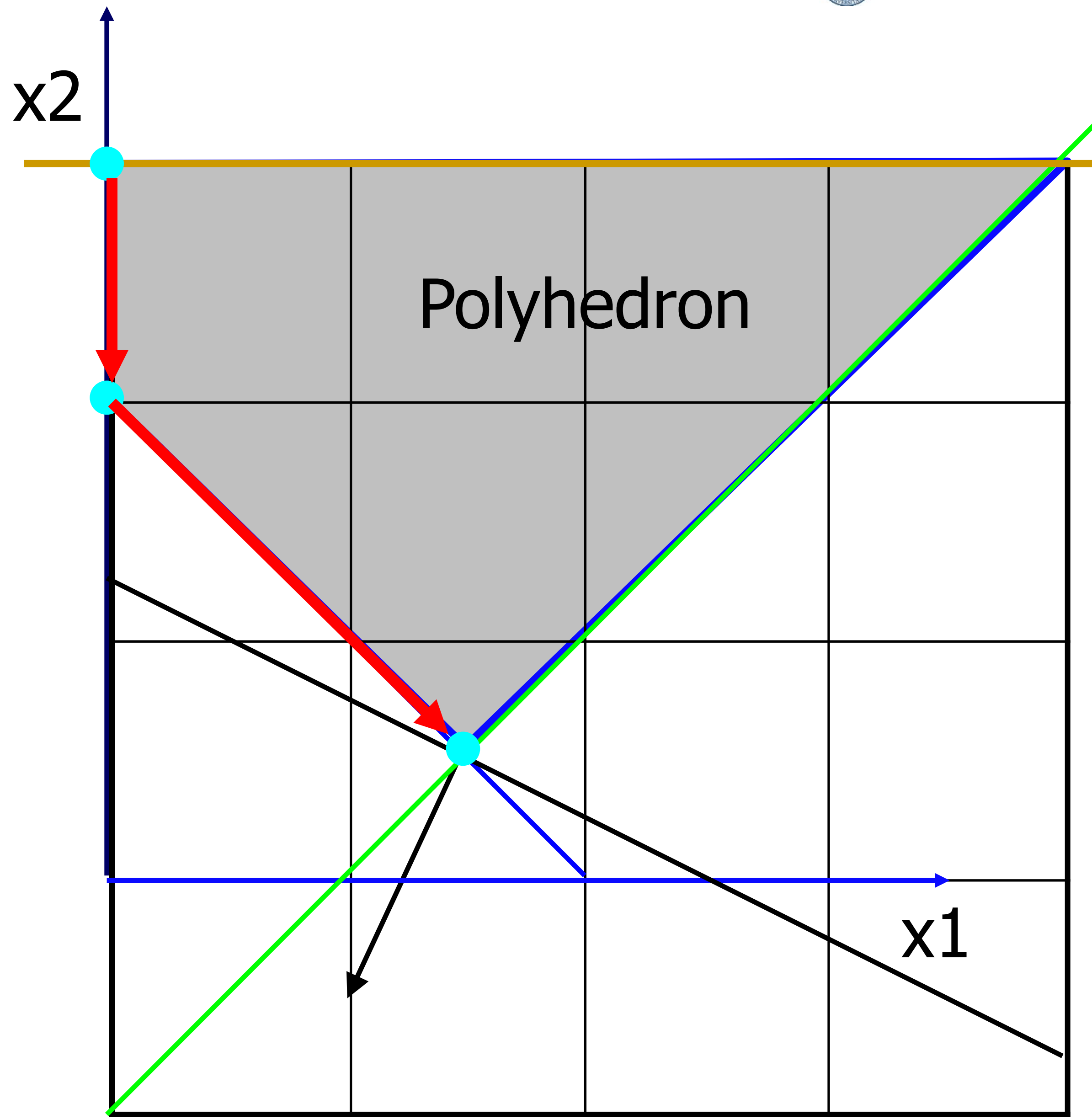
x1 + x2 ≥ 2

x1 - x2 ≤ 1

x2 ≤ 3

x1 ≥ 0

x2 ≥ 0



Simplex Algorithm

Linear Programming 1987-2000

(Bixby, Solving Real-World Linear Programs: A Decade and More of Progress. Oper. Res. 50(1) 3-15, 2002)

Hardware

Old Computer	*New Computer*	*Speedup*
Sun 3/50	Pentium 4, 1.7 GHz	800
Sun 3/50	Compaq Server ES 40, 667 MHz	900
Intel 386, 25 MHz	Compaq Server ES 40, 667 MHz	400
IBM 3090/108S	Compaq Server ES 40, 667 MHz	45

Software

Old Code	*New Code*	*Estimated Speedup*
XMP	Cplex 1.0	4.7
Cplex 1.0	Cplex 5.0	22,0
Cplex 5.0	Cplex 7.1	3.7
XMP	Cplex 7.1	960

"A Model that might have taken a year to solve 10 years ago, can now solve in less than 10 seconds."

Linear Programming

Freie Universität Berlin

Min x1 + 2x2

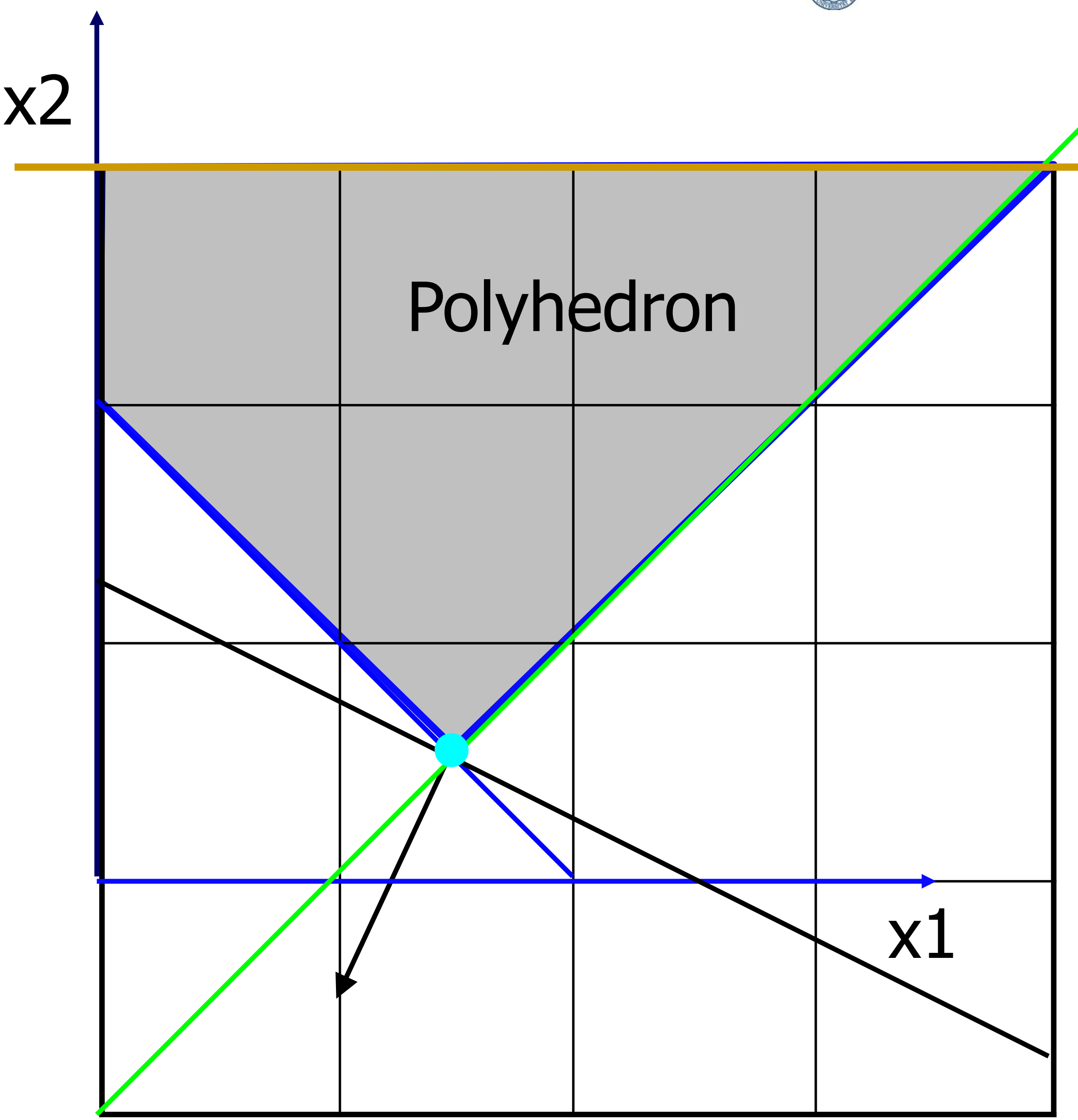
x1 + x2 ≥ 2

x1 - x2 ≤ 1

x2 ≤ 3

x1 ≥ 0

x2 ≥ 0



Simplex Algorithm

Integer Programming

Min x1 + 2x2

x1 + x2 ≥ 2

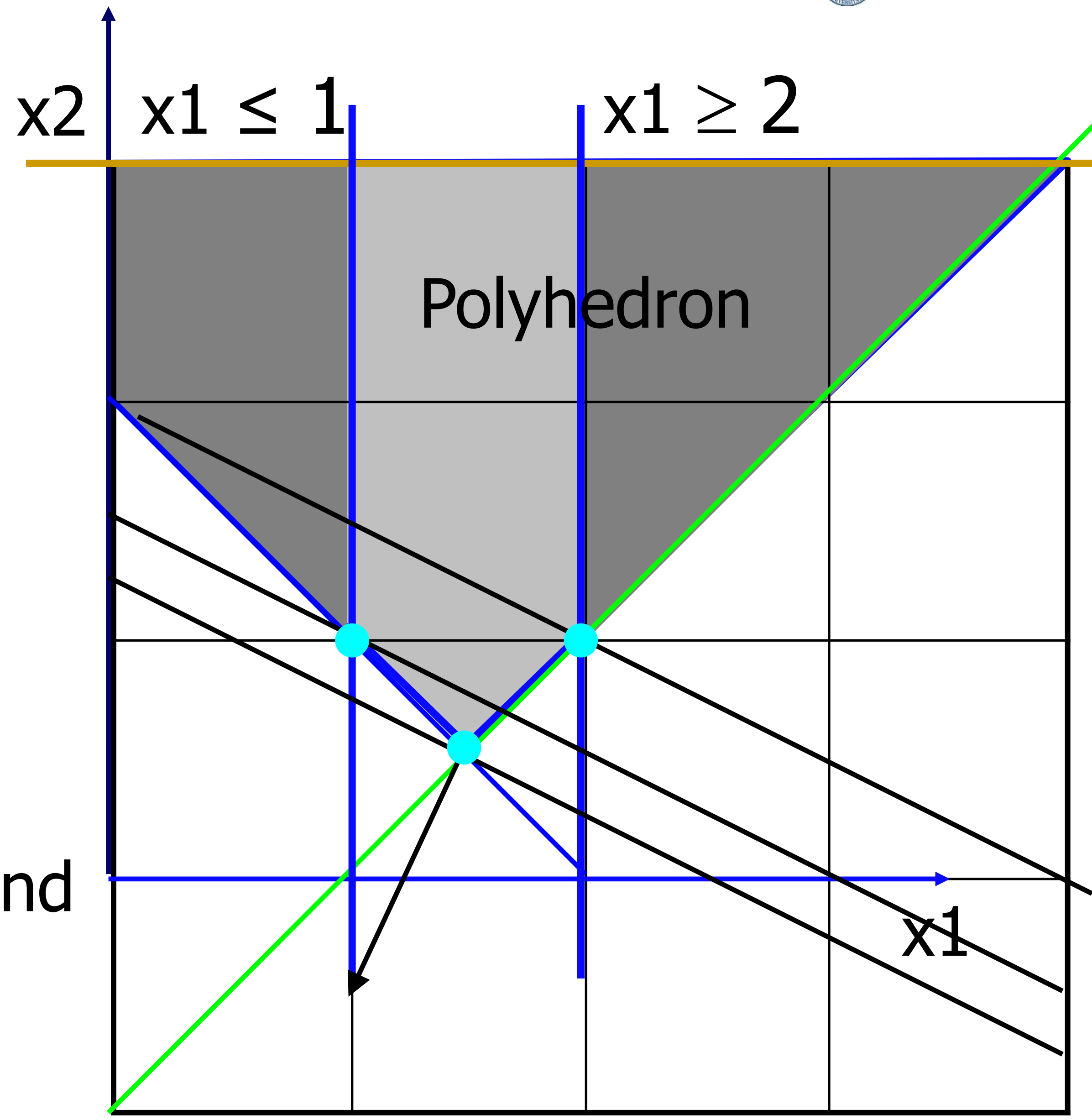
x1 - x2 ≤ 1

x2 ≤ 3

x1 ≥ 0

x2 ≥ 0

x1, x2 integer



Branch-and-Bound

Integer Programming

Min x1 + 2x2

x1 + x2 ≥ 2

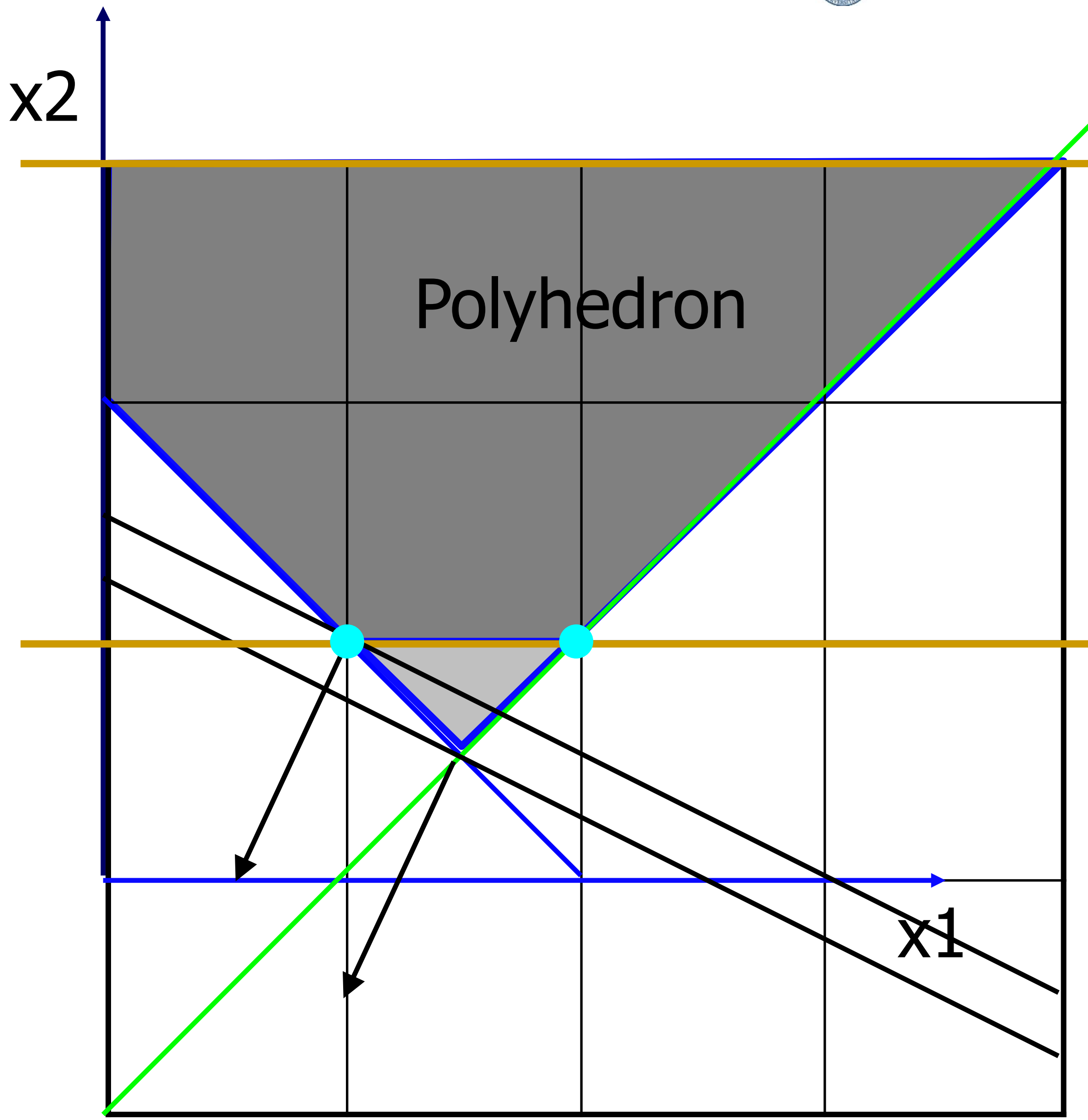
x1 - x2 ≤ 1

x2 ≤ 3

x1 ≥ 0

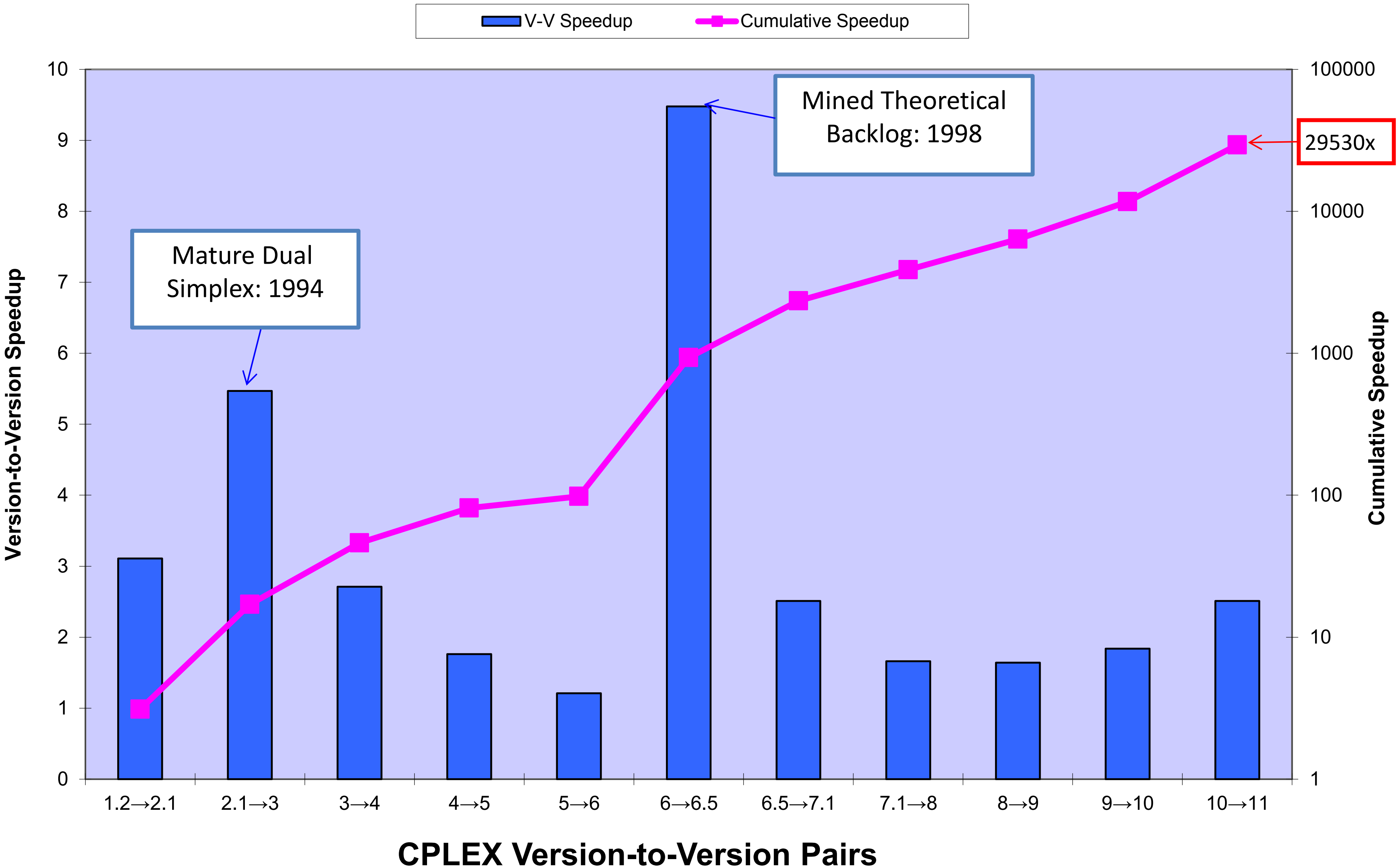
x2 ≥ 0

Cutting Planes



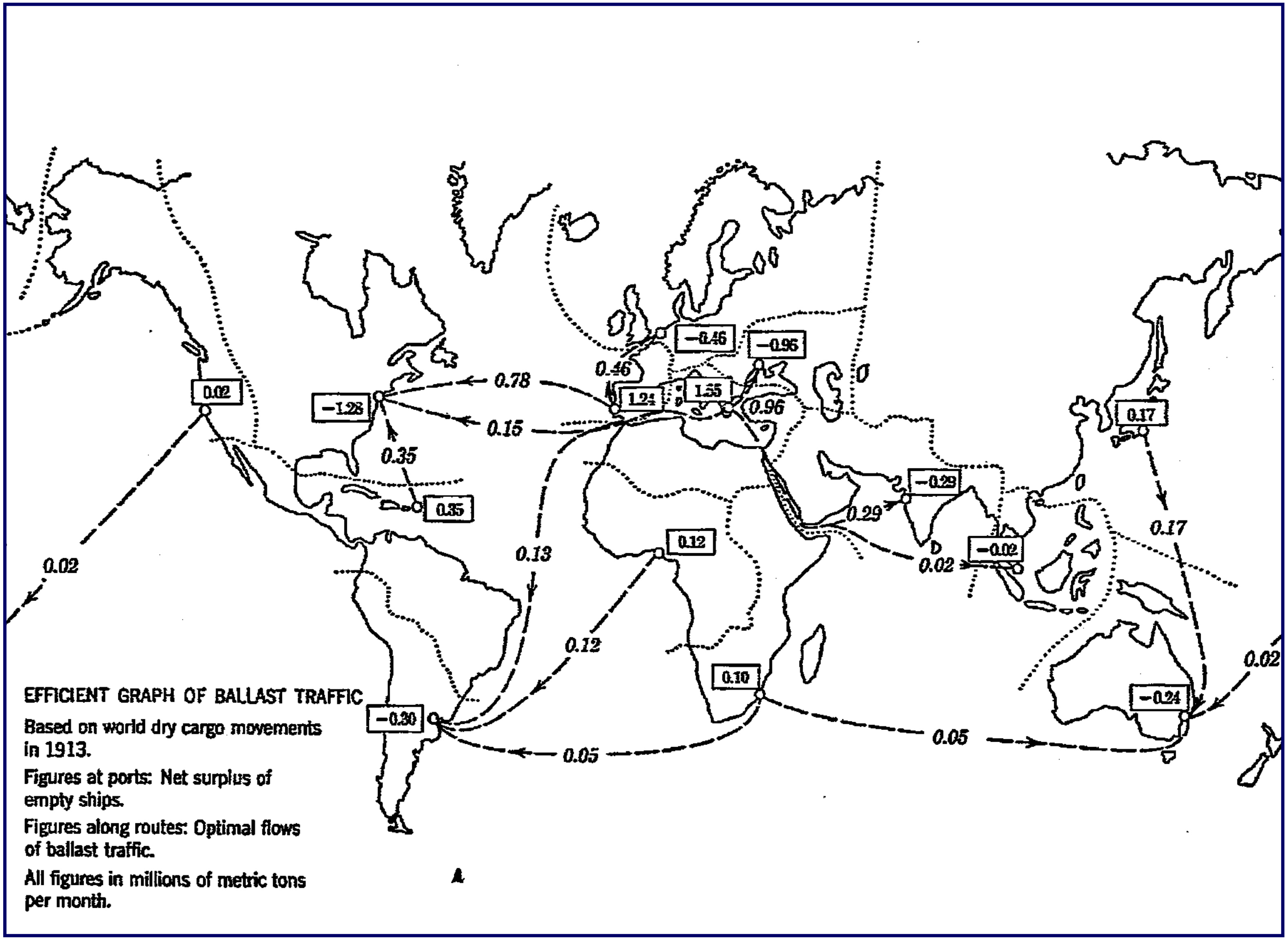
MIP-Speedup 1991-2010

(Bixby, Lecture on Mixed-Integer Programming, TU Berlin, 20.01.2010)



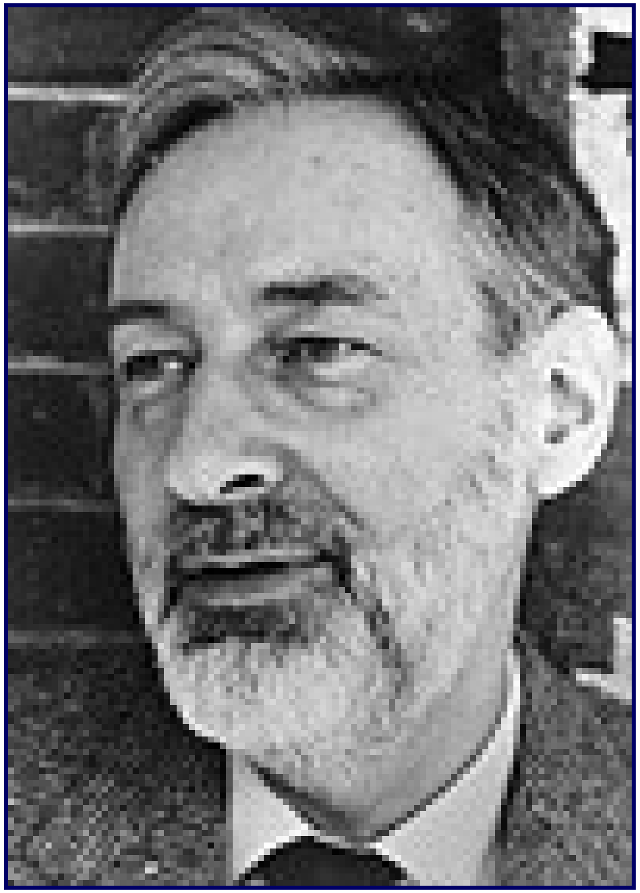
Sea Freight

(Koopmans [1965], 7 sources, 7 sinks, all sea links)



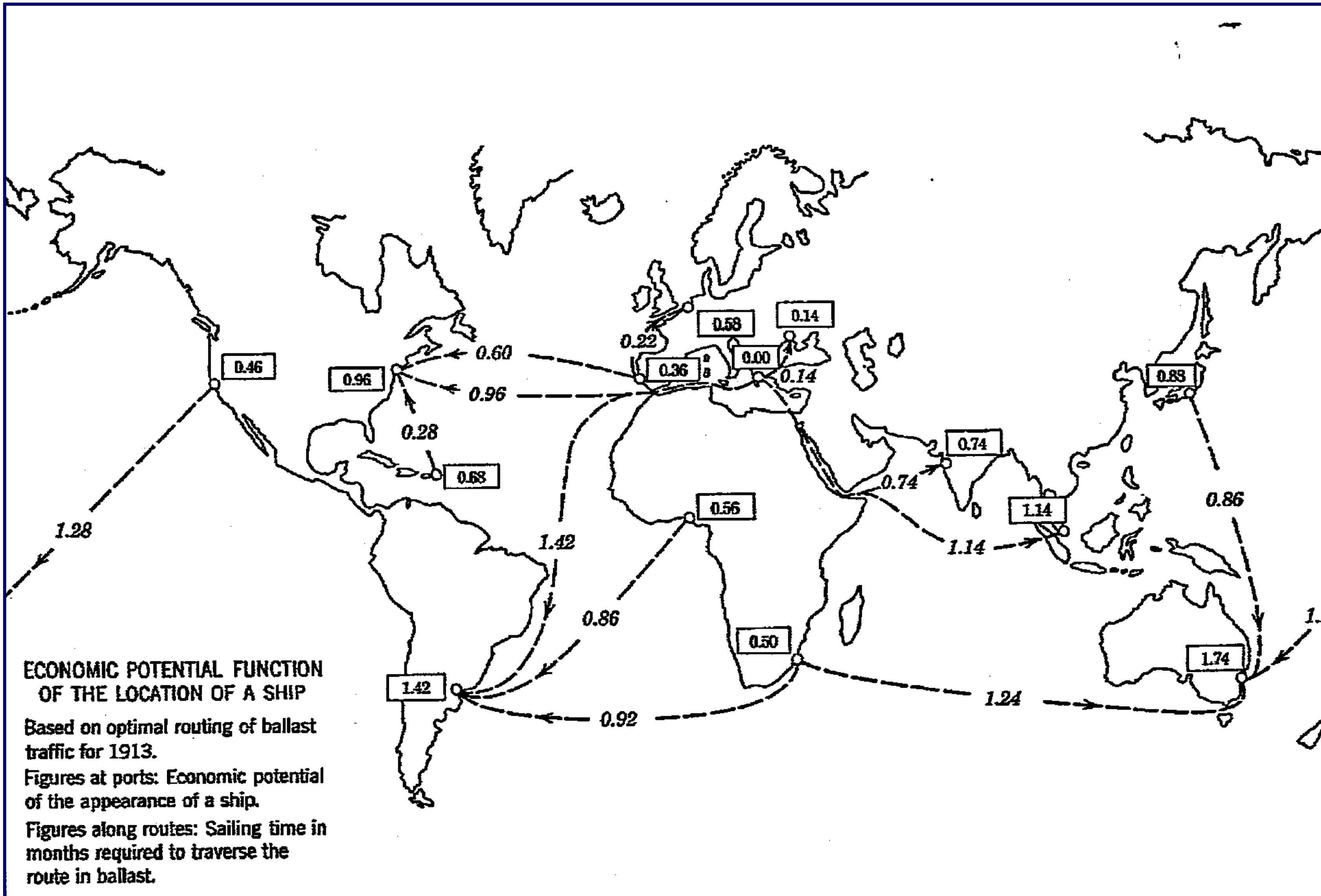
Optimal Allocation of Scarce Resources
(Nobel Price in Economics 1975)

Leonid V. Kantorovich

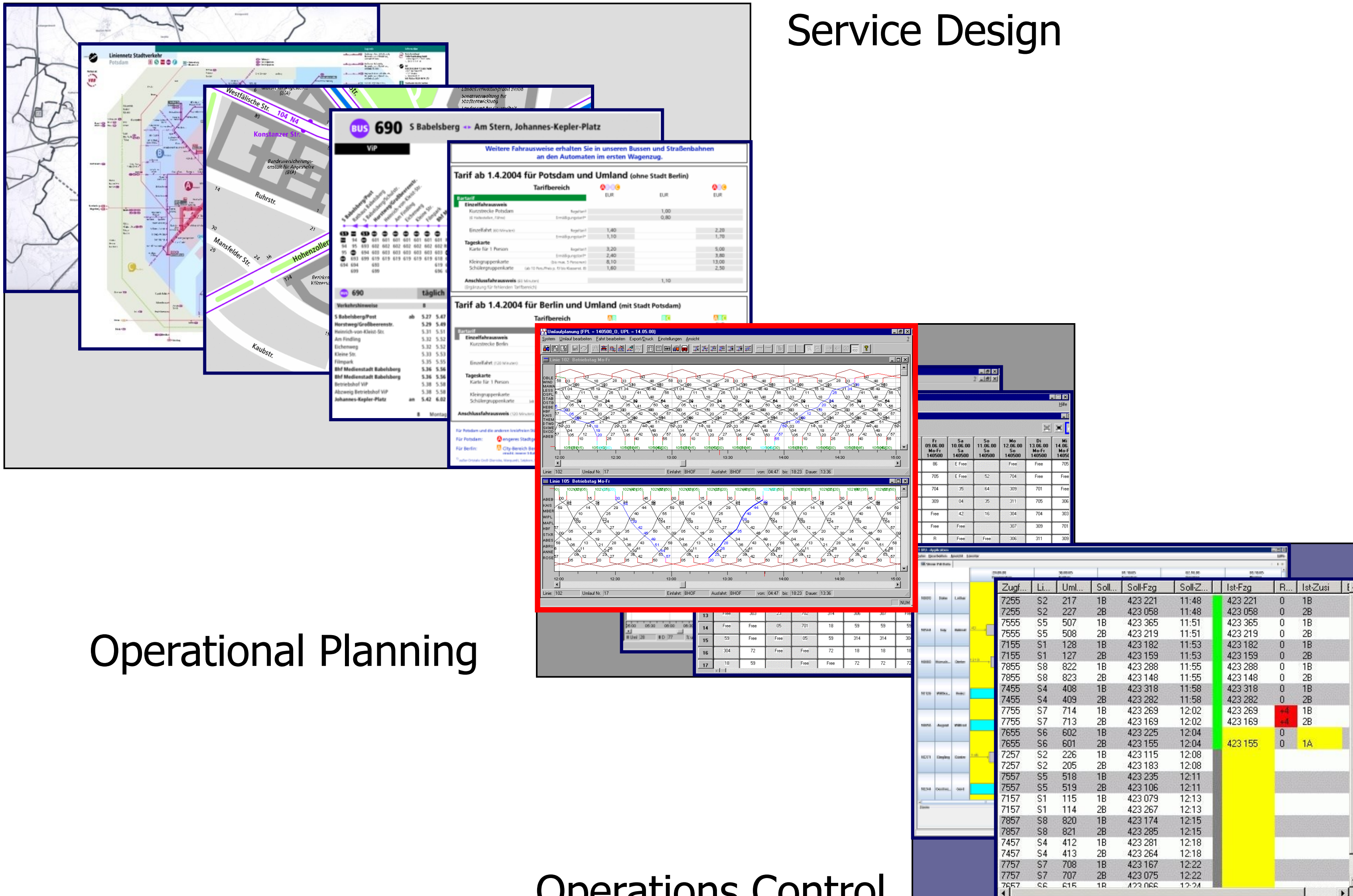


Tjalling C. Koopmans

LP Solution via Shadow Prices



Planning Problems in Public Transit

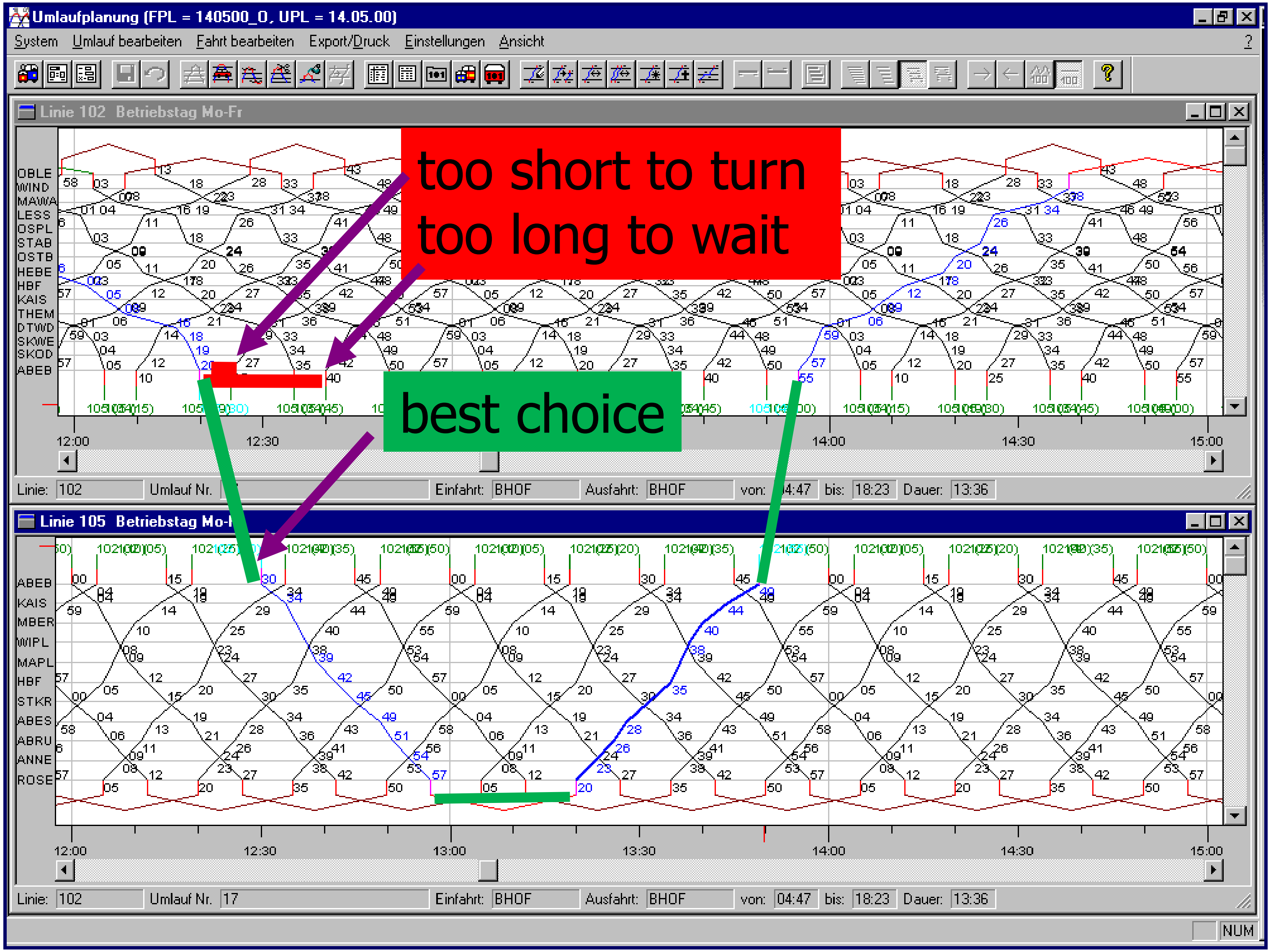


Service Design

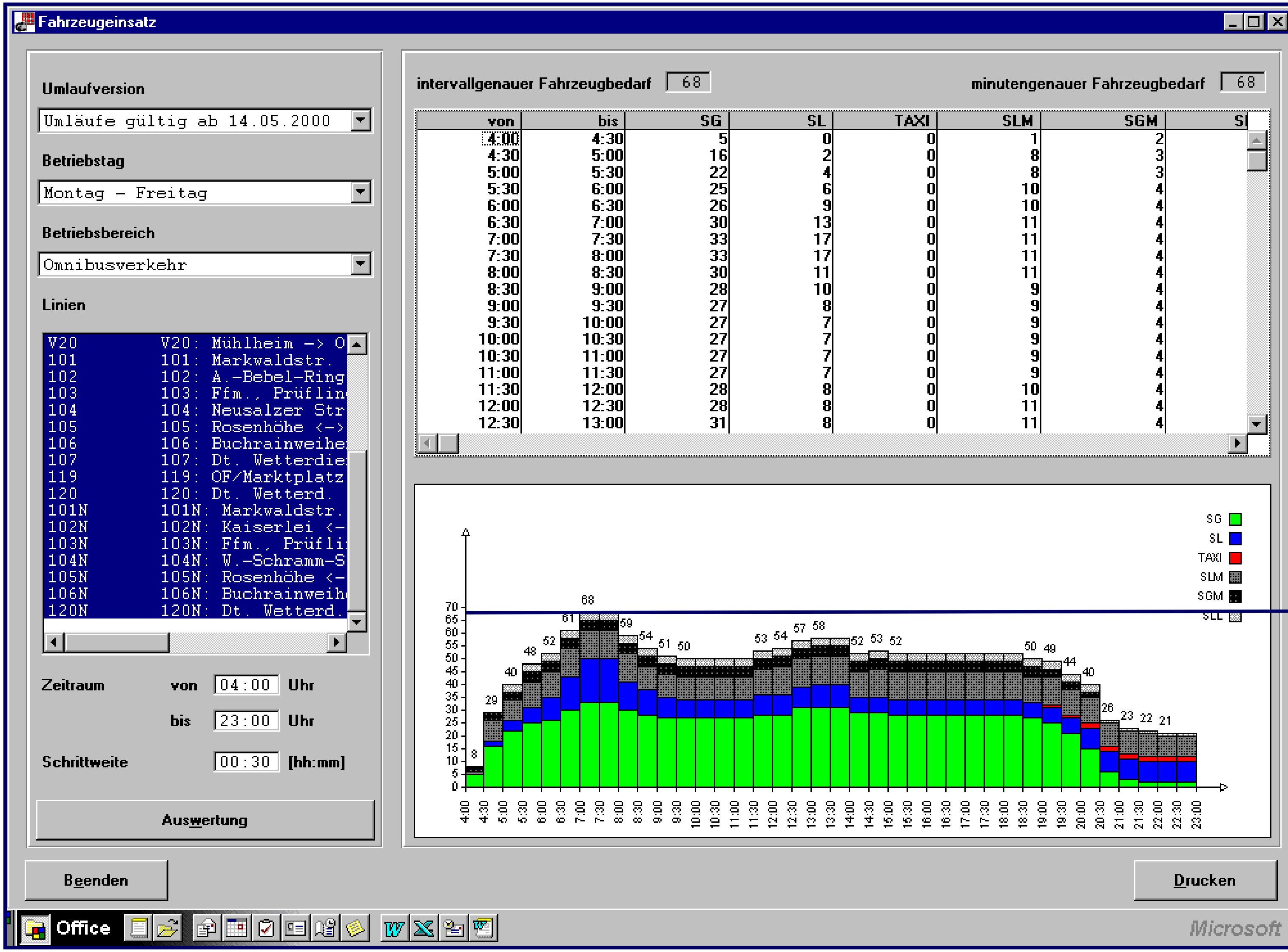
Operational Planning

Operations Control

Vehicle Scheduling



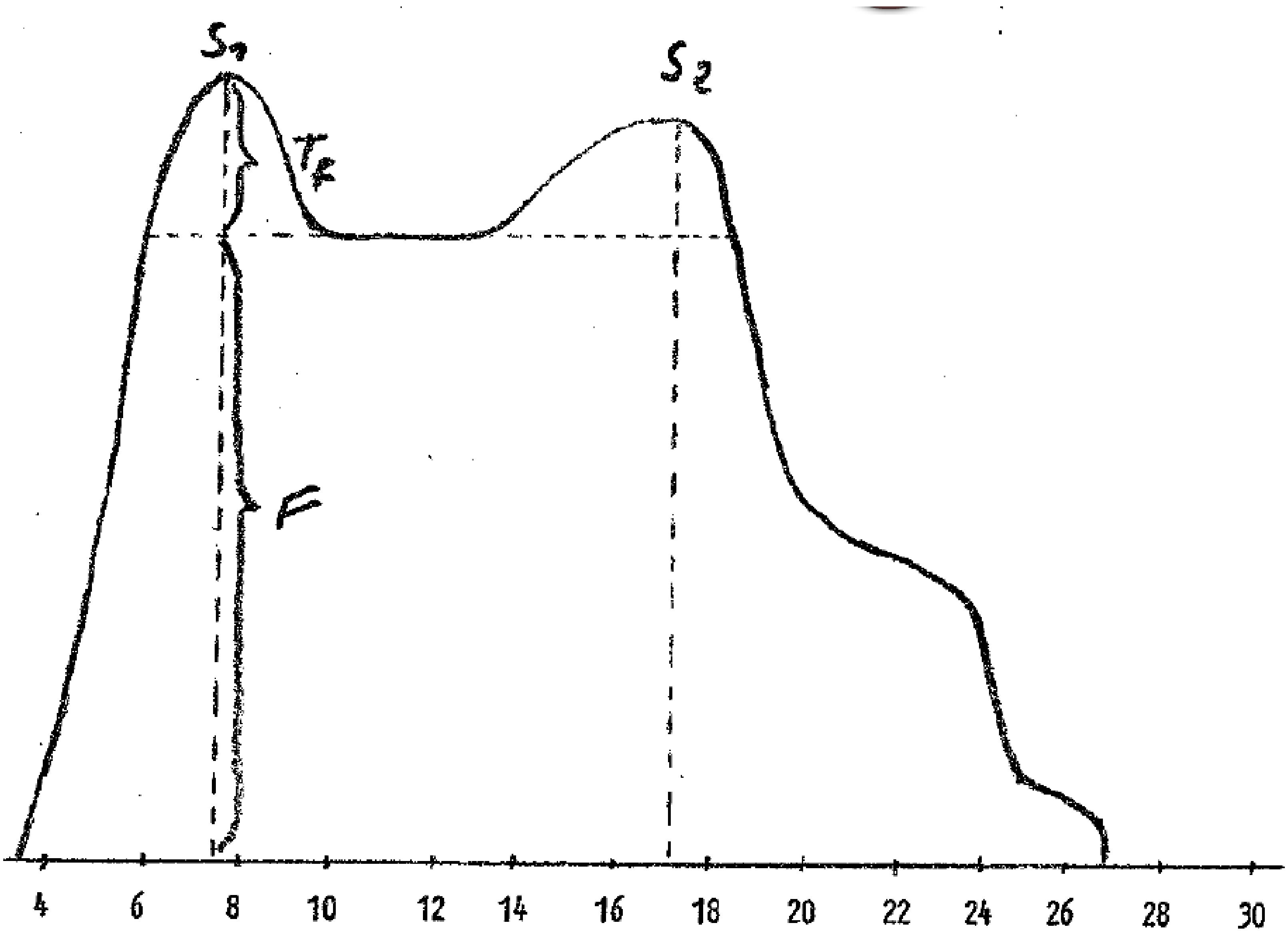
"Camel Curve"



von	bis	SG	SL	TAXI	SLM	SGM	S
4:00	4:30	5	0	0	1	2	
4:30	5:00	16	2	0	8	3	
5:00	5:30	22	4	0	8	3	
5:30	6:00	25	6	0	10	4	
6:00	6:30	26	9	0	10	4	
6:30	7:00	30	13	0	11	4	
7:00	7:30	33	17	0	11	4	
7:30	8:00	33	17	0	11	4	
8:00	8:30	30	11	0	11	4	
8:30	9:00	28	10	0	9	4	
9:00	9:30	27	8	0	9	4	
9:30	10:00	27	7	0	9	4	
10:00	10:30	27	7	0	9	4	
10:30	11:00	27	7	0	9	4	
11:00	11:30	27	7	0	9	4	
11:30	12:00	28	8	0	10	4	
12:00	12:30	28	8	0	11	4	
12:30	13:00	31	8	0	11	4	

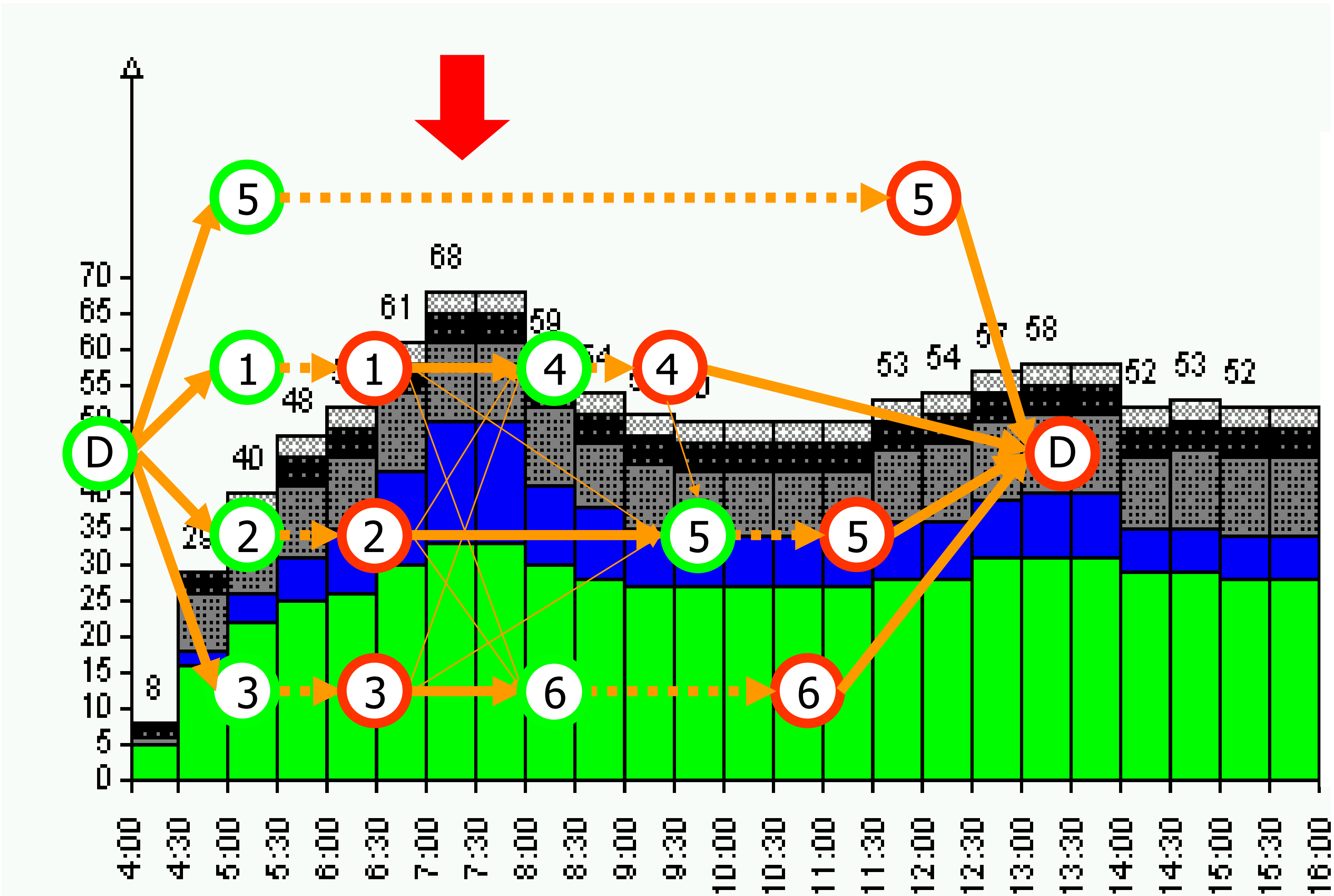
68

"Camel Curve"
(Ario, Böhring & Mojsilovic [1980])

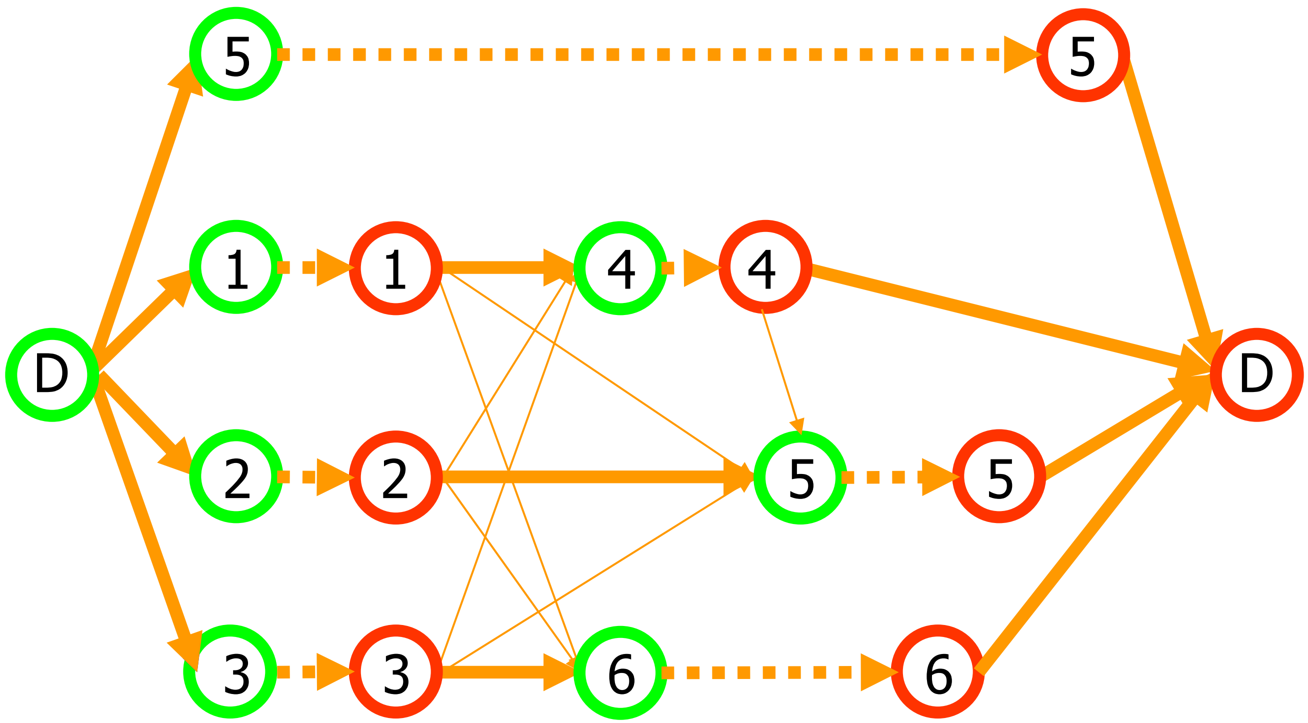


Assignment Approach
(Single Depot Vehicle Scheduling)

Freie Universität Berlin

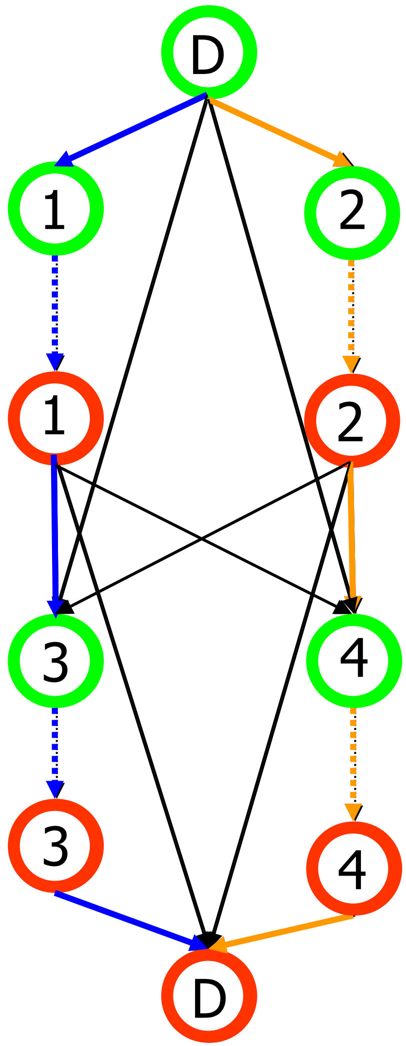


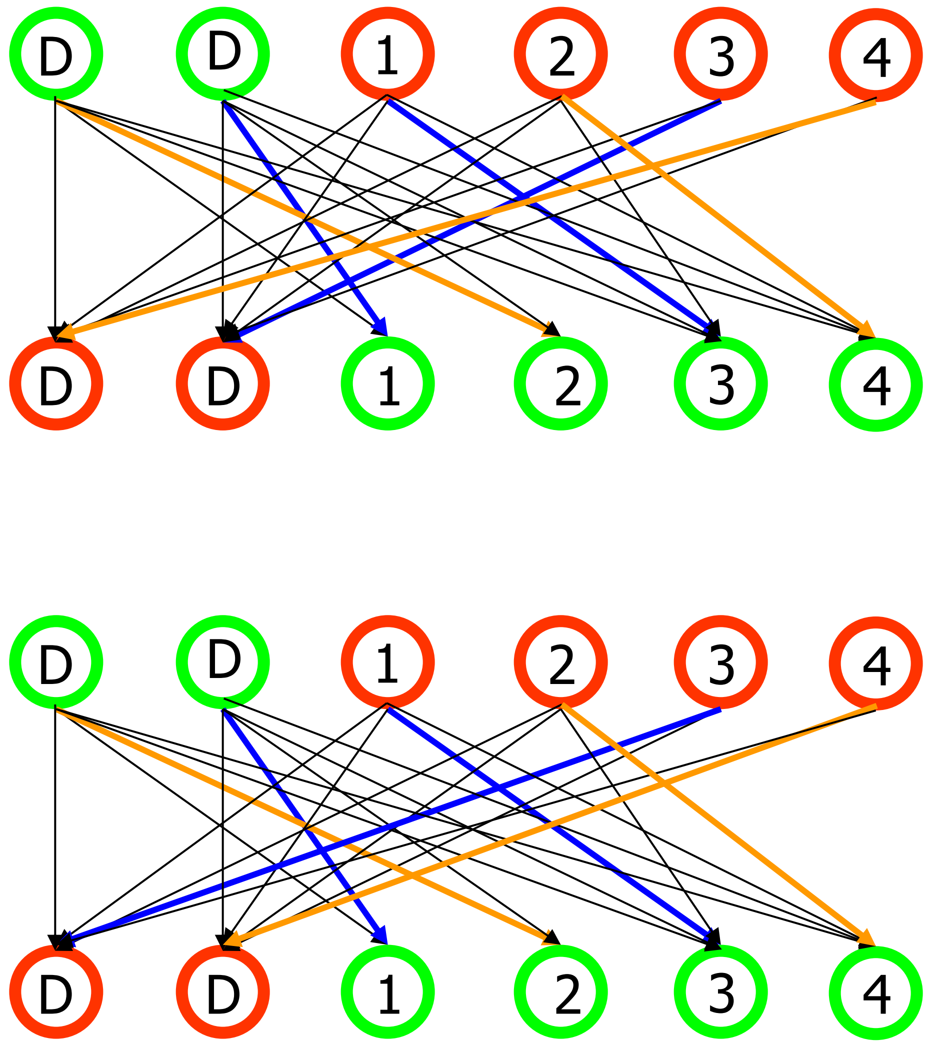
Assignment Approach
(Single Depot Vehicle Scheduling)



Single Depot Vehicle Scheduling

(Assignment Approach)

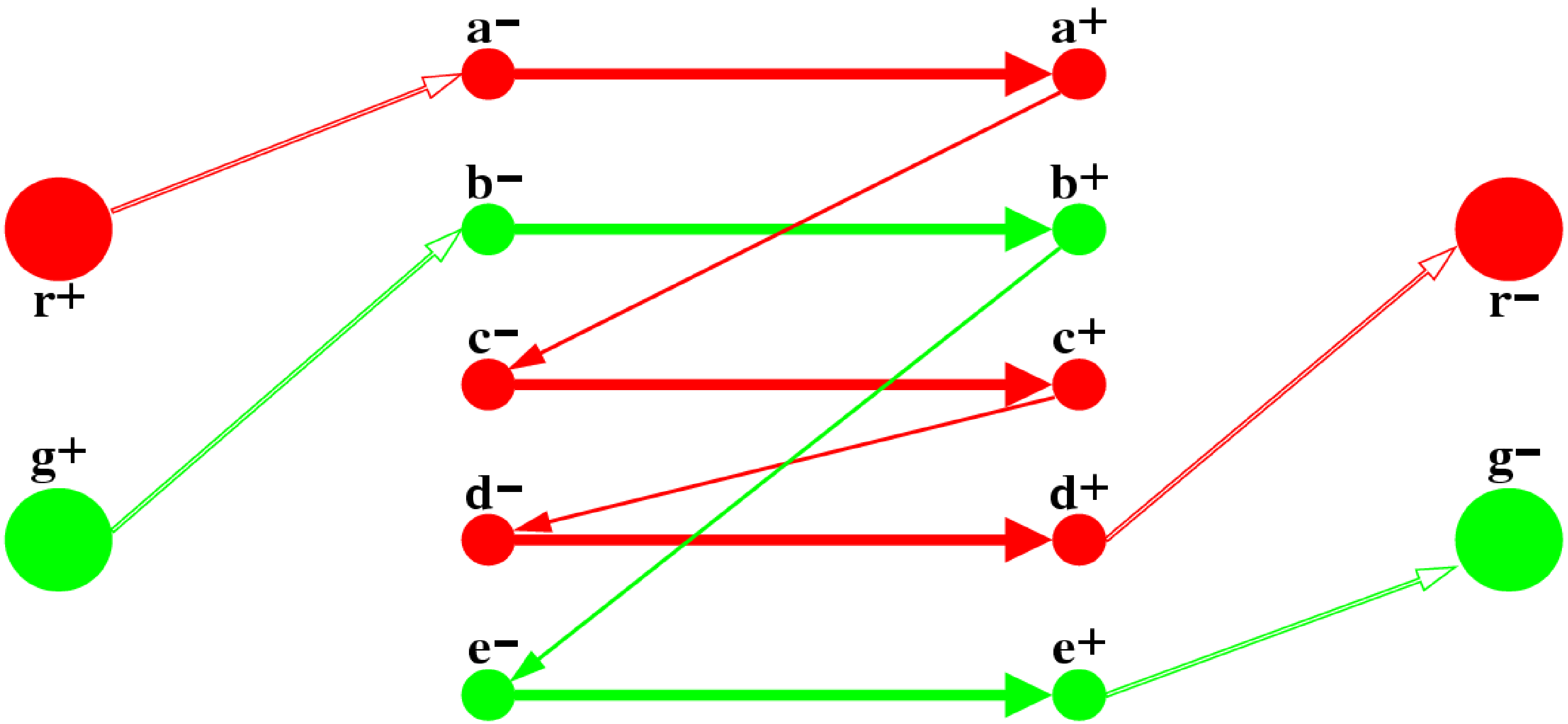




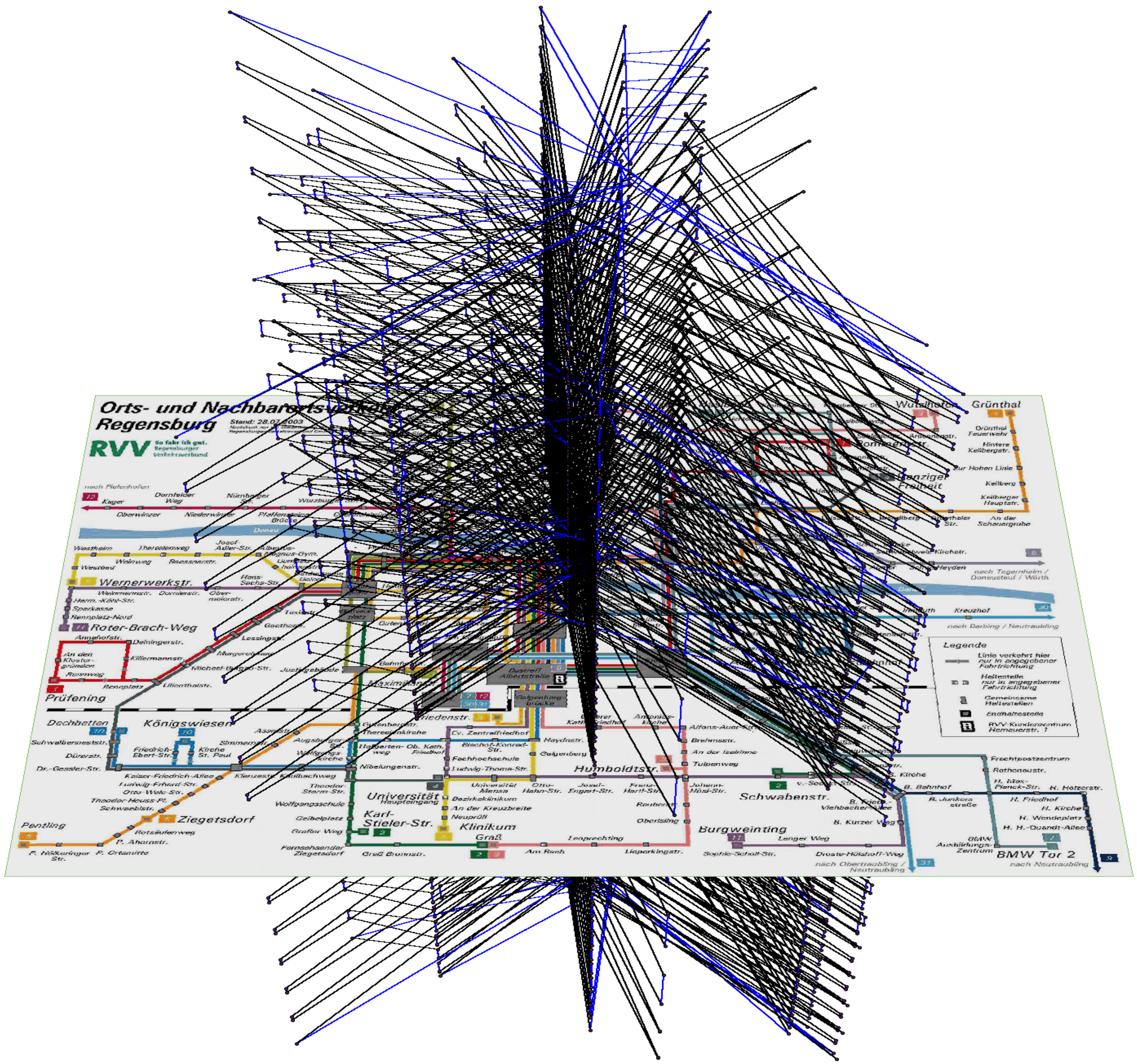
Vehicle Scheduling Problem

- Input
  Timetabled and deadhead trips
  Vehicle types and depot capacities
  Vehicle costs (fixed and variable)
- Output
  Vehicle rotations
- Problem
  Compute rotations to cover all timetabled trips
- Goals
  Minimize number of vehicles
  Minimize operation costs
  Minimize line hopping etc.

Multicommodity Flow Model



Vehicle Scheduling (VS-OPT)



Integer Programming Model
(Multi-Commodity Flow Problem)

$$\min \quad \sum_{d} \sum_{ij} c_{ij}^{d} x_{ij}^{d}$$

$$\begin{array}{rclll}
\sum_{i} x_{ij}^{d} - \sum_{k} x_{jk}^{d} & = & 0 & \forall j, d & \text{Vehicle Flow} \\
\sum_{d} \sum_{i} x_{ij}^{d} - \sum_{d} \sum_{k} x_{jk}^{d} & = & 0 & \forall j & \text{Aggregate Flow} \\
\sum_{d} x_{ij}^{d} & = & 1 & \forall j & \text{Trips} \\
\sum_{j} x_{dj}^{d} & \leq & \kappa^{d} & \forall d & \text{Capacities} \\
x_{ij}^{d} & \in & \{0,1\} & \forall ij, d & \text{Integrality}
\end{array}$$

Theoretical Results

Observation: The LP relaxation of the Multicommodity Flow Problem is in general not integer.

Theorem: The Multicommodity Flow Problem is NP-hard.

Theorem (Tardos et. al.): There are pseudo-polynomial time approximation algorithms to solve the LP-relaxation of Multicommodity Flow Problems which are faster than general LP methods.

Integer Programming Model
(Multi-Commodity Flow Problem)

Freie Universität Berlin

$$\min \quad \sum_d \sum_{ij} c_{ij}^d x_{ij}^d$$

$$\boxed{\sum_i x_{ij}^d - \sum_k x_{jk}^d = 0 \quad \forall j,d \quad \text{Vehicle Flow}}$$

$$\sum_d \sum_i x_{ij}^d - \sum_d \sum_k x_{jk}^d = 0 \quad \forall j \quad \text{Aggregate Flow}$$

$$\sum_d x_{ij}^d = 1 \quad \forall j \quad \text{Trips}$$

$$\sum_j x_{dj}^d \leq \kappa^d \quad \forall d \quad \text{Capacities}$$

$$x_{ij}^d \in \{0,1\} \quad \forall ij,d \quad \text{Integrality}$$

Lagrangean Relaxation
(Subproblem is a Min Cost Flow Problem)

$$\max_{\pi} \ \min \ \sum_{d}\sum_{ij} c_{ij}^{d} x_{ij}^{d} - \sum_{j,d} \pi_j^d \left(\sum_i x_{ij}^d - \sum_i x_{ji}^d \right) \qquad \text{Lagrangean}$$

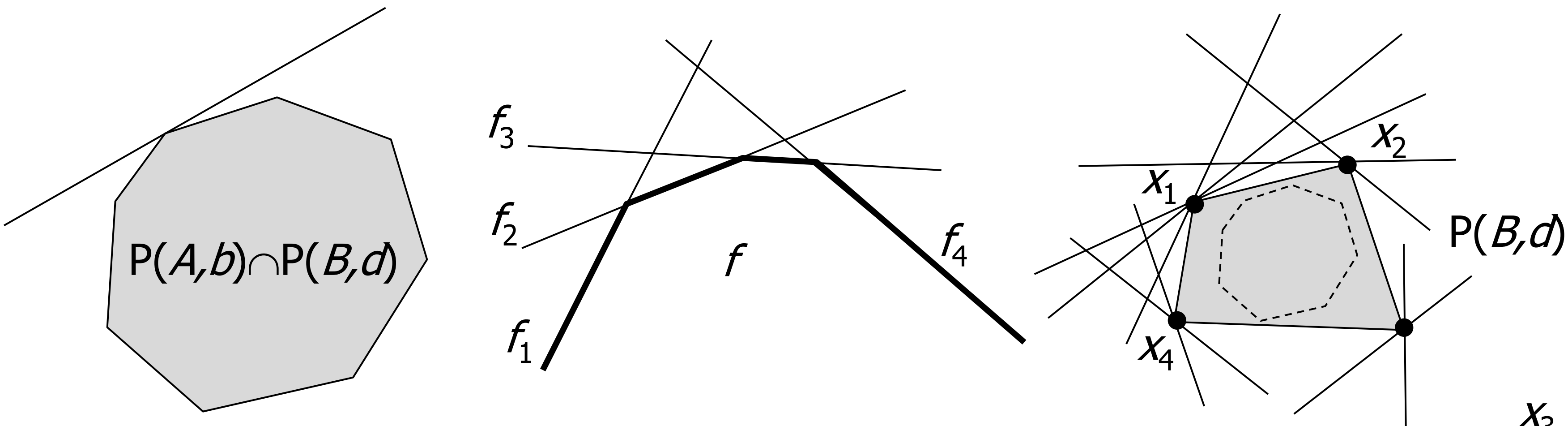
$$\begin{array}{lcccc}
\sum_d \sum_i x_{ij}^d - \sum_d \sum_i x_{ji}^d & = & 0 & \forall j & \text{Agg. Flow} \\
\sum_d x_{ij}^d & = & 1 & \forall j & \text{Trips} \\
\sum_j x_{dj}^d & \leq & \kappa^d & \forall d & \text{Capacities} \\
x_{ij}^d & \in & \{0,1\} & \forall ij,d & \text{Binary}
\end{array}$$

Subproblem: Min-Cost Flow

Lagrangean Relaxation

$$\begin{array}{llll} \min & c^T x & & \\ & Ax & = & b \\ & Bx & = & d \\ & x & \geq & 0 \end{array} \quad = \quad \begin{array}{llllll} \max\limits_{\lambda} & \min & c^T x & + & \lambda(b - Ax) \\ & & Bx & = & d \\ & & x & \geq & 0 \end{array}$$

$$= \max_{\lambda} \quad f(\lambda) \quad = \quad \max_{\lambda} \quad \min_{i} \quad c^T x_i \quad + \quad \lambda(b - Ax_i)$$



Integer Programming Model
(Multi-Commodity Flow Problem)

Freie Universität Berlin

$$
\begin{array}{llllll}
\min & \sum_d \sum_{ij} c_{ij}^d x_{ij}^d & & & & \\
& \boxed{\sum_i x_{ij}^d - \sum_k x_{jk}^d} & = & 0 & \forall j,d & \text{Vehicle Flow} \\
& \sum_d \sum_i x_{ij}^d - \sum_d \sum_k x_{jk}^d & = & 0 & \forall j & \text{Aggregate Flow} \\
& \sum_d x_{ij}^d & = & 1 & \forall j & \text{Trips} \\
& \sum_j x_{dj}^d & \le & \kappa^d & \forall d & \text{Capacities} \\
& x_{ij}^d & \in & \{0,1\} & \forall ij,d & \text{Integrality}
\end{array}
$$

Lagrangean Relaxation
(Subproblem is a Min Cost Flow Problem)

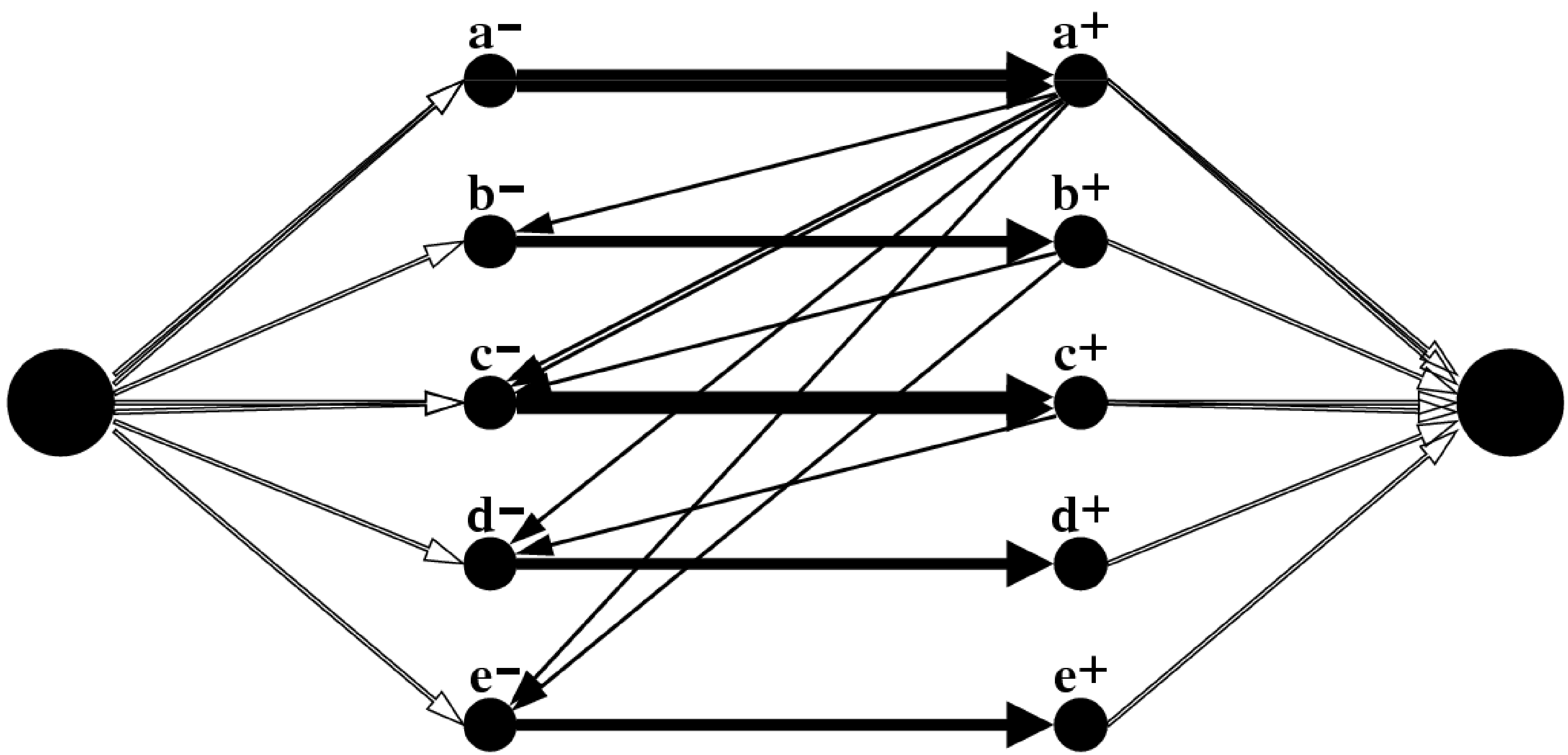
Freie Universität Berlin

$$\max_{\pi} \min \sum_{d} \sum_{ij} c_{ij}^{d} x_{ij}^{d} - \sum_{j,d} \pi_{j}^{d} \left(\sum_{i} x_{ij}^{d} - \sum_{i} x_{ji}^{d} \right) \qquad \text{Lagrangean}$$

$$\begin{array}{llll}
\sum_{d} \sum_{i} x_{ij}^{d} - \sum_{d} \sum_{i} x_{ji}^{d} = 0 & \forall j & \text{Agg. Flow} \\
\sum_{d} x_{ij}^{d} = 1 & \forall j & \text{Trips} \\
\sum_{j} x_{dj}^{d} \le \kappa^{d} & \forall d & \text{Capacities} \\
x_{ij}^{d} \in \{0,1\} & \forall ij, d & \text{Binary}
\end{array}$$

Subproblem: Min-Cost Flow

Lagrangean Relaxation
(Subproblem is a Min Cost Flow Problem)



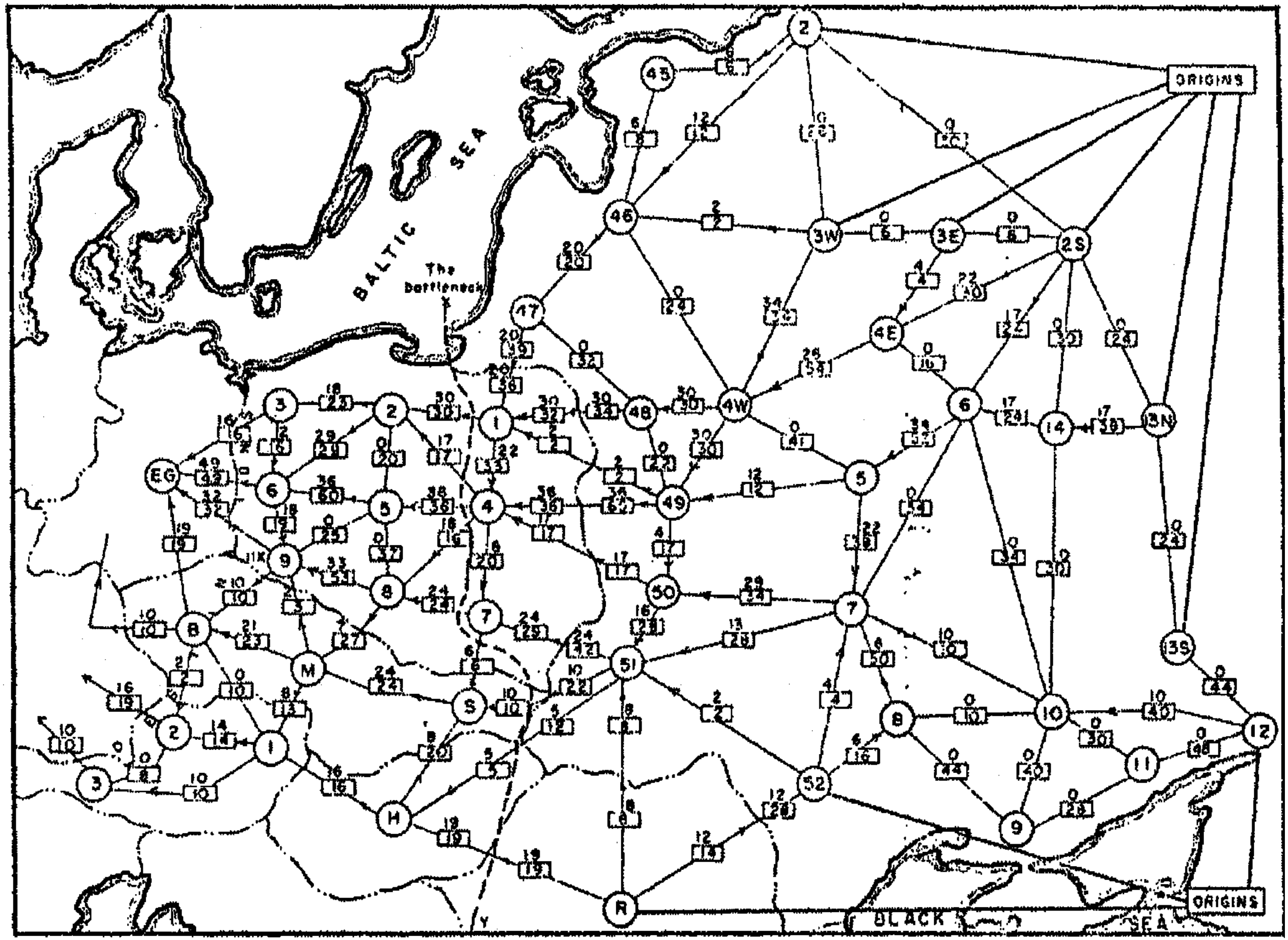
Network Flows

Delbert Ray Fulkerson

Lester Randolph Ford Jr.

Military Logistics

(Ford & Fulkerson [1955], Schrijver [2002])



MCF

Zuse Institute Berlin - Department Optimization - Software - MCF - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe http://www.zib.de/Optimization/Software/Mcf/

Zuse Institute Berlin - Department O...

ZIB Home Contact Publications Search Sitemap

General
About ZIB
Organization
Divisions
Maps / Directions

People
Staff
Fellows

Research
Research Units
Publications
Funded Projects
Software
Spin-offs

Services
Administration
Library
IT Services
Supercomputing
BRAIN
KOBV
Web Services

News
Press
Events
Positions

MCF - A network simplex implementation.

Current Release
Version 1.3

Author
Andreas Löbel

References
SPEC CPU2006: 429.mcf.
SPEC CPU2000: 181.mcf, a well investigated program.
The MCFClass Project by Antonio Frangioni

Documentation
README and README.changes
mcf.ps, mcf.pdf, or mcf.dvi

Supported Platforms
Gnu make (e.g., Linux/Unix/Cygwin)
MS Visual C++ 6.0 (Windows)

License Conditions
Free of charge for academic use (ZIB ACADEMIC LICENSE)
Commercial use requires an individual license agreement!

Download
Gnu make binary packages (tgz): Linux, SunOS, and Cygwin
MSVC++ 6.0 binary packages: tgz, zip, and self-extracting exe
Source packages: tgz, zip, and self-extracting exe

Sites for other min cost flow solvers
ZIB elib
Hans D. Mittelmann's site

Last Update: August 24, 2006
by Andreas Löbel

Fertig Jetzt: Bewölkt, 2 °C Sa: 7 °C So: 7 °C

SPEC CINT2006 Benchmarks - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe http://www.spec.org/cpu2006/CINT2006/

SPEC CINT2006 Benchmarks

Standard Performance Evaluation Corporation

home | benchmarks | results | contact | site map | site search | help

Results

- Published Results
- Results Search
- OSG Fair Use Policy

Information

- CPU2006
- Documentation
  - Documentation Overview
  - Run & Reporting Rules
  - Readme1st
- FAQ

Press and Publications

- V1.0 Release
- V1.1 Release
- Related Publications

Order Benchmarks

- Order CPU2006

Resources

- Site Map
- Site Search
- Site Index
- Glossary
- Performance Links

CINT2006 (Integer Component of SPEC CPU2006):

Benchmark	Language	Application Area	Brief Description
400.perlbench	C	Programming Language	Derived from Perl V5.8.7. The workload includes SpamAssassin, MHonArc (an email indexer), and specdiff (SPEC's tool that checks benchmark outputs).
401.bzip2	C	Compression	Julian Seward's bzip2 version 1.0.3, modified to do most work in memory, rather than doing I/O.
403.gcc	C	C Compiler	Based on gcc Version 3.2, generates code for Opteron.
429.mcf	C	Combinatorial Optimization	Vehicle scheduling. Uses a network simplex algorithm (which is also used in commercial products) to schedule public transport.
445.gobmk	C	Artificial Intelligence: Go	Plays the game of Go, a simply described but deeply complex game.
456.hmmer	C	Search Gene Sequence	Protein sequence analysis using profile hidden Markov models (profile HMMs)
458.sjeng	C	Artificial Intelligence: chess	A highly-ranked chess program that also plays several chess variants.
462.libquantum	C	Physics / Quantum Computing	Simulates a quantum computer, running Shor's polynomial-time factorization algorithm.
464.h264ref	C	Video Compression	A reference implementation of H.264/AVC, encodes a videostream using 2 parameter sets. The H.264/AVC standard is expected to replace MPEG2
471.omnetpp	C++	Discrete Event Simulation	Uses the OMNet++ discrete event simulator to model a large Ethernet campus network.
473.astar	C++	Path-finding Algorithms	Pathfinding library for 2D maps, including the well known A* algorithm.
483.xalancbmk	C++	XML Processing	A modified version of Xalan-C++, which transforms XML documents to other document types.

Home - Contact - Site Map - Privacy - About SPEC

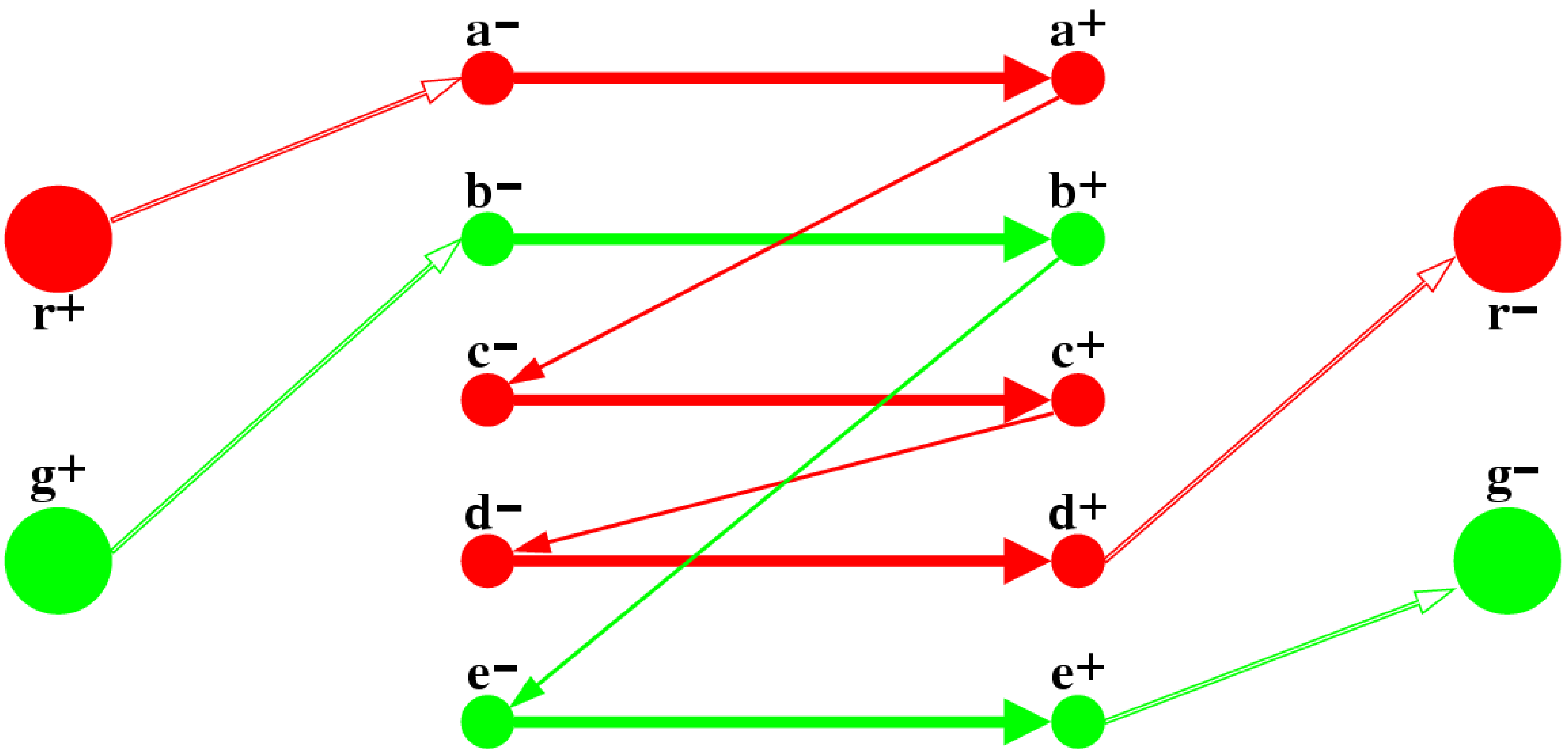
webmaster@spec.org
Last updated: *Thu Aug 24 00:44:00 EDT 2006*

URL: *http://www.spec.org/cpu2006/CINT2006/index.html*

Suchen: bench Abwärts Aufwärts Hervorheben Groß-/Kleinschreibung

http://www.spec.org/cpu2006/Docs/429.mcf.html Now: Stark bewölkt, 5° C Mi: 7° C Do: 8° C

Lagrangean Pricing Algorithm
(Löbel [1997])



Lagrangean Relaxation II

$$\min \quad \sum_{d} \sum_{ij} c_{ij}^{d} x_{ij}^{d}$$

$\sum_{i} x_{ij}^{d} - \sum_{k} x_{jk}^{d}$	$=$	0	$\forall\, j,d$	Vehicle flow
$\sum_{d} \sum_{i} x_{ij}^{d} - \sum_{d} \sum_{k} x_{jk}^{d}$	$=$	0	$\forall\, j$	Aggregated flow
$\sum_{d} \sum_{i} x_{ij}^{d}$	$=$	1	$\forall\, j$	Timetabled trips
$\sum_{j} x_{0j}^{d}$	$\leq$	κ_{d}	$\forall\, d$	Depot capacities
x_{ij}^{d}	$\in$	$\{0,1\}$	$\forall\, ij,d$	Deadhead trips

Lagrangean Relaxation II

$$
\begin{array}{lllll}
\max\limits_{\lambda} \; \min & \sum\limits_{d}\sum\limits_{ij} c_{ij}^{d} x_{ij}^{d} & + \; \lambda & \left(1 - \sum\limits_{d}\sum\limits_{i} x_{ij}^{d}\right) & \\
 & \sum\limits_{i} x_{ij}^{d} - \sum\limits_{k} x_{jk}^{d} & = \; 0 & \forall\, j,d & \text{Vehicle flow} \\
 & \sum\limits_{d}\sum\limits_{i} x_{ij}^{d} - \sum\limits_{d}\sum\limits_{k} x_{jk}^{d} & = \; 0 & \forall\, j & \text{Aggregated flow} \\
 & & & & \text{Timetabled trips} \\
 & \sum\limits_{j} x_{0j}^{d} & \le \; \kappa_{d} & \forall\, d & \text{Depot capacities} \\
 & x_{ij}^{d} & \in \; \{0,1\} & \forall\, ij,d & \text{Deadhead trips}
\end{array}
$$

Subproblem: Several independent Min-Cost-Flows (single-depot)

Heuristics

Cluster First – Schedule Second

- "Nearest-depot" heuristic
- Lagrange Relaxation II + tie breaker

Schedule First – Cluster Second

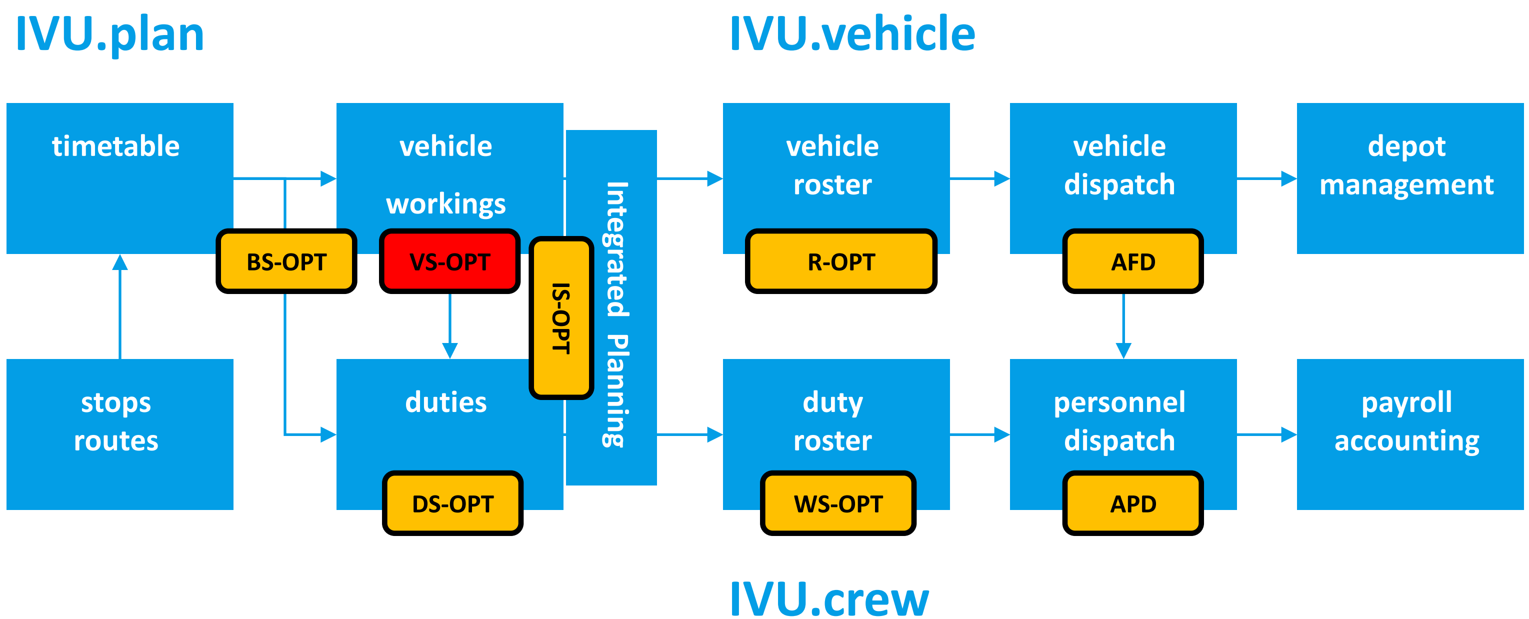
- Lagrange relaxation I

Schedule – Cluster – Reschedule

- Schedule: Lagrange relaxation I
- Cluster: Look at paths
- Solve a final min-cost flow

Plus tabu search

IVU.plan / IVU.crew / IVU.vehicle
System Overview with Optimization Modules



BVG (Berlin)

Systematisierter Einsatz

Die neuen Optimierungsmethoden, die die BVG jetzt nach und nach nutzen will, stammen vom Konrad-Zuse-Zentrum für Informationstechnik und garantieren nach Roß' Angaben Einsparungen von maximal 100 Millionen Mark im Jahr. „Sie sind nötig, um unser Angebot in dieser schweren Lage stabilisieren und dem Einsparungsdruck überhaupt standhalten zu können."

Bereits 1991 beauftragte die BVG die Berliner Software-Firma IVU, ein EDV-System zur Betriebsplanung zu entwikkeln. IVU steht für „Gesellschaft für Information, Verkehrs- und Umweltplanung GmbH", ein Unternehmen mit 120 Mitarbeitern, das auf Verkehrsplanung und Logistik spezialisiert ist. Der BVG ging es bei dem Auftrag vor allem darum, die Einsatzplanung ihrer Fahrzeuge zu systematisieren.

WISSENSCHAFT UND PRAXIS

Rheinischer Merkur

Nummer 39 · 26. September 1997 **37**

INFORMATIK / *Ein Lehrbeispiel, wie sich Mathematik und Wirtschaft ergänzen*

Auf Sparkurs zum Ziel

Das Berliner Busnetz kostet jährlich Millionen. Mit Hilfe moderner Software könnte man auf gewaltige Zuschüsse verzichten.

■ VASCO ALEXANDER SCHMIDT

Die Mathematik ist die Wissenschaft abstrakter Probleme. Deshalb erscheint sie so oft als weltabgewandte Spielerei. Doch das ist nur die halbe Wahrheit. Längst mischt sich die Mathematik in die Praxis ein. Überall, wo in der Wirtschaft gespart und verbessert werden muß, kann sie helfen. Professor Martin Grötschel, Vizepräsident des Konrad-Zuse-Zentrums für Informationstechnik in Berlin, steht für das neue Selbstbewußtsein der angewandten Mathematik. Er ist Experte für die kombinatorische Optimierung. Seine Botschaft: „Wer heute die großen Verkehrsnetze und komplizierten Fabriken optimieren will, muß sich mit Mathematik beschäftigen, um unnötige Kosten zu vermeiden."

Nun gibt es in Berlin ein Lehrbeispiel, wie Mathematik und Wirtschaft zueinanderkommen können. Die Berliner Verkehrsbetriebe (BVG) haben vor wenigen Wochen begonnen, die Einsatzplanung ihrer Fahrzeuge zu verbessern. Bisher wurden die Pläne, wenn auch mit Computerhilfe, per Hand erstellt. Von einer kostengünstigen Planung war man weit entfernt.

Um ihre Kosten zu decken, hat die BVG Jahr für Jahr große Zuschüsse vom Berliner Senat bekommen. Doch damit ist nun Schluß; dem Land Berlin geht das Geld aus, so daß auch die BVG unter einem enormen Sparzwang steht. „Wir müssen jährlich dreistellige Millionenbeträge einsparen", erklärt Jürgen Roß, Planungsingenieur bei der BVG. „Bis zum Jahr 2000 können wir von den heute rund 20 000 Mitarbeitern nur noch 15 000 beschäftigen."

Systematisierter Einsatz

Die neuen Optimierungsmethoden, die die BVG jetzt nach und nach nutzen will, stammen vom Konrad-Zuse-Zentrum für Informationstechnik und garantieren nach Roß' Angaben Einsparungen von maximal 100 Millionen Mark im Jahr. „Sie sind nötig, um unser Angebot in dieser schweren Lage stabilisieren und dem Einsparungsdruck überhaupt standhalten zu können."

Bereits 1991 beauftragte die BVG die Berliner Software-Firma IVU, ein EDV-System zur Betriebsplanung zu entwikkeln. IVU steht für „Gesellschaft für Information, Verkehrs- und Umweltplanung GmbH", ein Unternehmen mit 120 Mitarbeitern, das auf Verkehrsplanung und Logistik spezialisiert ist. Der BVG ging es bei dem Auftrag vor allem darum, die Einsatzplanung ihrer Fahrzeuge zu systematisieren.

So sollten Linien und Fahrpläne [illegible] erstellt werden können. Außerdem wollte man die Dienstpläne und die Fahrpläne, die an den über 10 000 U-, Straßenbahn- und Bushaltestellen der Stadt aushängen, auf Knopfdruck und ohne Umwege über eine Druckerei im eigenen Haus produzieren. Nach und nach wurden diese Werkzeuge eingeführt. Das jetzt installierte Optimierungsmodul für die Umlaufplanung von Straßenbahnen und Bussen ist der bisher bedeutendste Schritt der Zusammenarbeit.

Komfortable Hilfe

Zwar werden in Deutschland für die Verkehrsplanung meist schon Computer eingesetzt; diese aber unterstützen die Planer oft nur als ein einfaches, wenn auch komfortables Hilfsmittel. Andere Verkehrsunternehmen, etwa die Hamburger Hochbahn AG, setzen Computer schon seit Ende der siebziger Jahre ein, um Näherungslösungen für optimale Pläne zu berechnen. Die Forscher am Konrad-Zuse-Zentrum konnten noch einen Schritt weiter gehen: Ihre Computer berechnen das Optimum der Kosteneinsparung nicht nur annähernd, sondern ganz exakt. Ein wissenschaftlicher Durchbruch.

Die mathematische Grundlage ihrer Rechner bildet eine komplexes Gleichungssystem, eine sogenannte Matrix, mit mehr als 100 000 Zeilen und 70 Millionen Spalten. Die Zahlen in der gigantischen Tabelle geben an, wie die Busse eingesetzt werden sollen. Die Größe der Matrix weist auf ungezählte Details hin, die beachtet werden müssen: Der Computer muß garantieren, daß jeder der rund 1800 BVG-Busse morgens sein Depot verläßt und nach Dienstschluß dort wieder landet.

Es gibt Eindecker, Doppeldecker, Gelenk- und Minibusse, aber nicht jeder Bustyp kann jede Route bedienen. Außerdem sind komplizierte betriebliche und rechtliche Bedingungen zu beachten, etwa Pausenregelungen. Ziel ist es, die einzelnen Busfahrten so zu verketten, daß die Arbeitszeit der Fahrer gut ausgenutzt wird und die Leerfahrten von einem Einsatzort zum nächsten möglichst kurz sind. Alle diese Zielvorgaben stecken in der riesigen Matrix, die die Mathematiker „ganzzahliges lineares Programm" nennen. Ein lineares Programm ist für Mathematiker eine alltägliche Struktur. Sie taucht bei fast allen Fragen nach optimalen Mischverhältnissen, bei Güterflüssen in einer Fabrik und auch bei Stundenplänen auf.

Bildlich gesprochen besteht das Problem – in seiner einfachsten Form – aus einem Vieleck in der Ebene und einer Geraden, die durch das Vieleck verläuft. Nun verschiebt man die Gerade nach rechts. Ziel ist es, den letzten Punkt des Vielecks zu bestimmen, den die Gerade bei diesem Verschiebe-Prozeß gerade noch berührt. Hier liegt die kostengünstigste Lösung des Optimierungsproblems. Das Vieleck bei den Bus-Umlaufplänen ist sehr viel komplexer, man sollte es sich eher als einen verwinkelten Kristall mit mehreren Billionen Ecken und Kanten vorstellen.

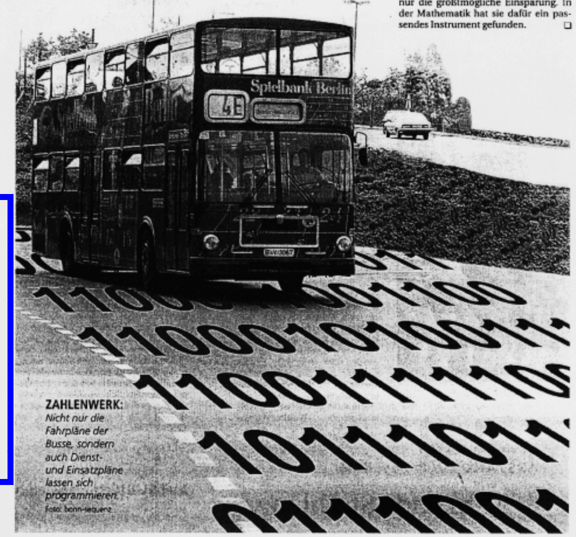
„Eigentlich ist die Matrix für die Umlaufplanung bei der BVG so groß, daß man sie noch nicht einmal vollständig im Computer speichern könnte", verrät Andreas Löbel, Wissenschaftlicher Mitarbeiter von Martin Grötschel. „Pro Spalte gibt es aber nur drei Einträge, die von Null verschieden sind." Das bedeutet, daß sich die Matrix stark vereinfachen läßt. Und genau diese Eigenschaft machen sich die Mathematiker zunutze. So versucht ihr Algorithmus zuerst mit einem Teil der Matrix zu operieren. Findet er für diesen kleinen Teil eine Lösung, so vergrößert er nach und nach das Problem, bis er eine Lösung für die gesamte Matrix gefunden hat.

Pfiffige Programme

Von der Mathematik spüren die Planer bei der BVG wenig. Nicht einmal Großrechner werden gebraucht – so pfiffig wurde das System programmiert. Je nach Teilproblem dauert die Bearbeitung wenige Stunden oder sogar nur Minuten. „Das System wurde gut aufgenommen, da die Planer in kürzester Zeit verschiedene Szenarien testen und vergleichen können", berichtet Uwe Strubbe, der als Projektleiter bei der IVU für die Entwicklung des fertigen Softwareprodukts zuständig ist.

Auch Martin Grötschel betont den Nutzen für den Planer mehr als das Einsparungspotential: „Wir geben den Leuten Hilfsmittel in die Hand, die Kosten zu sparen: Ob dabei am Ende etwas eingespart oder der Service kostenneutral verbessert wird, ist nicht unsere, sondern eine politische Frage."

Die Politik freilich erwartet zur Zeit nur die größtmögliche Einsparung. In der Mathematik hat sie dafür ein passendes Instrument gefunden. □



ZAHLENWERK: *Nicht nur die Fahrpläne der Busse, sondern auch Dienst- und Einsatzpläne lassen sich programmieren.*

Foto: bonn-sequenz

Urban Scenarios

	BVG	HHA	VHH
depots	10	14	10
vehicle types	44	40	19
timetabled trips	25 000	16 000	5 500
deadheads	70 000 000	15 100 000	10 000 000
cpu mins	200	50	28

Integer Programming Model
(Multi-Commodity Flow Problem)

Freie Universität Berlin

$$\min \quad \sum_{d} \sum_{ij} c_{ij}^{d} x_{ij}^{d}$$

$$\sum_{i} x_{ij}^{d} - \sum_{k} x_{jk}^{d} = 0 \quad \forall j, d \quad \text{Vehicle Flow}$$

$$\sum_{d} \sum_{i} x_{ij}^{d} - \sum_{d} \sum_{k} x_{jk}^{d} = 0 \quad \forall j \quad \text{Aggregate Flow}$$

$$\sum_{d} x_{ij}^{d} = 1 \quad \forall j \quad \text{Trips}$$

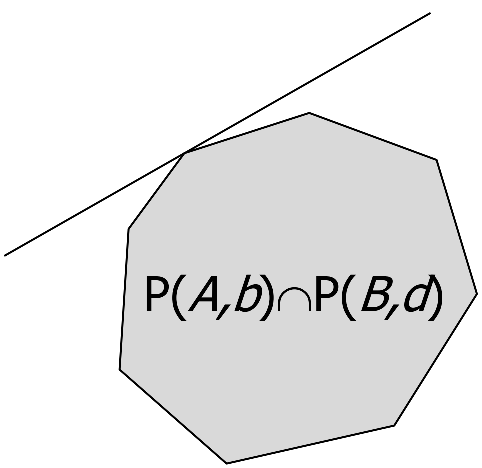
$$\sum_{j} x_{dj}^{d} \leq \kappa^{d} \quad \forall d \quad \text{Capacities}$$

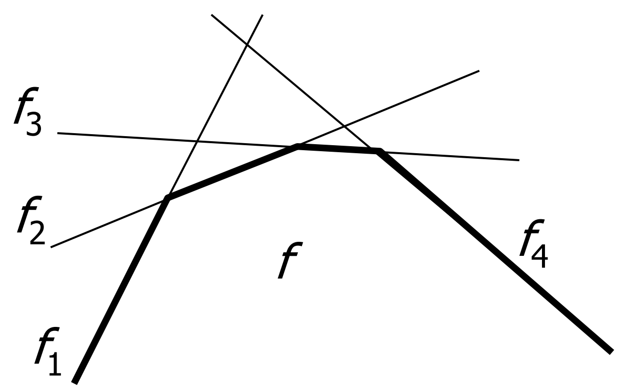
$$x_{ij}^{d} \in \{0,1\} \quad \forall ij, d \quad \text{Integrality}$$

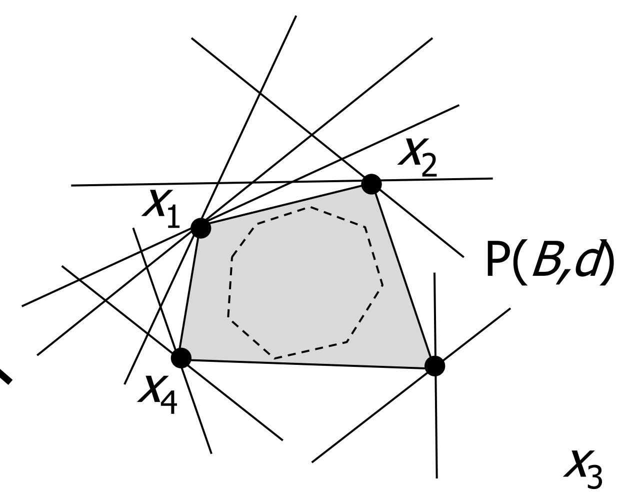
Lagrangean Relaxation

$$
\begin{array}{llll}
\min & c^T x & & \\
 & Ax & = & b \\
 & Bx & = & d \\
 & x & \geq & 0
\end{array}
\quad = \quad
\begin{array}{llllll}
\max\limits_{\lambda} & \min & c^T x & + & \lambda(b - Ax) \\
 & & Bx & = & d \\
 & & x & \geq & 0
\end{array}
$$

$$
= \max_{\lambda} \quad f(\lambda) \quad = \quad \max_{\lambda} \quad \min_{i} \quad c^T x_i \quad + \quad \lambda(b - Ax_i)
$$



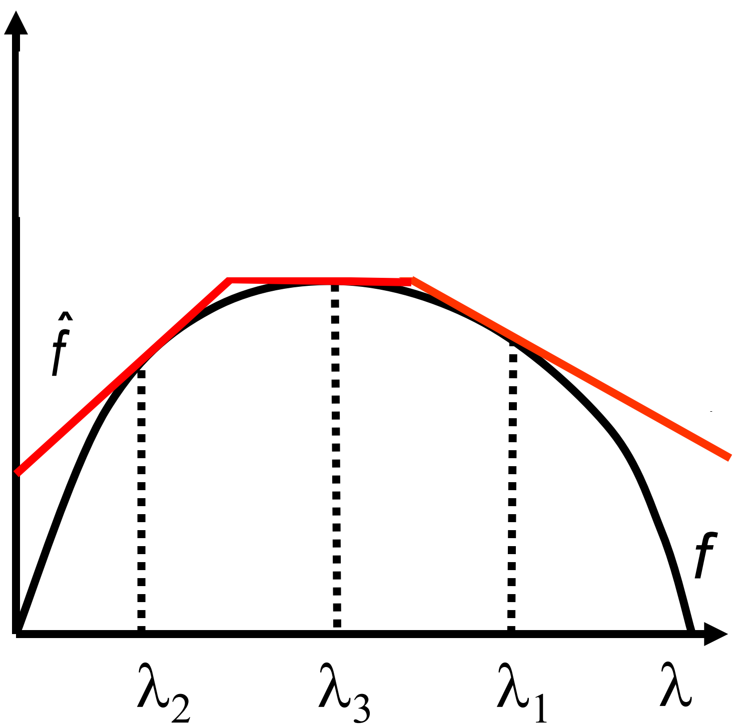




Subgradient Method

max $\quad f(\lambda) := \min_{x \in X} c^T x + \lambda^T (b - Ax)$

X =conv $\{x_\mu\}$ polyhedral (piecewise linear)



$$\bar{f}_\mu(\lambda) = c^T x_\mu + \lambda^T (b - Ax_\mu)$$

$$\hat{f}_k(\lambda) := \min_{\mu \in J_k} \bar{f}_\mu(\lambda)$$

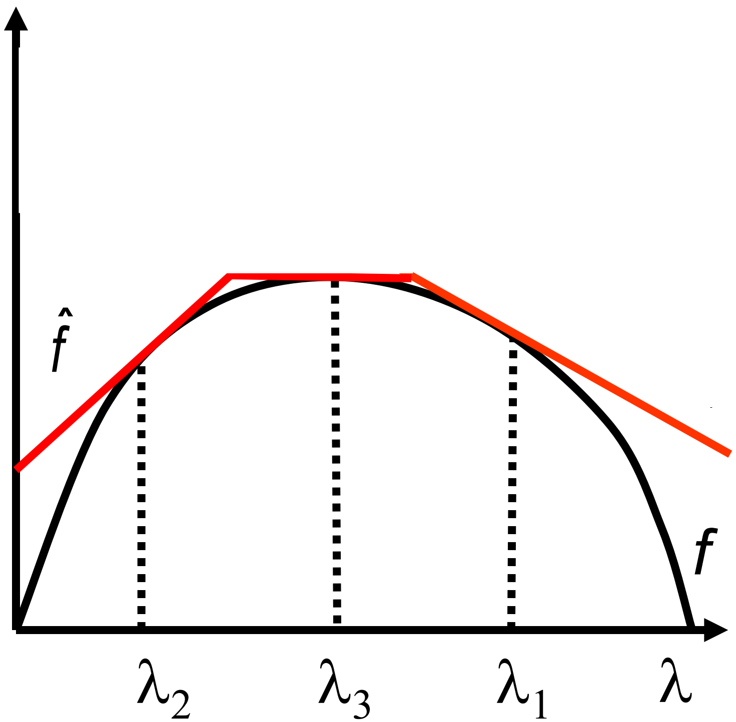
$$\lambda_{k+1} := \lambda_k + u_k \underbrace{(b - Ax_{\mu_k})}_{\text{subgradient}}$$

Bundle Method

(Kiwiel [1990], Helmberg [2000])

max $\quad f(\lambda) := \min_{x \in X} c^T x + \lambda^T (b - Ax)$

X =conv $\{x_\mu\}$ polyhedral (piecewise linear)



$$\bar{f}_\mu(\lambda) = c^T x_\mu + \lambda^T (b - Ax_\mu)$$

$$\hat{f}_k(\lambda) := \min_{\mu \in J_k} \bar{f}_\mu(\lambda)$$

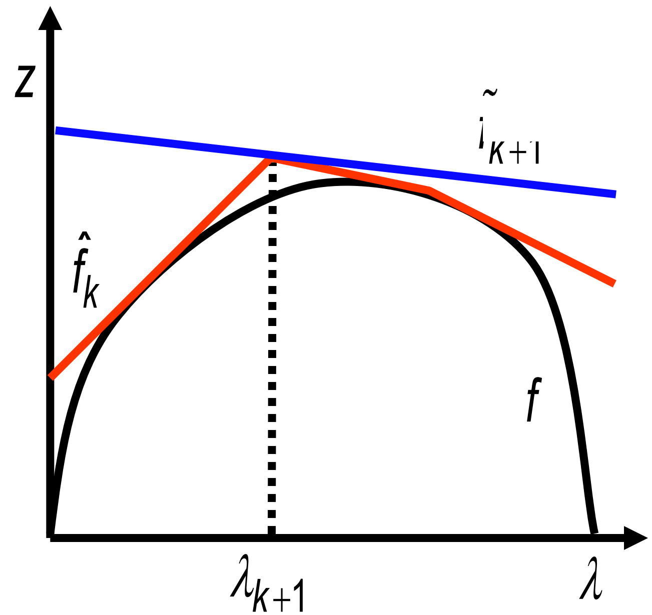
$$\lambda_{k+1} = \operatorname*{argmax}_{\lambda} \hat{f}_k(\lambda) - \frac{u_k}{2} \left\| \lambda - \hat{\lambda}_k \right\|^2$$

Primal Approximation

Theorem:

$$\lambda_{k+1} = \hat{\lambda}_k + \frac{1}{u} \sum_{\mu \in J_k} \alpha_\mu (b - Ax_\mu)$$

$$\tilde{x}_{k+1} \quad \sum_{\mu \in J_k} \alpha_\mu x_\mu$$



$$\|b - A\tilde{x}_k\| \quad (k \to \infty)$$

$$\Rightarrow (\tilde{x}_k)_{k \in N} \text{ converges to a point } \bar{x} \in \{x : Ax = b, x \in X\}$$

Quadratic Subproblem

$$(1) \quad \max \hat{f}_k(\lambda) - \frac{u_k}{2}\left\|\lambda - \hat{\lambda}_k\right\|^2$$

$\Leftrightarrow$

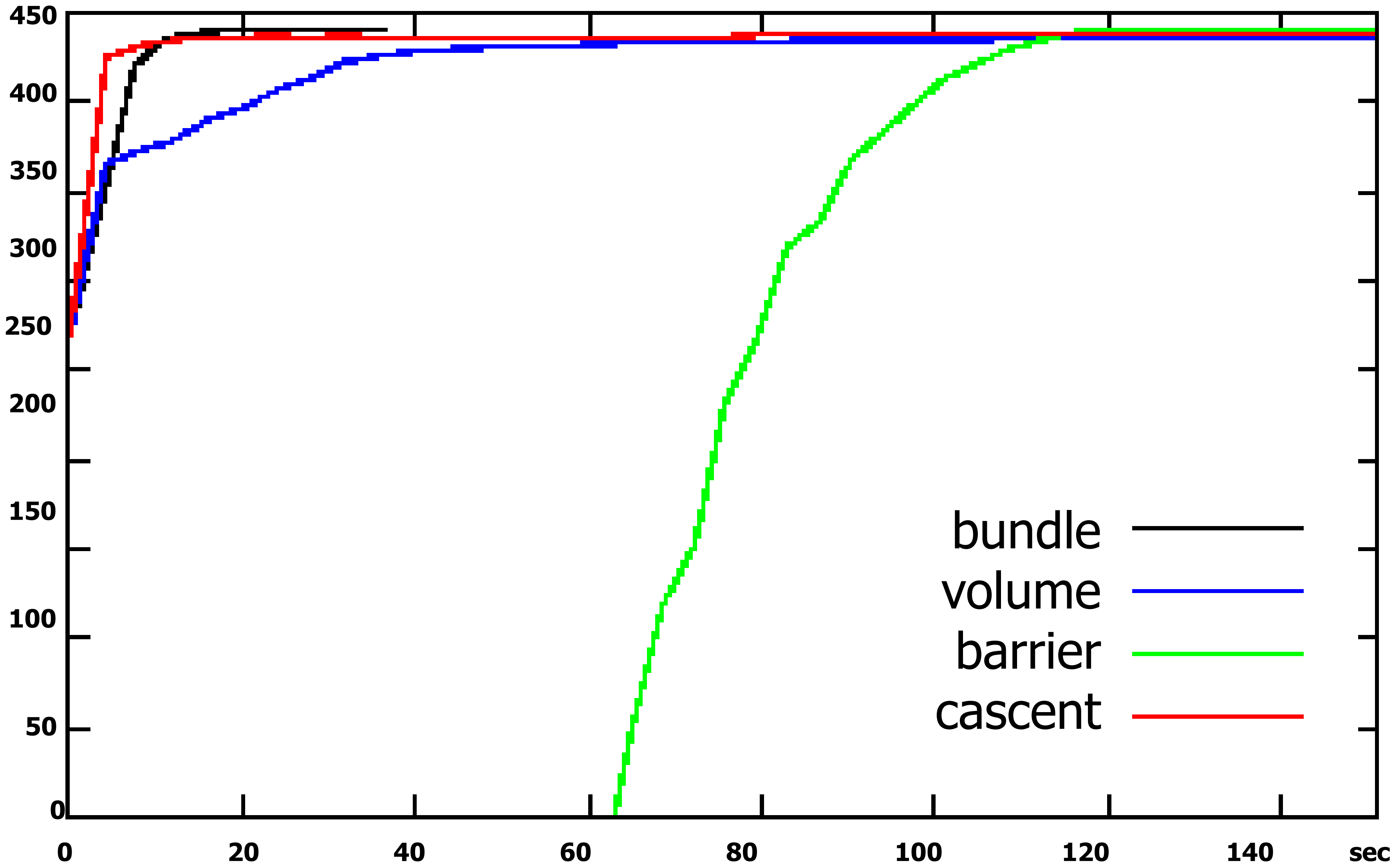
$$(2) \quad \max \quad v - \frac{u_k}{2}\left\|\lambda - \hat{\lambda}^k\right\|^2$$
$$\text{s.t.} \quad v \leq \bar{f}_\mu(\lambda), \text{ for all } \mu \in J_k$$

$\Leftrightarrow$

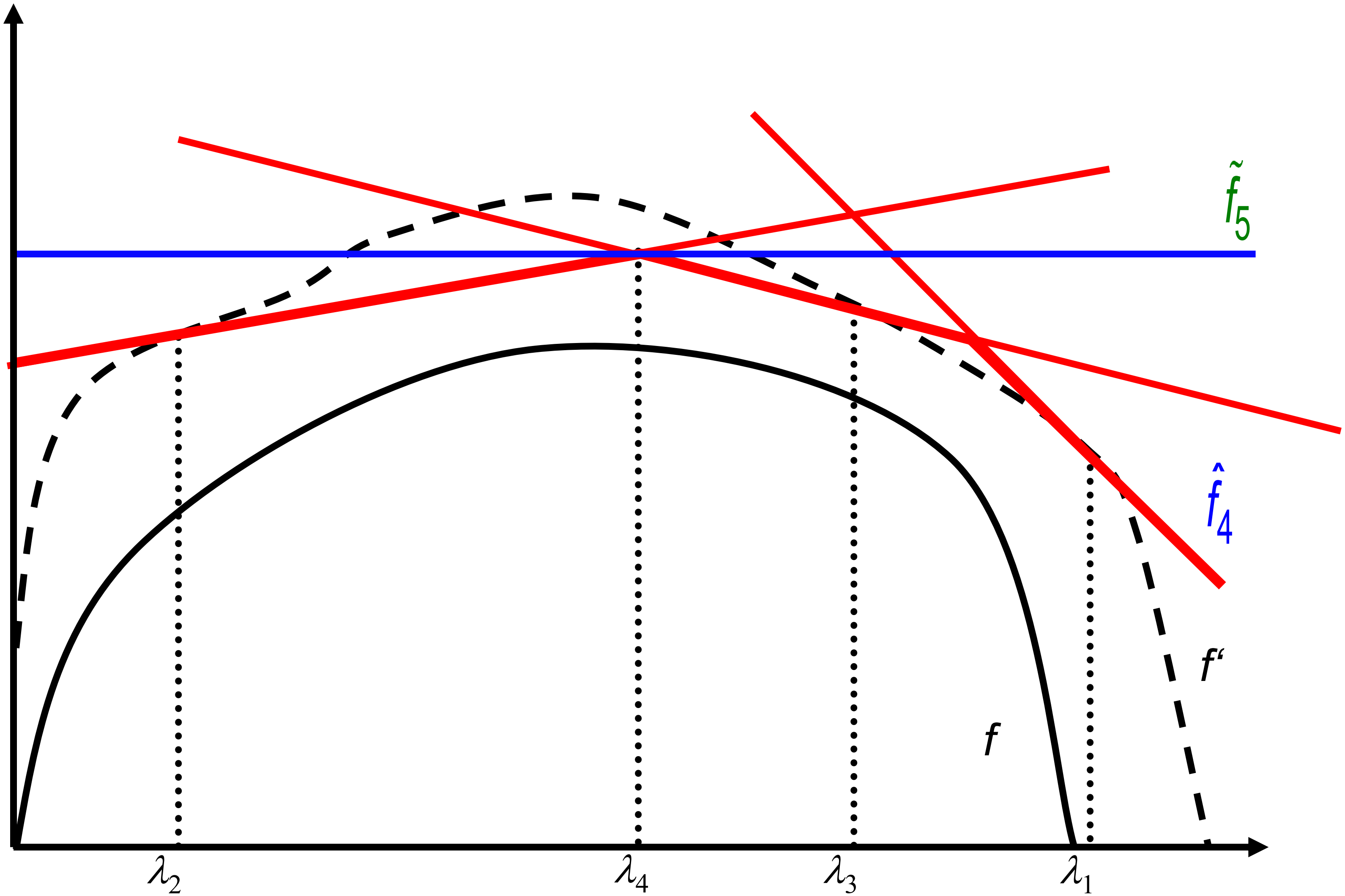
$$(3) \quad \min \quad \sum_{\mu \in J_k} \alpha_\mu \bar{f}_\mu(\hat{\lambda}) - \frac{1}{2u_k}\left\|\sum_{\mu \in J_k} \alpha_\mu (b - Ax_\mu)\right\|^2$$
$$\text{s.t.} \quad \sum_{\mu \in J_k} \alpha_\mu = 1$$
$$0 \leq \alpha_\mu \leq 1, \qquad \text{for all } \mu \in J_k$$

Bundle Method

(IVU41 838,500 x 3,570, 10.5 NNEs per column)



Heuristic Evaluation of f



Thank you for your attention

Ralf Borndörfer

Freie Universität Berlin
Zuse-Institute Berlin
Takustr. 7
14195 Berlin-Dahlem

Fon (+49 30) 84185-243
Fax (+49 30) 84185-269

borndoerfer@zib.de

www.zib.de/borndoerfer