

Vehicle Scheduling

CO@Work Berlin

Ralf Borndörfer

03.10.2009



DFG Research Center MATHEON
Mathematics for key technologies



- ▶ **Introduction**
- ▶ Single Depot Vehicle Scheduling
- ▶ Multiple Depot Vehicle Scheduling
- ▶ Lagrangean Relaxation
- ▶ Extensions
 - ▶ Trip Shifting
 - ▶ Integrated Vehicle and Duty Scheduling
 - ▶ Trip Scheduling

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 entwickeln. 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.

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 weitabgewandte Spielerei. Doch in der Wirtschaft greifen sie immer wieder in die Praxis ein. Oberall, wo in der Wirtschaft große und komplexe Probleme gelöst werden müssen, kommt die Mathematik zum Einsatz. Ein Beispiel ist die Verkehrsplanung. 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 für das Jahr große Zuschüsse von der Berliner Senat bekommen. Doch damit ist nun Schluss, denn 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, Planungsdirektor 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 entwickeln. 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.

Stadt anhängen, 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. Anders Verkehlsunternehmen, etwa die Hamburger Hochbahn AG, setzen Computer schon seit Ende der sechziger 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 Kostengünstigkeit nicht nur annähernd, sondern ganz exakt. Ein wissenschaftlicher Durchbruch.

Die mathematische Grundlage ihrer Rechner bildet eine komplexe Gleichungssysteme, 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 berücksichtigen, 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 sucht 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 Viereck in der Ebene und einer Gerade, die durch das Viereck verläuft. Nun verschiebt man die Gerade nach rechts. Ziel ist es, den letzten Punkt des Vierecks zu bestimmen, den die Gerade bei diesem Verschiebe-Prozess gerade noch berührt. Hier liegt die kostengünstigste Lösung des Optimierungsproblems. Das Viereck bei den Bus-Umlaufplänen ist sehr viel komplexer, man sollte es sich eher als einen verwickelten Kristall mit mehreren Billionen Ecken und Kanten vorstellen.

Eigentlich ist die Matrix für die Umlaufplanung bei der BVG so groß, daß

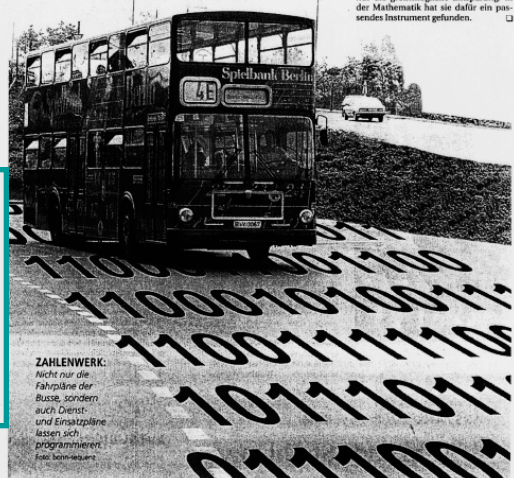
man sie noch nicht einmal vollständig im Computer speichern könnte“, verrät Andreas Löbel, Wissenschaftlicher Mitarbeiter von Martin Grötschel. „Pro Spalte gibt es aber nur drei Einträge, die von Null verschieden sind.“ Das bedeutet, daß sich die Matrix stark vereinfachen läßt. Und genau diese Eigenschaft 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 pfiffige wurde das System programmiert. Je nach Teilproblem dauern die Bearbeitungen 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 Straube, 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. □



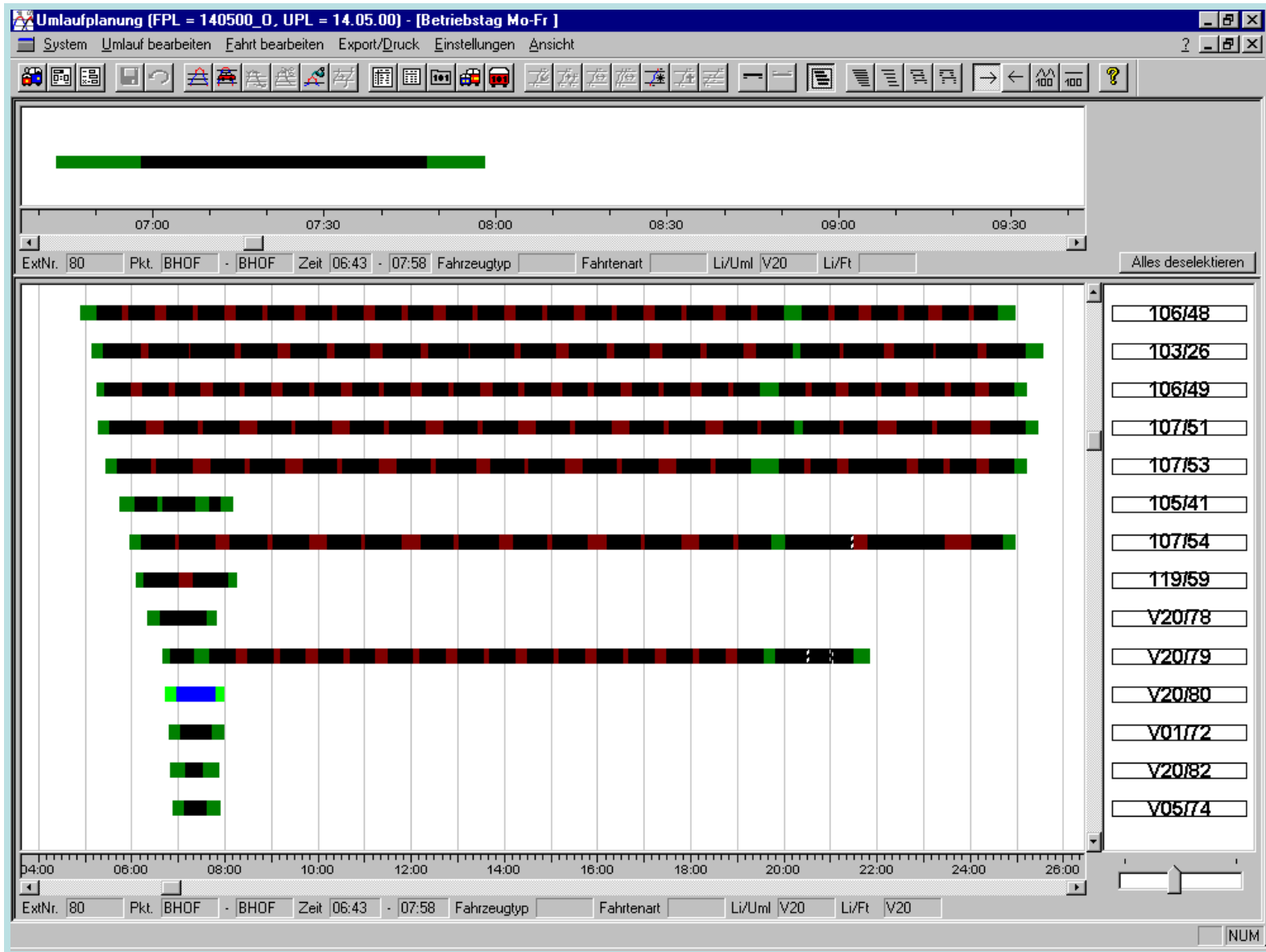
ZAHLENWERK: Nicht nur die Fahrpläne der Busse, sondern auch Dienst- und Einsatzpläne lassen sich programmieren. Foto: Konrad-Zuse-Zentrum

Leuthardt Survey

(Leuthardt 1998, Kostenstrukturen von Stadt-, Überland- und Reisebussen, DER NAHVERKEHR 6/98, pp. 19-23.)

<i>bus costs (DM)</i>	<i>urban</i>	<i>%</i>	<i>regional</i>	<i>%</i>
crew	349,600	73.5	195,000	67.5
depreciation	35,400	7.4	30,000	10.4
calc. interest	15,300	3.2	12,900	4.5
materials	14,000	2.9	10,000	3.5
fuel	22,200	4.7	18,000	6.2
repairs	5,000	1.0	5,000	1.7
other	34,000	7.1	18,000	7.2
total	475,500	100.0	288,900	100.0

Vehicle Utilization



"Camel Curve"

Fahrzeugeinsatz
intervallgenauer Fahrzeugbedarf
minutengenaue Fahrzeugbedarf

Umlaufversion
Umläufe gültig ab 14.05.2000

Betriebstag
Montag - Freitag

Betriebsbereich
Omnibusverkehr

Linien

- V20 V20: Mühlheim -> O
- 101 101: Markwaldstr.
- 102 102: A.-Bebel-Ring
- 103 103: Ffm., Prüflin
- 104 104: Neusalzer Str
- 105 105: Rosenhöhe <->
- 106 106: Buchrainweihe
- 107 107: Dt. Wetterdie
- 119 119: OF/Marktplatz
- 120 120: Dt. Wetterd.
- 101N 101N: Markwaldstr.
- 102N 102N: Kaiserlei <-
- 103N 103N: Ffm., Prüfli
- 104N 104N: W.-Schramm-S
- 105N 105N: Rosenhöhe <-
- 106N 106N: Buchrainwei
- 120N 120N: Dt. Wetterd.

Zeitraum von Uhr bis Uhr

Schrittweite [hh:mm]

Auswertung

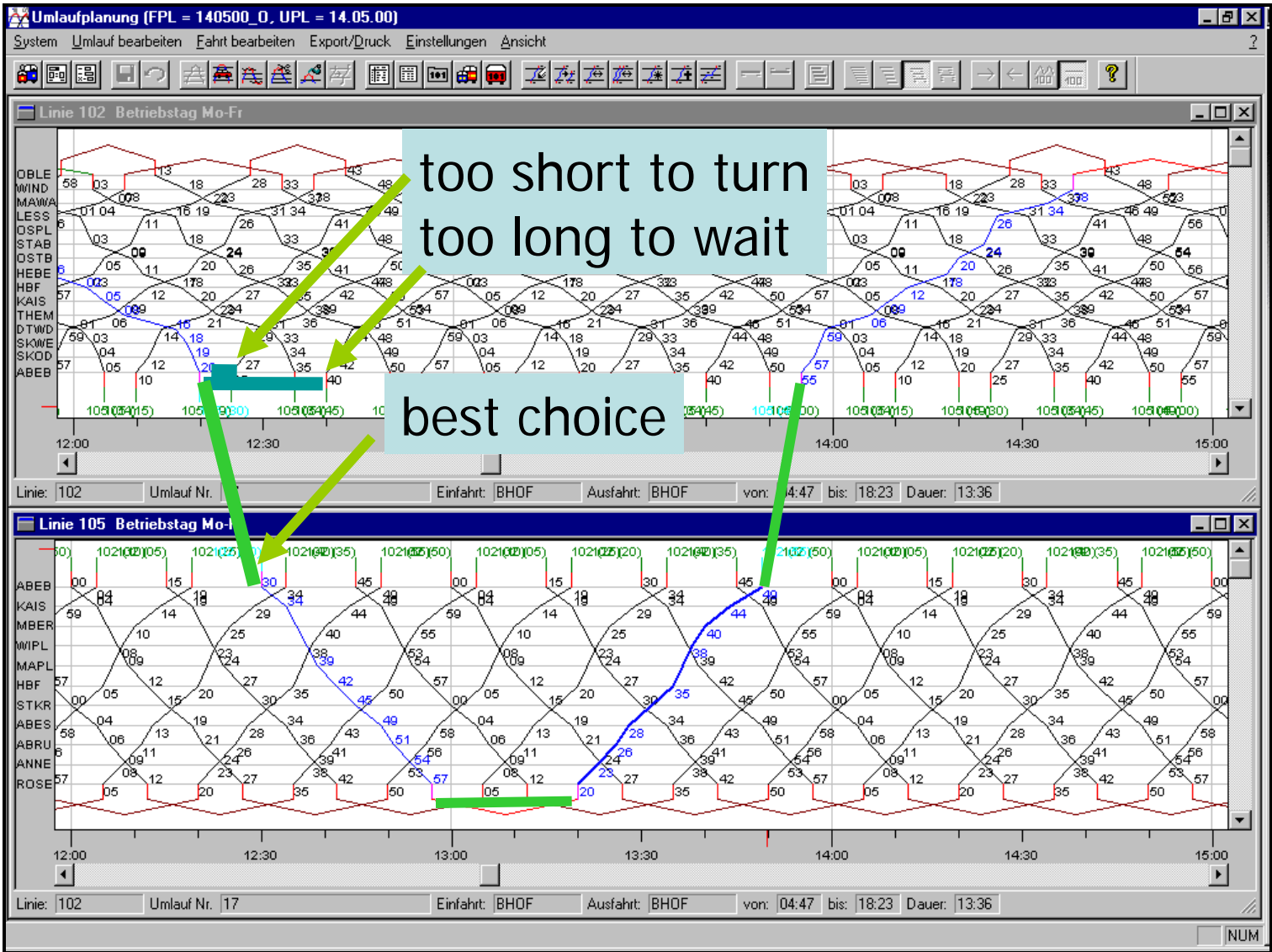
Beenden

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

Drucken

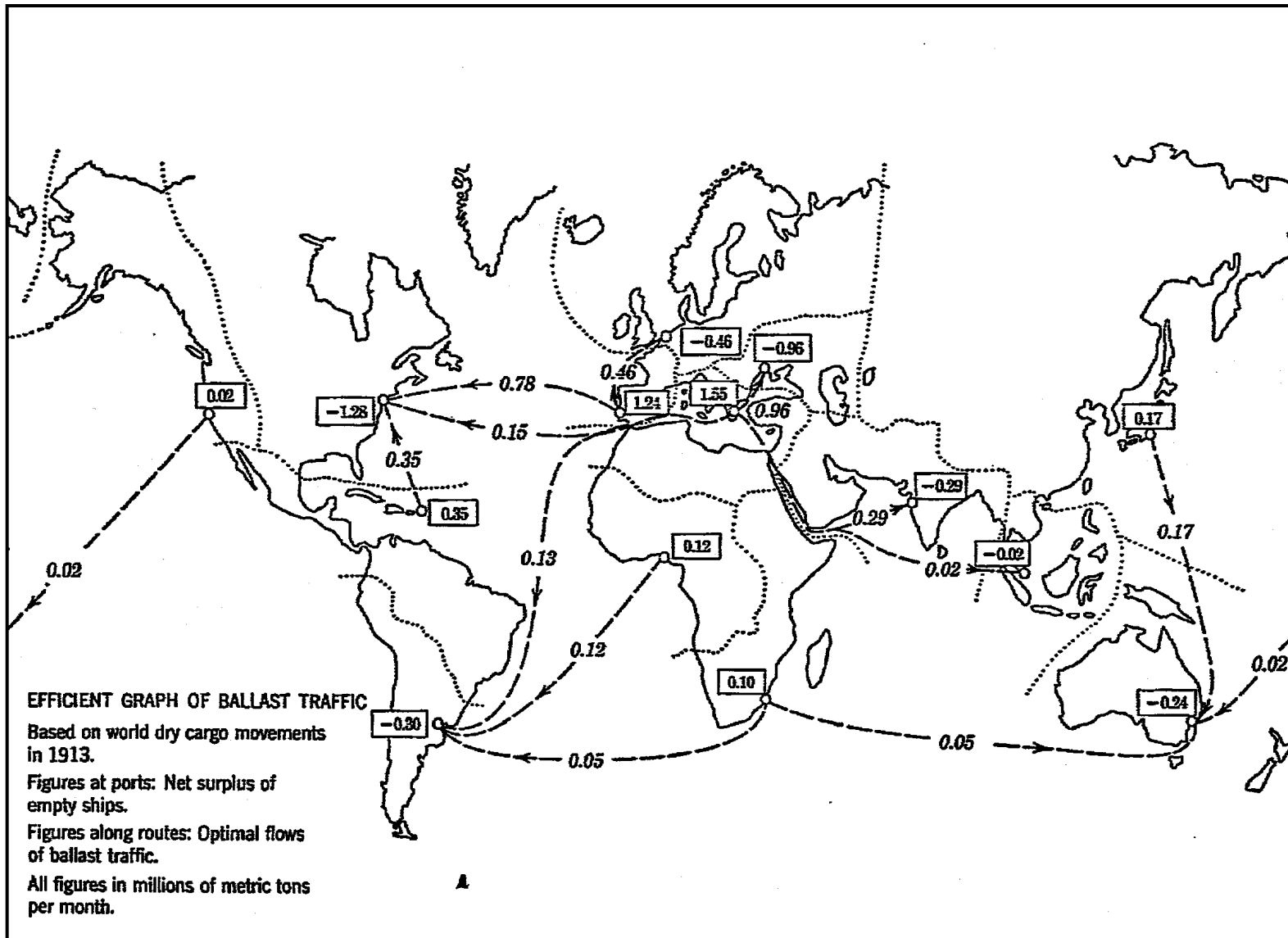
Interlining



- ▶ Introduction
- ▶ **Single Depot Vehicle Scheduling**
- ▶ Multiple Depot Vehicle Scheduling
- ▶ Lagrangean Relaxation
- ▶ Extensions
 - ▶ Trip Shifting
 - ▶ Integrated Vehicle and Duty Scheduling
 - ▶ Trip Scheduling

Sea Freight

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



Optimal Allocation of Scarce Resources

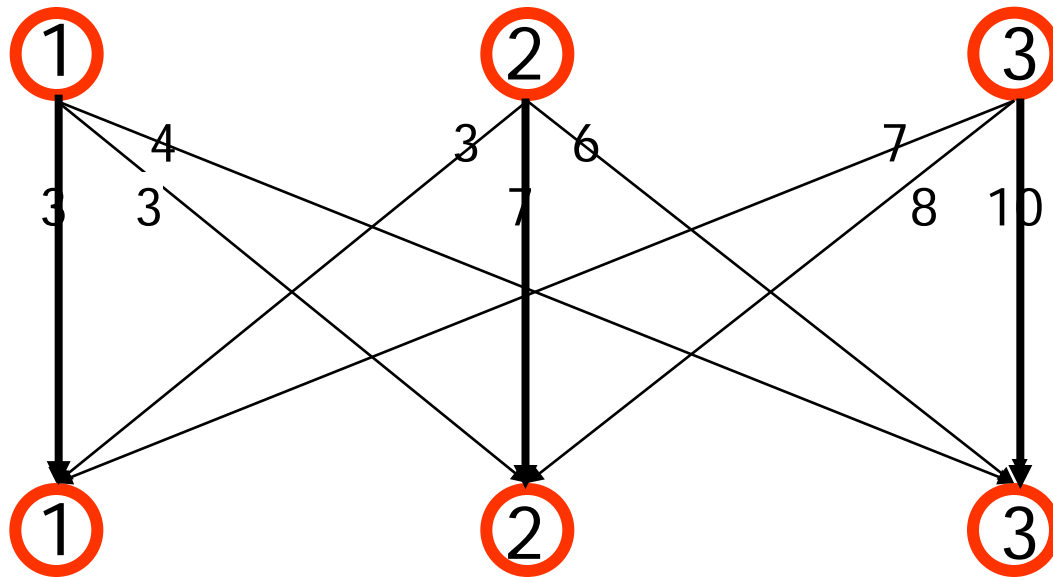
(Nobel Prize in Economics 1975)



▶ Leonid V. Kantorovich



▶ Tjalling C. Koopmans

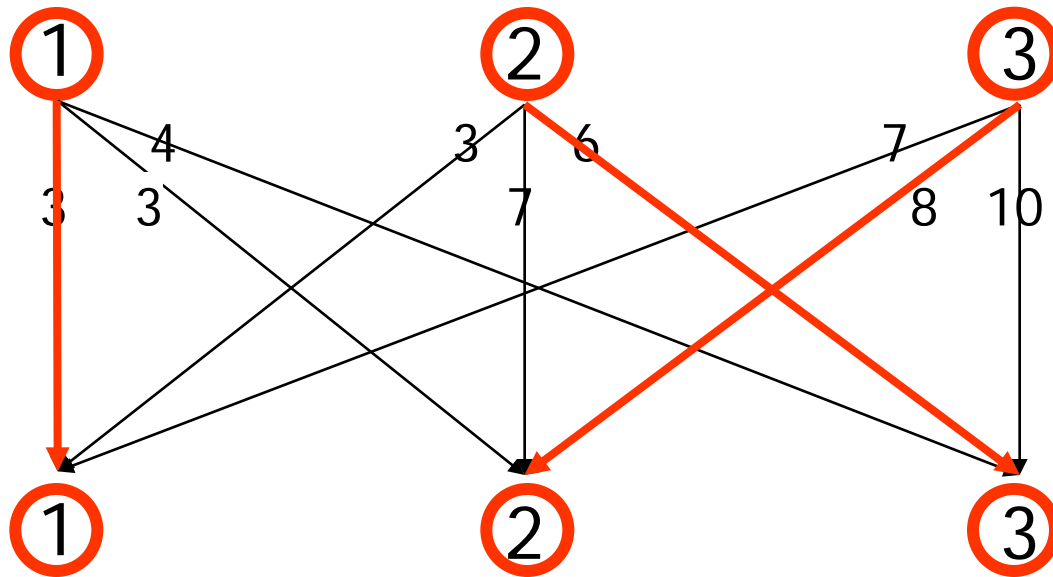


Buses

Solution
Cost = 20

Trips

- ▶ The Assignment Problem
 - ▶ Input: 3 Buses, 3 trips, costs
 - ▶ Output: cost minimal assignment

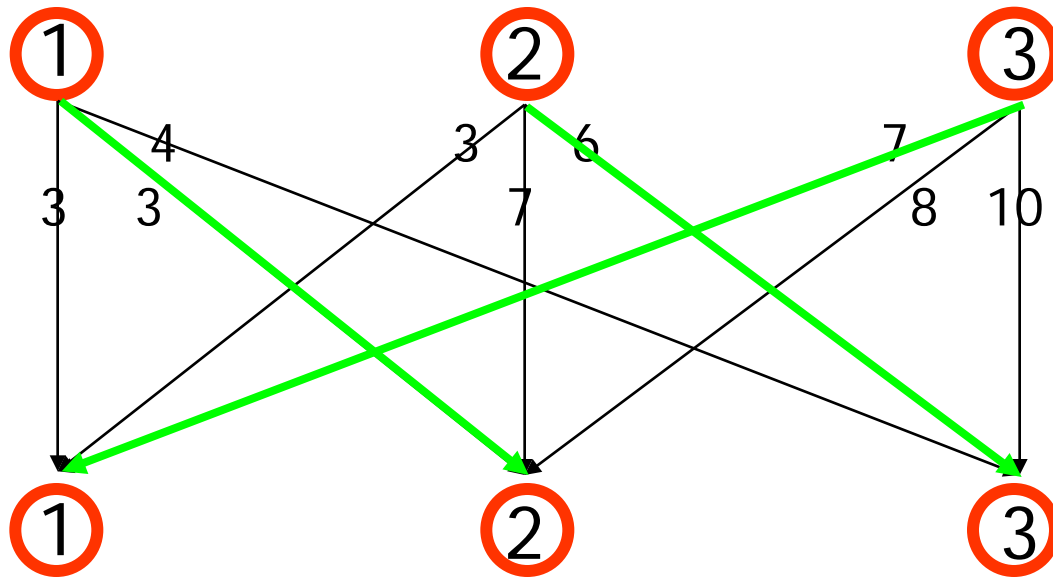


Buses

Solution
Cost = 17

Trips

- ▶ The Greedy-Heuristik
 - ▶ heuretikos (gr.): inventive
 - heuristicos (gr.): to find

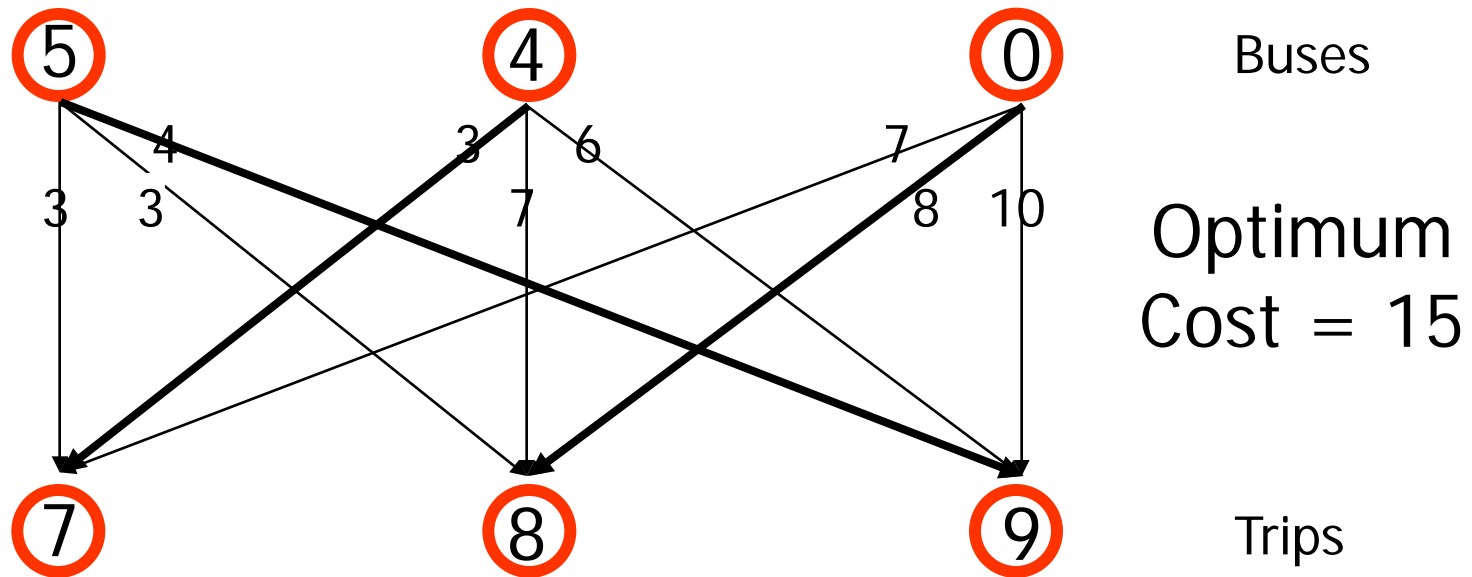


Buses

Solution
Cost = 16

Trips

- ▶ The Greedy-Heuristik
 - ▶ heuretikos (gr.): inventive
 - heuristicos (gr.): to find



▶ The "Primal Problem"

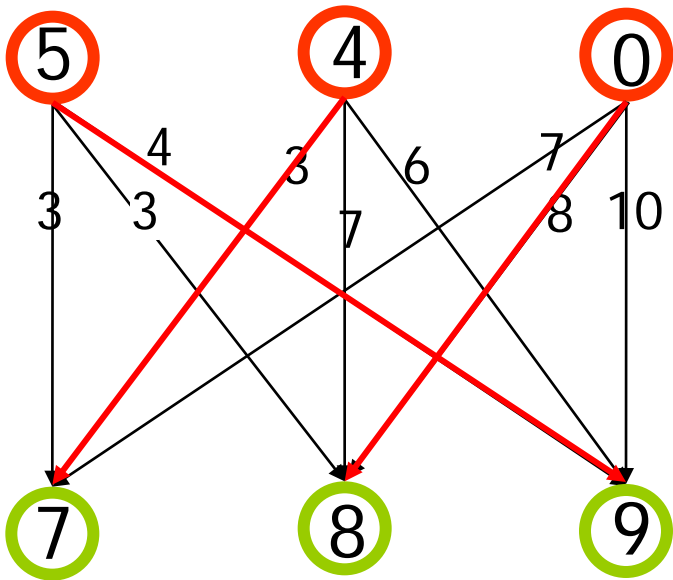
- ▶ Minimum Cost
- ▶ Assignment

▶ The "Dual Problem"

- ▶ Maximum Sales Revenues
- ▶ "Shadow Prices"

Mathematical Models

(Assignment Problem)



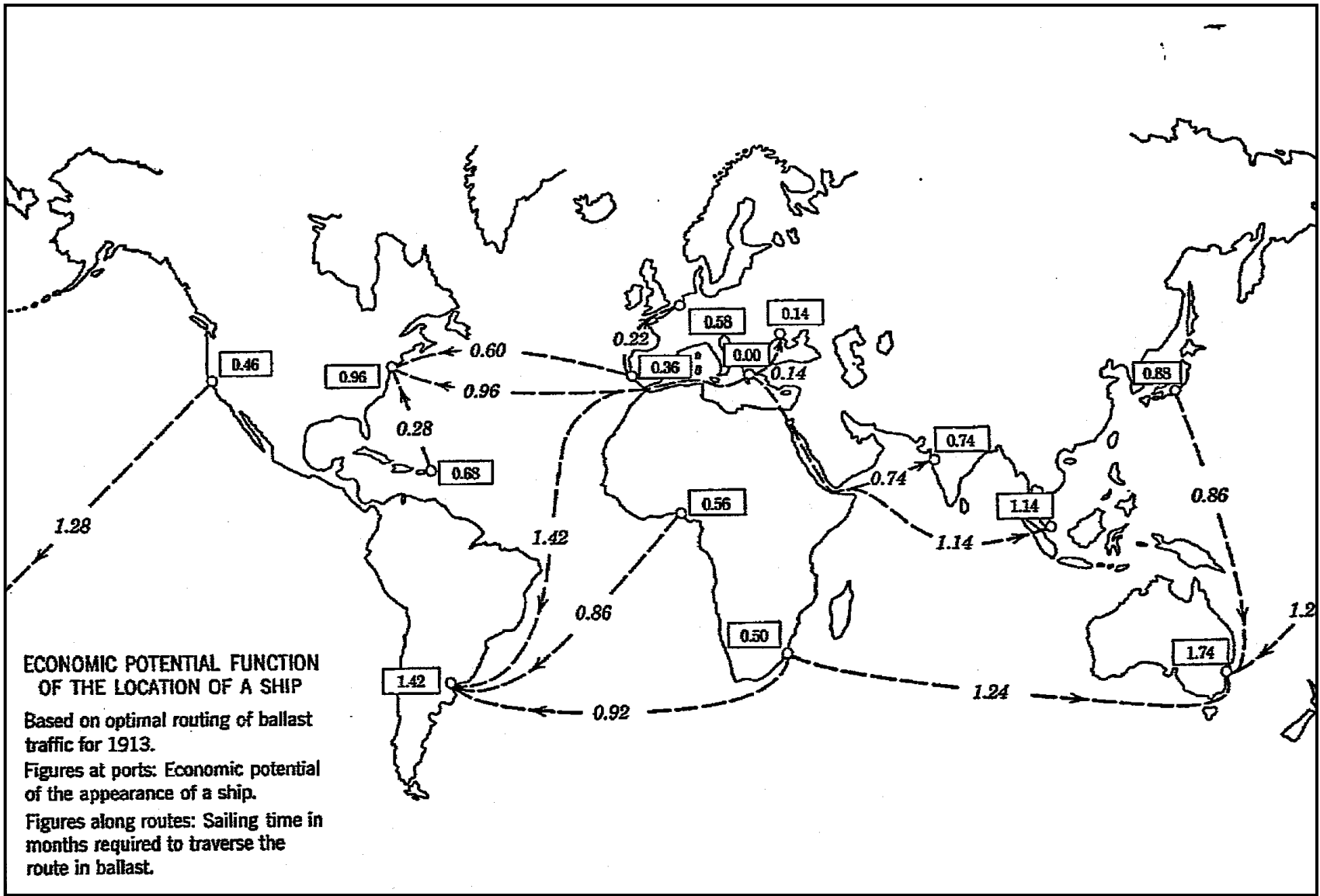
► Graph Theoretic Model

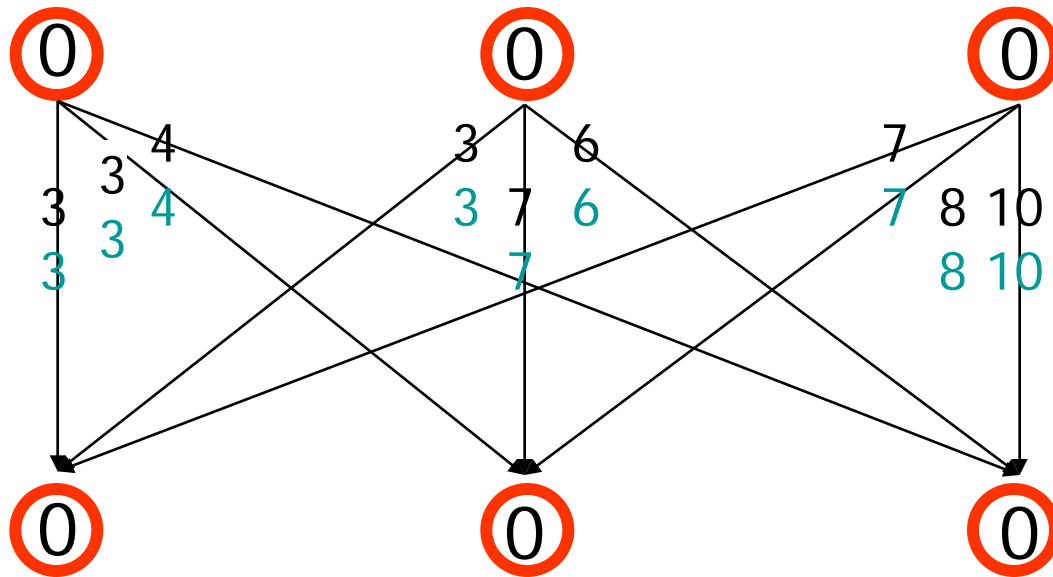
$$\begin{array}{llll}
 \min & 3x_{11} & +3x_{12} & +4x_{13} \\
 & +3x_{21} & +7x_{22} & +6x_{23} \\
 & +7x_{31} & +8x_{32} & +10x_{33} \\
 \text{s.t.} & x_{11} & +x_{12} & +x_{13} & = 1 \\
 & x_{21} & +x_{22} & +x_{23} & = 1 \\
 & x_{31} & +x_{32} & +x_{33} & = 1 \\
 & x_{11} & +x_{21} & +x_{31} & =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

► Integer Programming Model

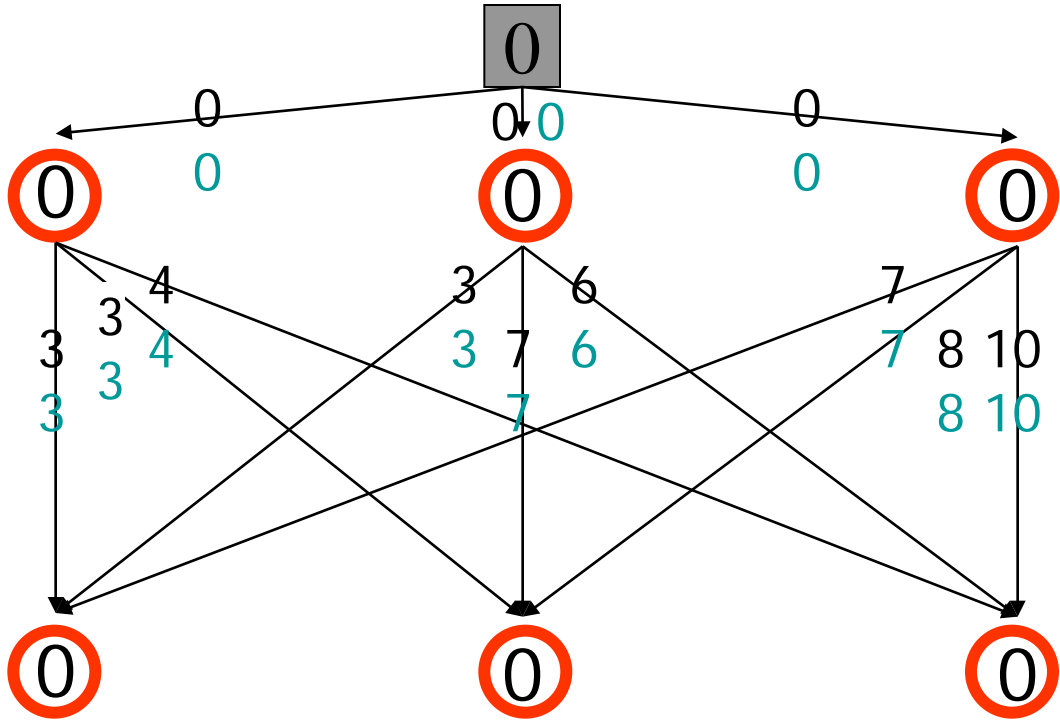
► Linear Programming Relaxation

"Shadow Prices"





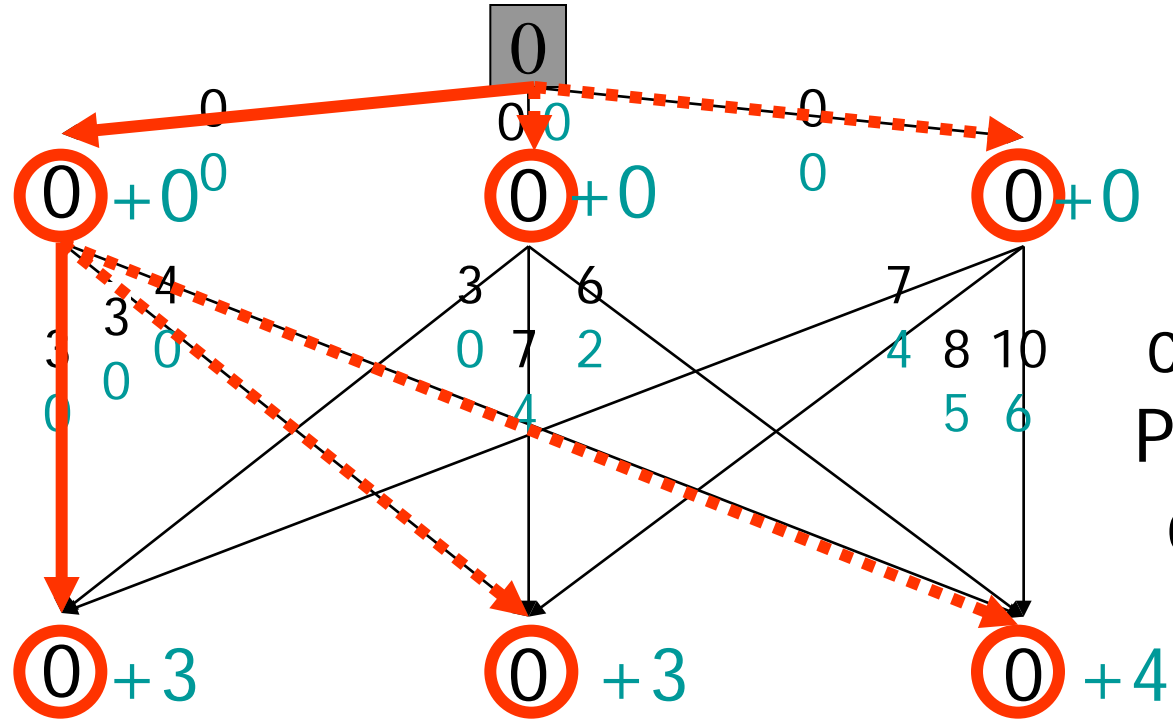
- ▶ The „Successive Shortest Path“ Algorithm



Buses
 Bound
 cost = 0
 Partial sol.
 cost = 0
 Trips

► The „Successive Shortest Path“-Algorithm

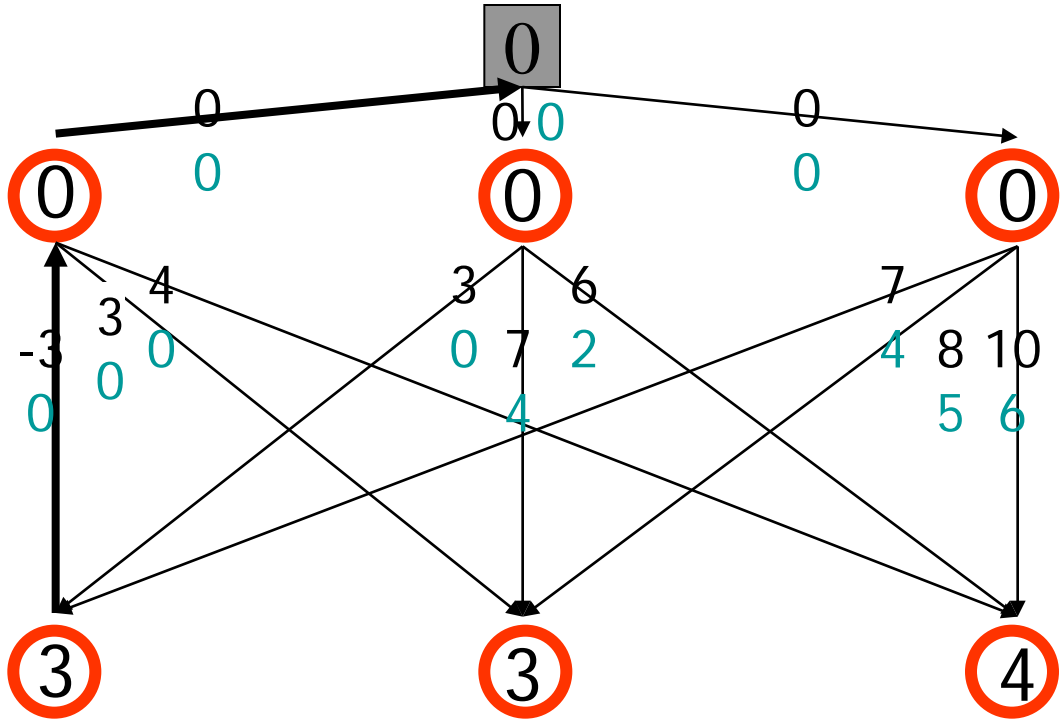
Single Depot Vehicle Scheduling



Buses
 Bound
 cost = 10
 Partial sol.
 cost = 3
 Trips

► The „Successive Shortest Path“ Algorithm

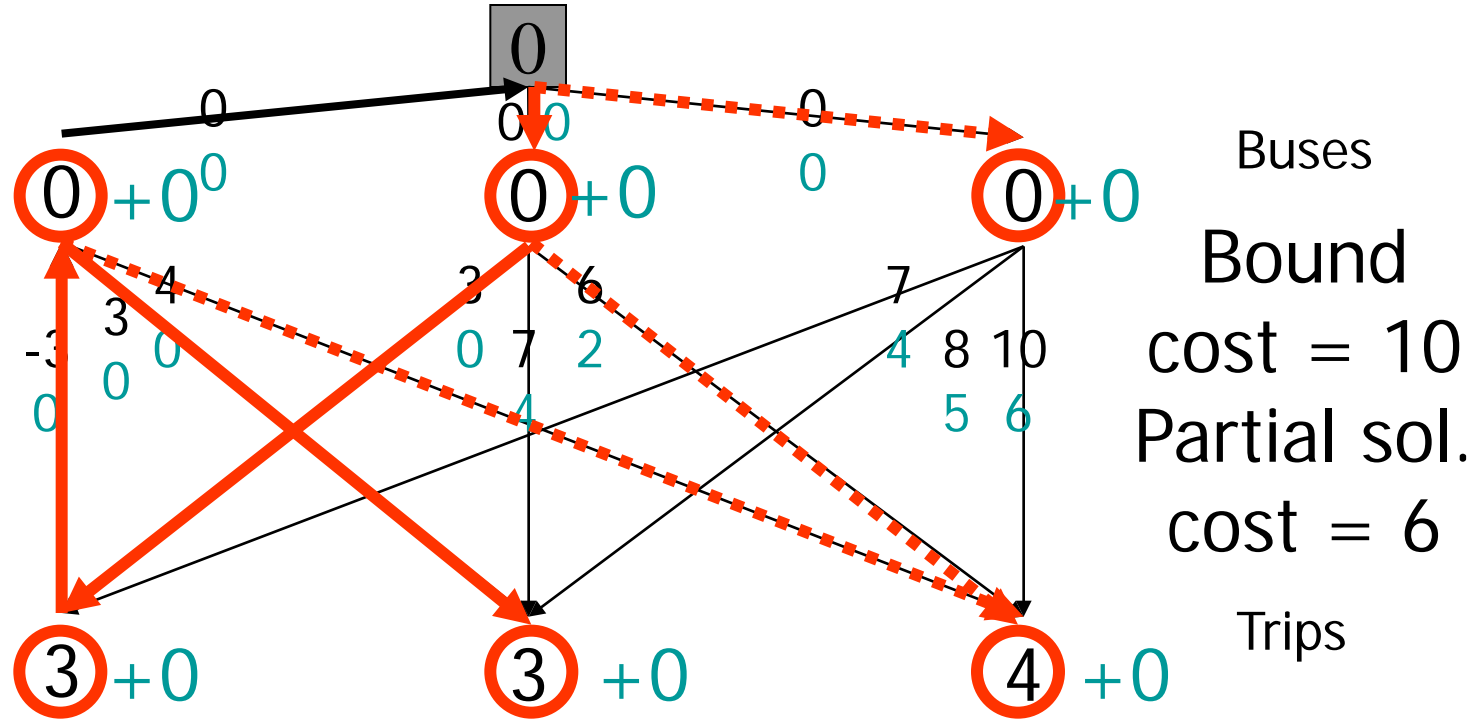
Single Depot Vehicle Scheduling



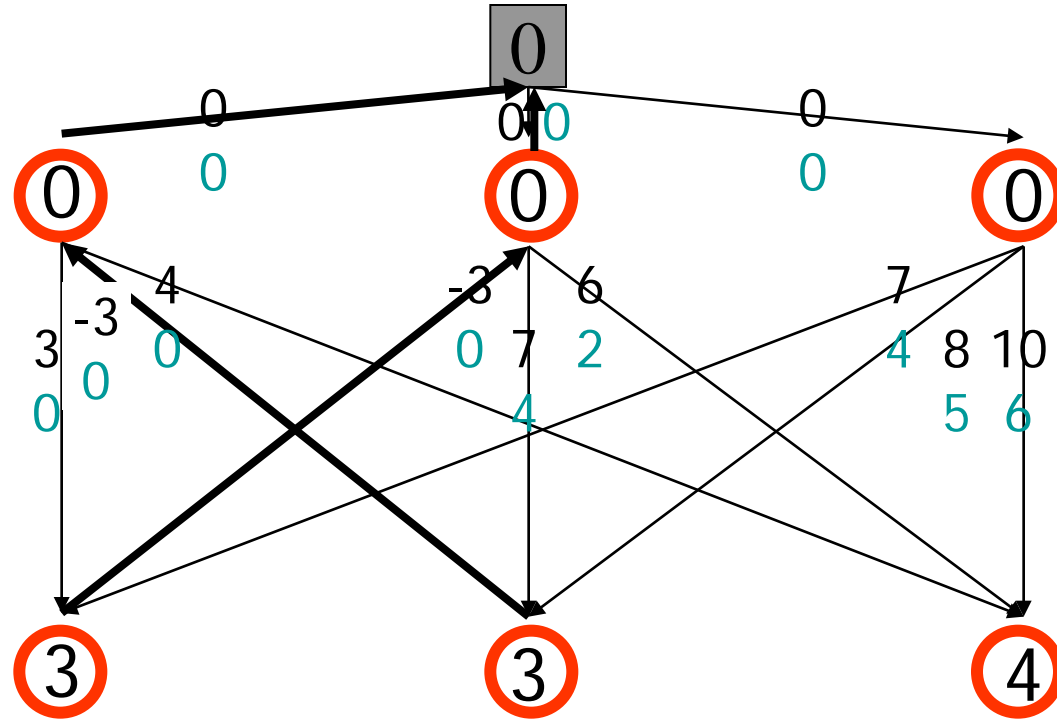
Buses
 Bound
 cost = 10
 Partial sol.
 cost = 3
 Trips

► The „Successive Shortest Path“ Algorithm

Single Depot Vehicle Scheduling



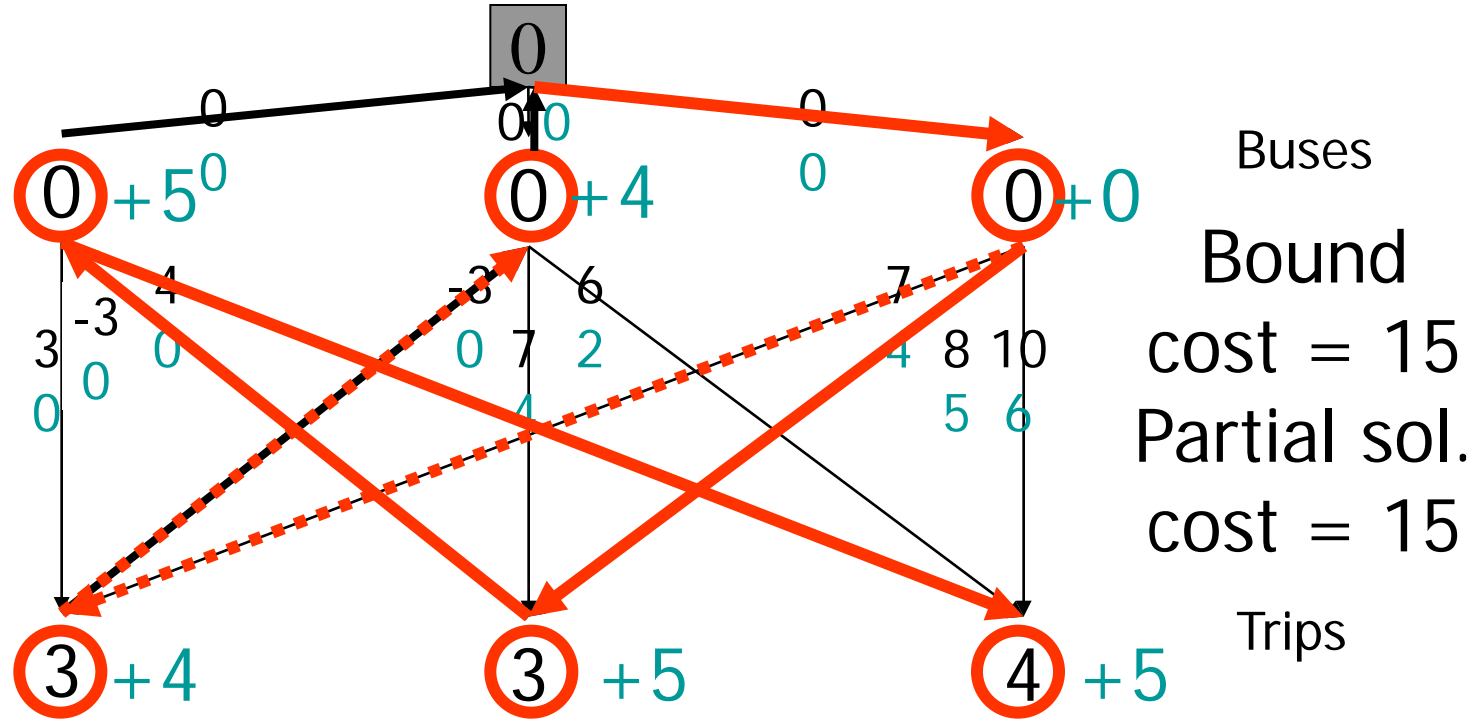
► The „Successive Shortest Path“ Algorithm



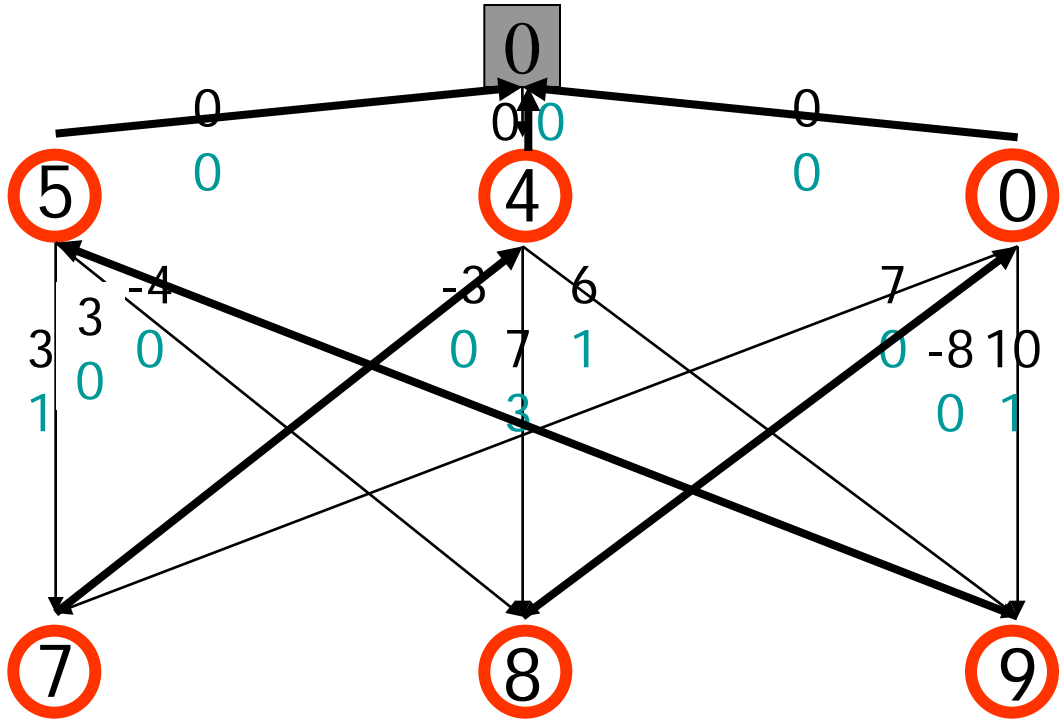
Buses
 Bound
 cost = 10
 Partial sol.
 cost = 6
 Trips

► The „Successive Shortest Path“ Algorithm

Single Depot Vehicle Scheduling

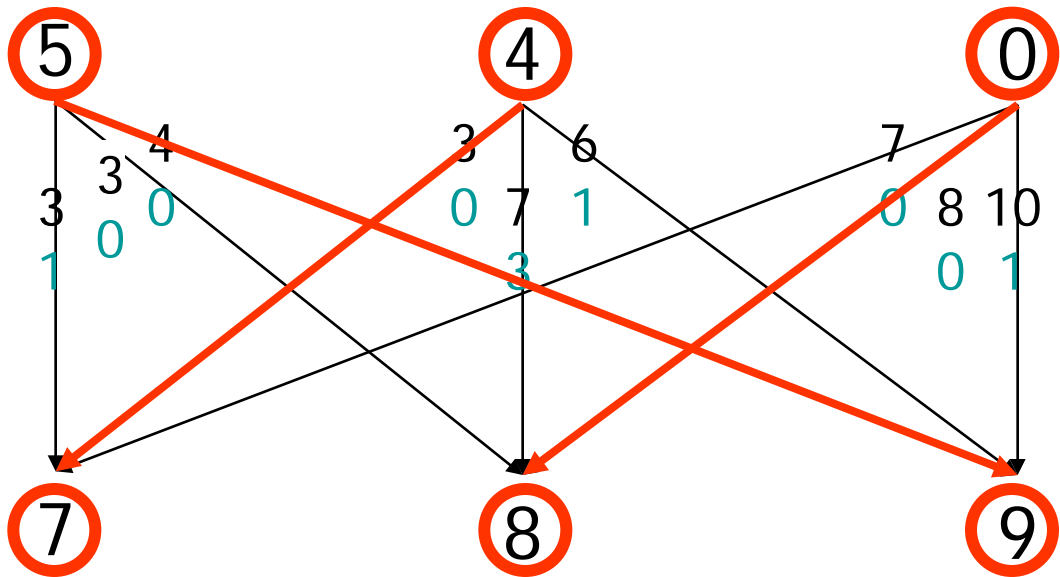


► The „Successive Shortest Path“ Algorithm



Buses
 Bound
 cost = 15
 Partial sol.
 cost = 15
 Trips

► The „Successive Shortest Path“ Algorithm

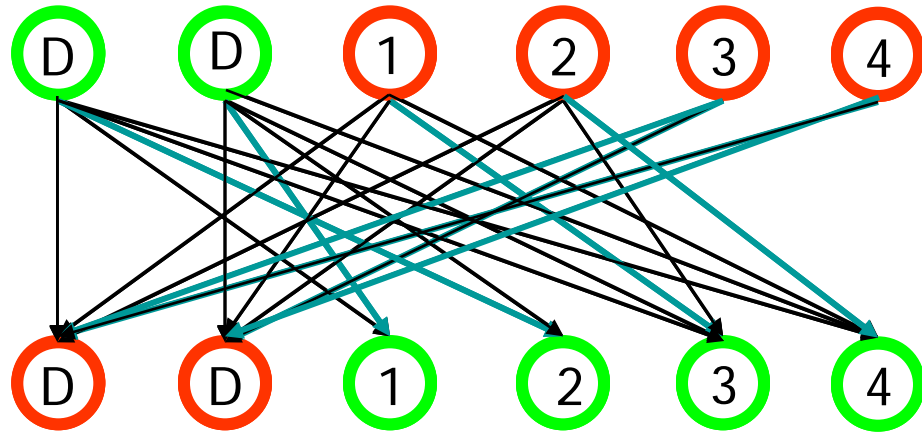
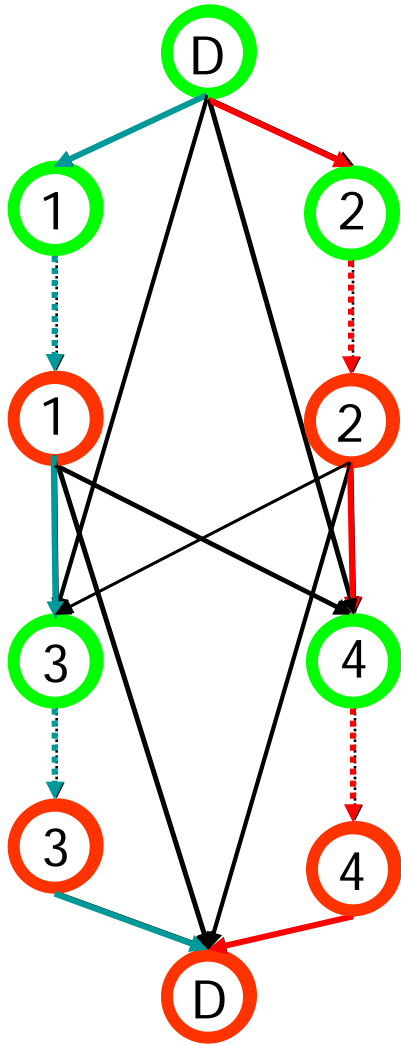


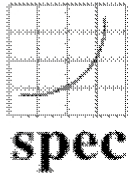
Buses
 Bound
 cost = 15
 Solution
 cost = 15
 Trips

- ▶ The „Successive Shortest Path“ Algorithm
 - ▶ Path Search
 - ▶ Solution + Proof
 - ▶ Efficient

Single Depot Vehicle Scheduling

(Assignment Model)





Standard Performance Evaluation Corporation

CINT2000 (Integer Component of SPEC CPU2000):

Benchmark	Language	Category	Full Descriptions	
164.gzip	C	Compression	HTML	Text
175.vpr	C	FPGA Circuit Placement and Routing	HTML	Text
176.gcc	C	C Programming Language Compiler	HTML	Text
181.mcf	C	Combinatorial Optimization	HTML	Text
186.crafty	C	Game Playing: Chess	HTML	Text
197.parser	C	Word Processing	HTML	Text
252.eon	C++	Computer Visualization	HTML	Text
253.perlbnk	C	PERL Programming Language	HTML	Text
254.gap	C	Group Theory, Interpreter	HTML	Text
255.vortex	C	Object-oriented Database	HTML	Text
256.bzip2	C	Compression	HTML	Text
300.twolf	C	Place and Route Simulator	HTML	Text

webmaster@spec.org

Last updated: *Fri Sep 26 11:10:06 EDT 2003*

Copyright © 1995 - 2004 Standard Performance Evaluation Corporation

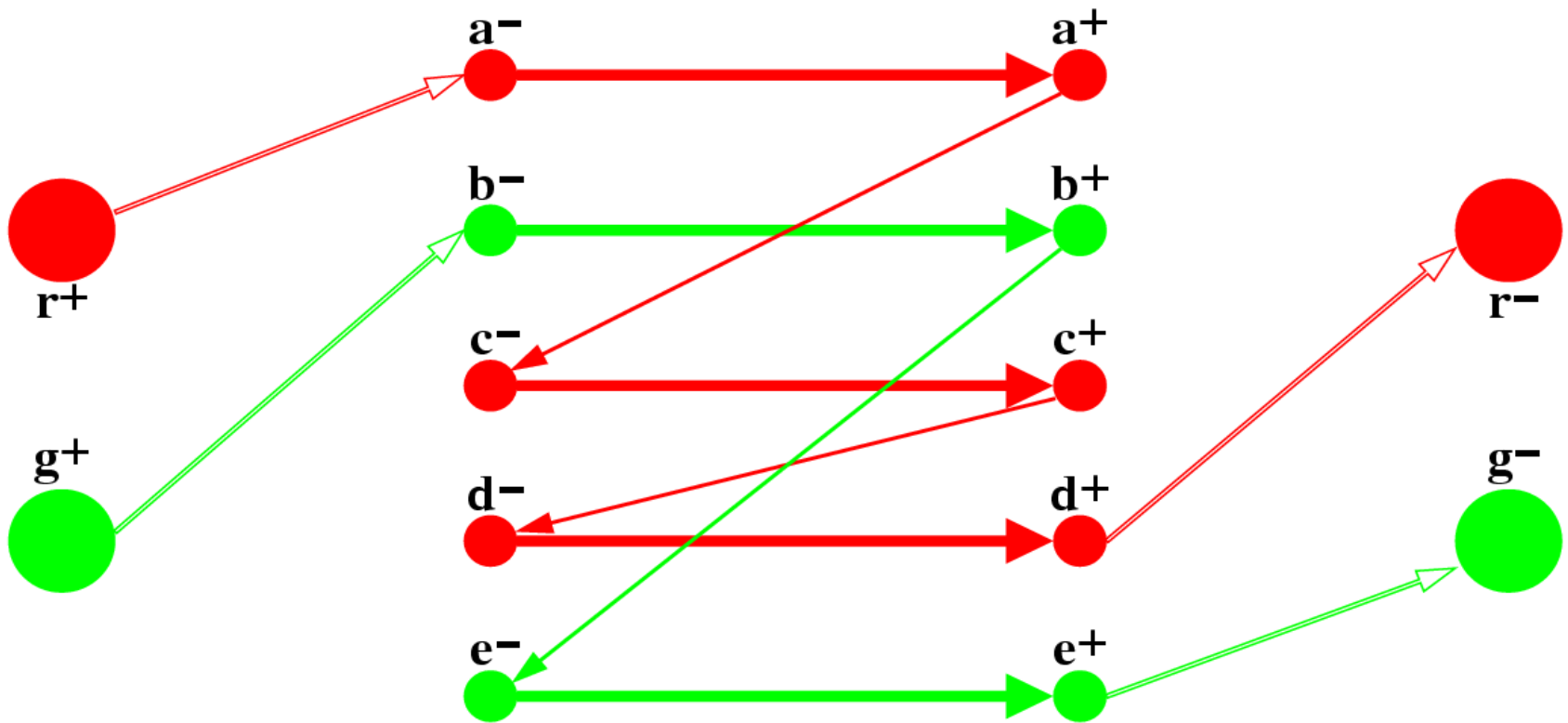
URL: <http://www.spec.org/osg/cpu2000/CINT2000/index.html>

- ▶ Introduction
- ▶ Single Depot Vehicle Scheduling
- ▶ **Multiple Depot Vehicle Scheduling**
- ▶ Lagrangean Relaxation
- ▶ Extensions
 - ▶ Trip Shifting
 - ▶ Integrated Vehicle and Duty Scheduling
 - ▶ Trip Scheduling

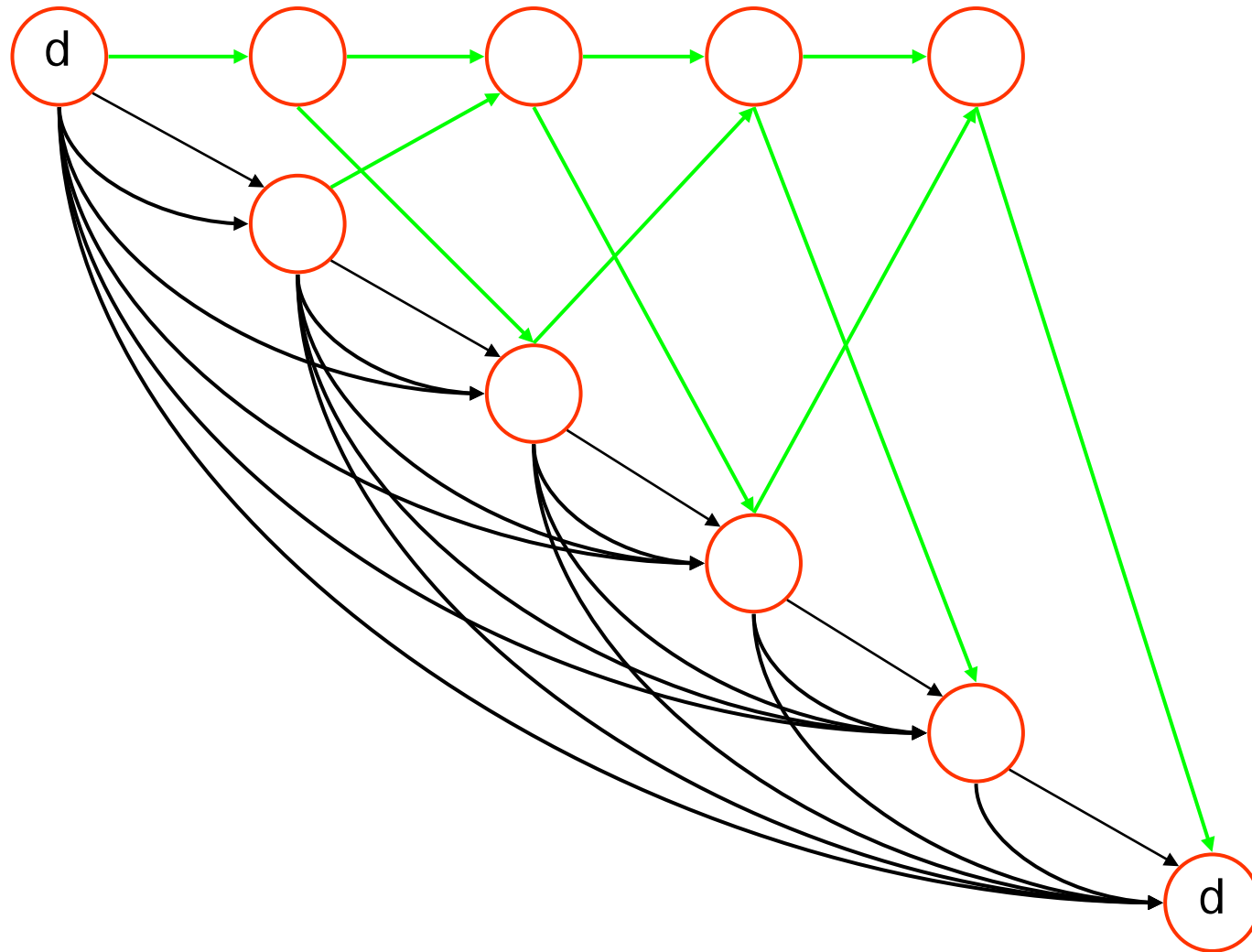
Vehicle Scheduling

- ▶ 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.

Graph Theoretic Model



Timelines



Integer Programming Model

(Multicommodity Flow Problem)

$$\begin{aligned} \min \quad & \sum_d \sum_{ij} c_{ij}^d x_{ij}^d \\ & \sum_i x_{ij}^d - \sum_k x_{jk}^d = 0 \quad \forall j, d && \text{Vehicle flow} \\ & \sum_d \sum_i x_{ij}^d - \sum_d \sum_k x_{jk}^d = 0 \quad \forall j && \text{Aggregated flow} \\ & \sum_d \sum_i x_{ij}^d = 1 \quad \forall j && \text{Timetabled trips} \\ & \sum_j x_{0j}^d \leq \kappa_d \quad \forall d && \text{Depot capacities} \\ & x_{ij}^d \in \{0,1\} \quad \forall ij, d && \text{Deadhead trips} \end{aligned}$$

- ▶ **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.

- ▶ Introduction
- ▶ Single Depot Vehicle Scheduling
- ▶ Multiple Depot Vehicle Scheduling
- ▶ Lagrangean Relaxation
- ▶ Extensions
 - ▶ Trip Shifting
 - ▶ Integrated Vehicle and Duty Scheduling
 - ▶ Trip Scheduling



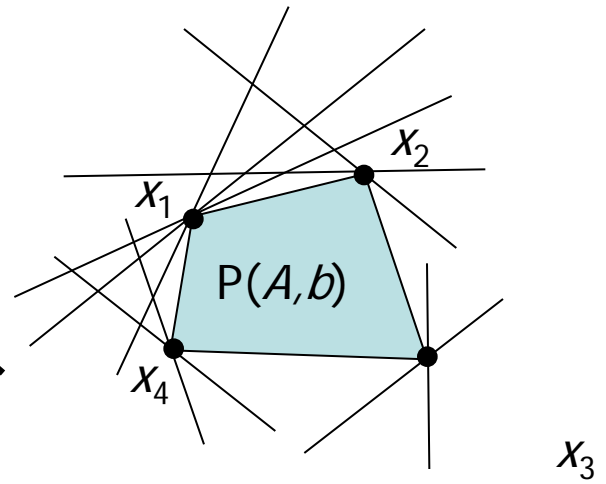
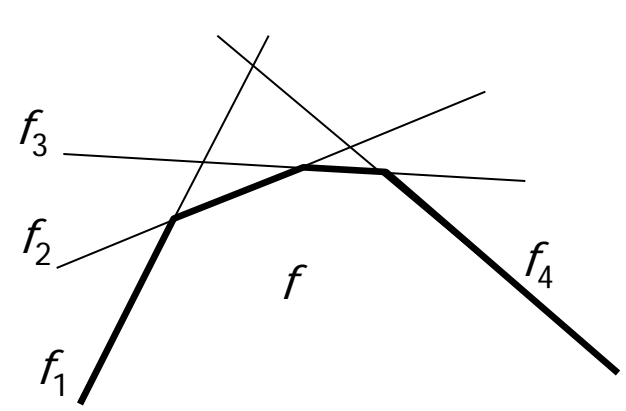
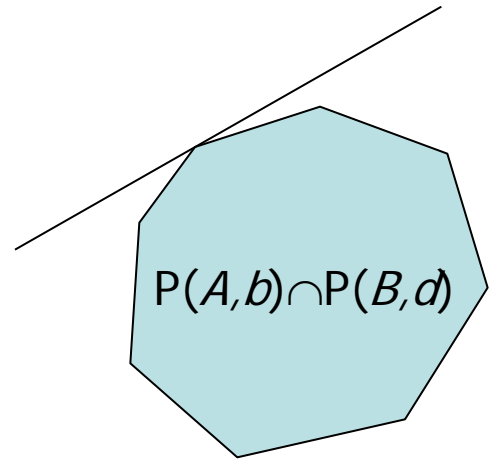
Lagrangean Relaxation

$$\begin{aligned} \min \quad & c^T x \\ & Ax = b \\ & Bx = d \\ & x \geq 0 \end{aligned}$$

$$= \max_{\lambda} \min \begin{aligned} & c^T x + \lambda(b - Ax) \\ & Bx = d \\ & x \geq 0 \end{aligned}$$

$$= \max_{\lambda} f(\lambda)$$

$$= \max_{\lambda} \min_i c^T x_i + \lambda(b - Ax_i)$$



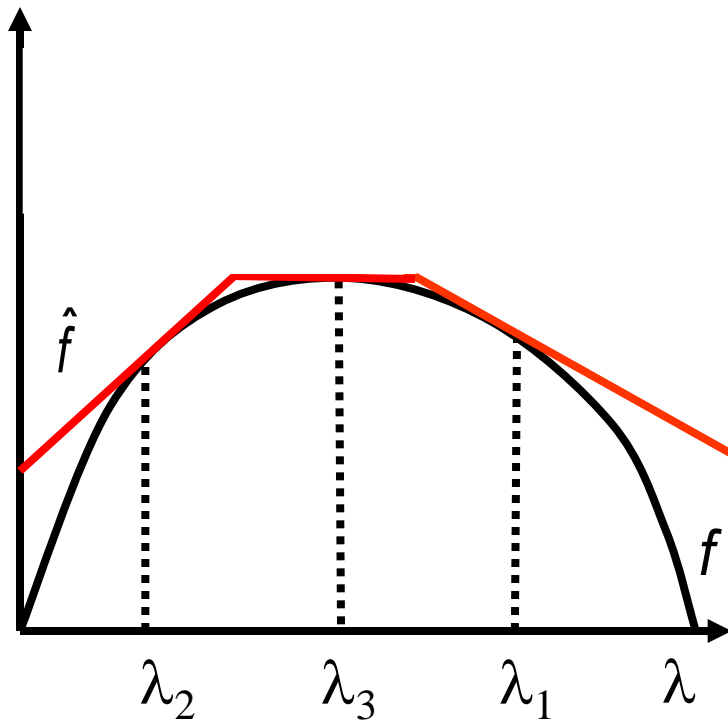


Bundle Method

(Kiwiel [1990], Helmberg [2000])

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

X polyhedral (piecewise linear)



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

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

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

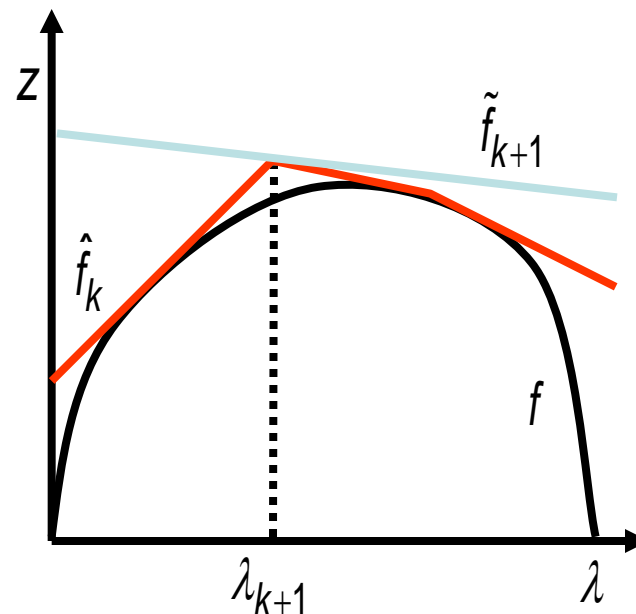


Primal Approximation

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

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

$$\tilde{f}_k(\lambda) = c^T \tilde{x}_k + \lambda(b - A\tilde{x}_k)$$



► Theorem

$$\|b - A\tilde{x}_k\| \rightarrow 0 \quad (k \rightarrow \infty)$$

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



Quadratic Subproblem

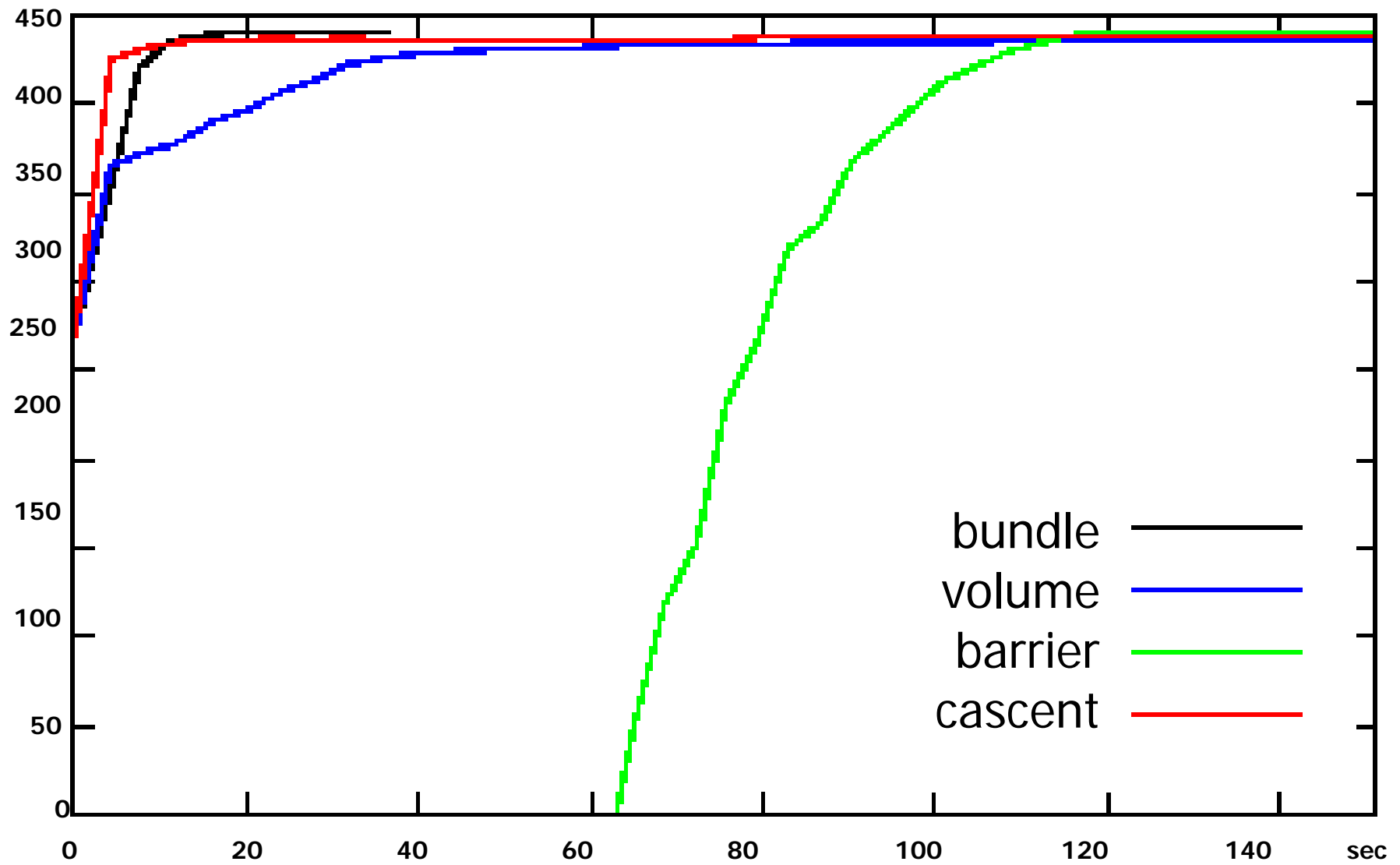
$$(1) \quad \max \hat{f}_k(\lambda) - \frac{u_k}{2} \|\lambda - \hat{\lambda}_k\|^2$$

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

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

Bundle Method

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



Lagrangean Relaxation I

$$\begin{aligned} \min \quad & \sum_d \sum_{ij} c_{ij}^d x_{ij}^d \\ & \sum_i x_{ij}^d - \sum_k x_{jk}^d = 0 \quad \forall j, d \quad \text{Vehicle flow} \\ & \sum_d \sum_i x_{ij}^d - \sum_d \sum_k x_{jk}^d = 0 \quad \forall j \quad \text{Aggregated flow} \\ & \sum_d \sum_i x_{ij}^d = 1 \quad \forall j \quad \text{Timetabled trips} \\ & \sum_j x_{0j}^d \leq \kappa_d \quad \forall d \quad \text{Depot capacities} \\ & x_{ij}^d \in \{0,1\} \quad \forall ij, d \quad \text{Deadhead trips} \end{aligned}$$

Lagrangian Relaxation I

$$\max_{\lambda} \min \sum_d \sum_{ij} c_{ij}^d x_{ij}^d + \lambda \left[0 - \left(\sum_i x_{ij}^d - \sum_k x_{jk}^d \right) \right]$$

Vehicle flow

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

Aggregated flow

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

Timetabled trips

$$\sum_j x_{0j}^d \leq \kappa_d \quad \forall d$$

Depot capacities

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

Deadhead trips

- ▶ Subproblem: Min-Cost-Flow (single-depot)

Lagrangian Relaxation II

$$\begin{aligned} \min \quad & \sum_d \sum_{ij} c_{ij}^d x_{ij}^d \\ & \sum_i x_{ij}^d - \sum_k x_{jk}^d = 0 \quad \forall j, d \quad \text{Vehicle flow} \\ & \sum_d \sum_i x_{ij}^d - \sum_d \sum_k x_{jk}^d = 0 \quad \forall j \quad \text{Aggregated flow} \\ & \sum_d \sum_i x_{ij}^d = 1 \quad \forall j \quad \text{Timetabled trips} \\ & \sum_j x_{0j}^d \leq \kappa_d \quad \forall d \quad \text{Depot capacities} \\ & x_{ij}^d \in \{0,1\} \quad \forall ij, d \quad \text{Deadhead trips} \end{aligned}$$

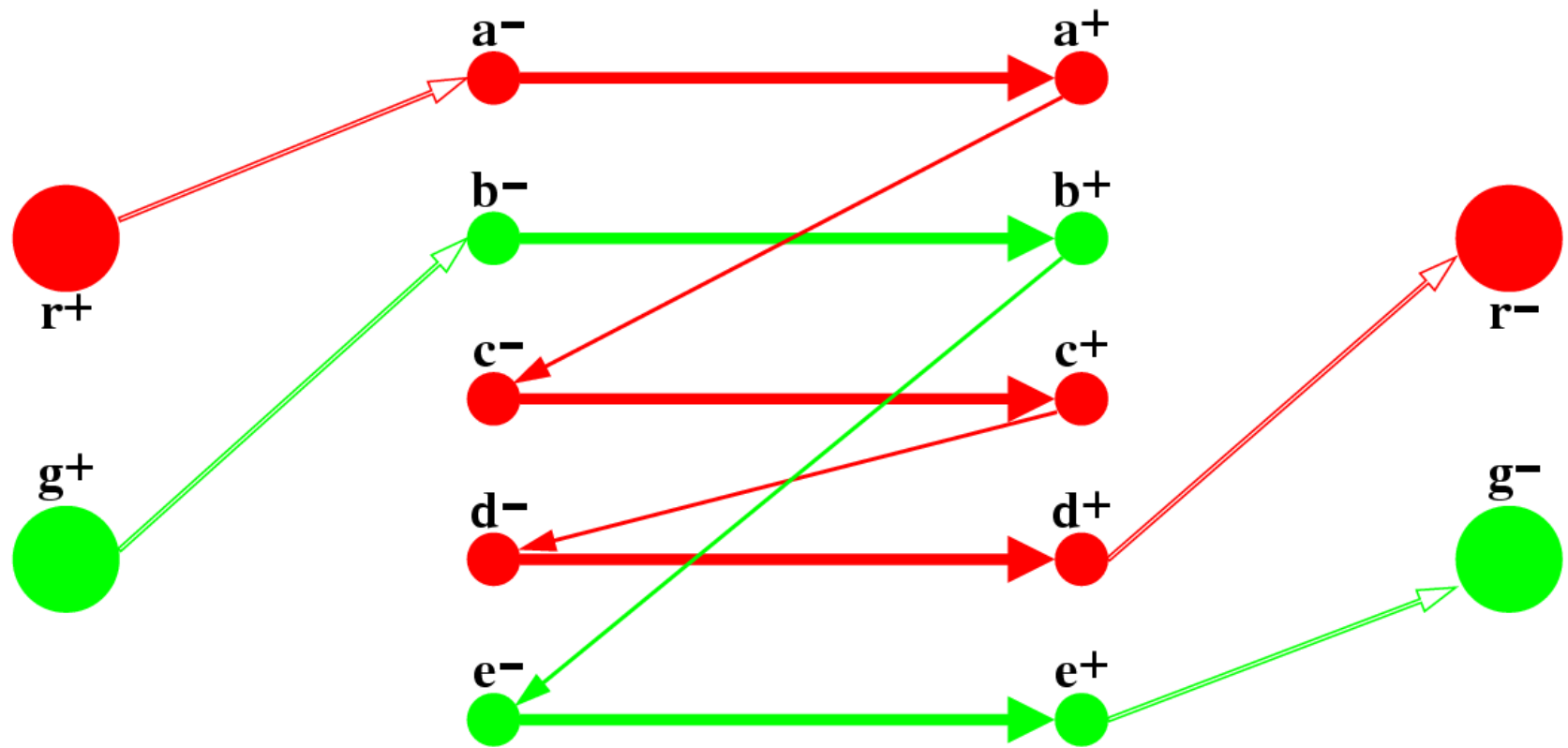
Lagrangian Relaxation II

$$\begin{aligned} \max_{\lambda} \min \quad & \sum_d \sum_{ij} c_{ij}^d x_{ij}^d + \lambda \left(1 - \sum_d \sum_i x_{ij}^d \right) \\ & \sum_i x_{ij}^d - \sum_k x_{jk}^d = 0 \quad \forall j, d && \text{Vehicle flow} \\ & \sum_d \sum_i x_{ij}^d - \sum_d \sum_k x_{jk}^d = 0 \quad \forall j && \text{Aggregated flow} \\ & \sum_j x_{0j}^d \leq \kappa_d \quad \forall d && \text{Timetabled trips} \\ & && \text{Depot capacities} \\ & x_{ij}^d \in \{0,1\} \quad \forall ij, d && \text{Deadhead trips} \end{aligned}$$

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

- ▶ 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

Lagrangian Relaxation Algorithm



Computational Results

	<i>BVG</i>	<i>HHA</i>	<i>VHH</i>
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

Erzielte Einsparungen durch die Umlaufoptimierung:



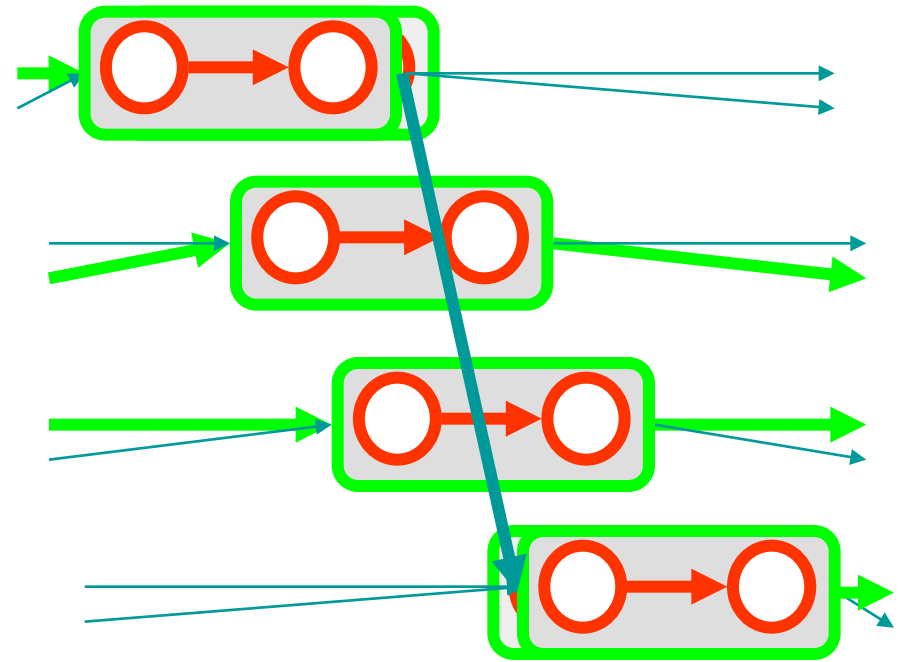
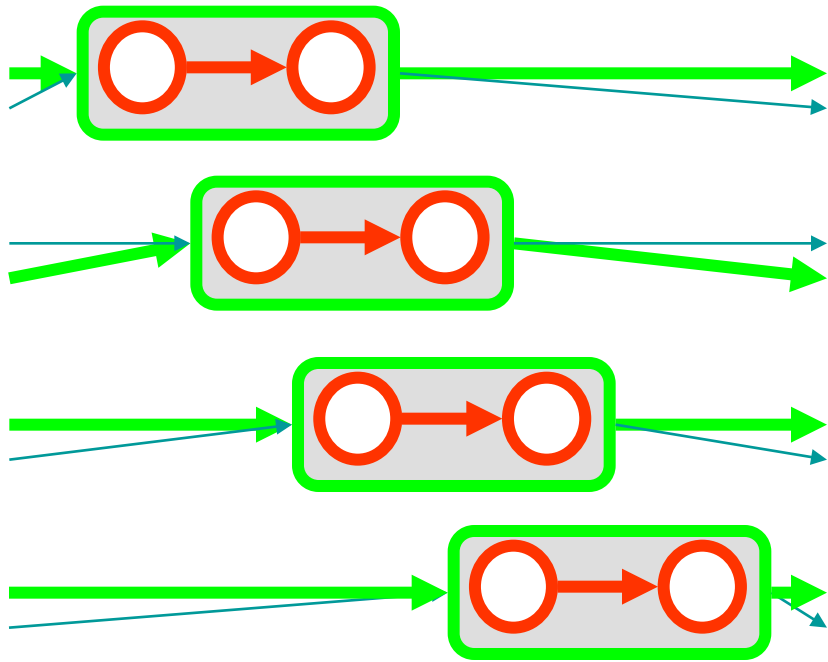
Im Busbereich wurden bei einer Gesamtzahl von knapp über 200 Fahrzeugen 5 Busse eingespart.



Im Bahnbereich wurden aufgrund der fehlenden Leerfahrt- und Überholmöglichkeiten keine Fahrzeuge eingespart.

- ▶ Introduction
- ▶ Single Depot Vehicle Scheduling
- ▶ Multiple Depot Vehicle Scheduling
- ▶ Lagrangean Relaxation
- ▶ Extensions
 - ▶ **Trip Shifting**
 - ▶ Integrated Vehicle and Duty Scheduling
 - ▶ Trip Scheduling

Trip Shifting

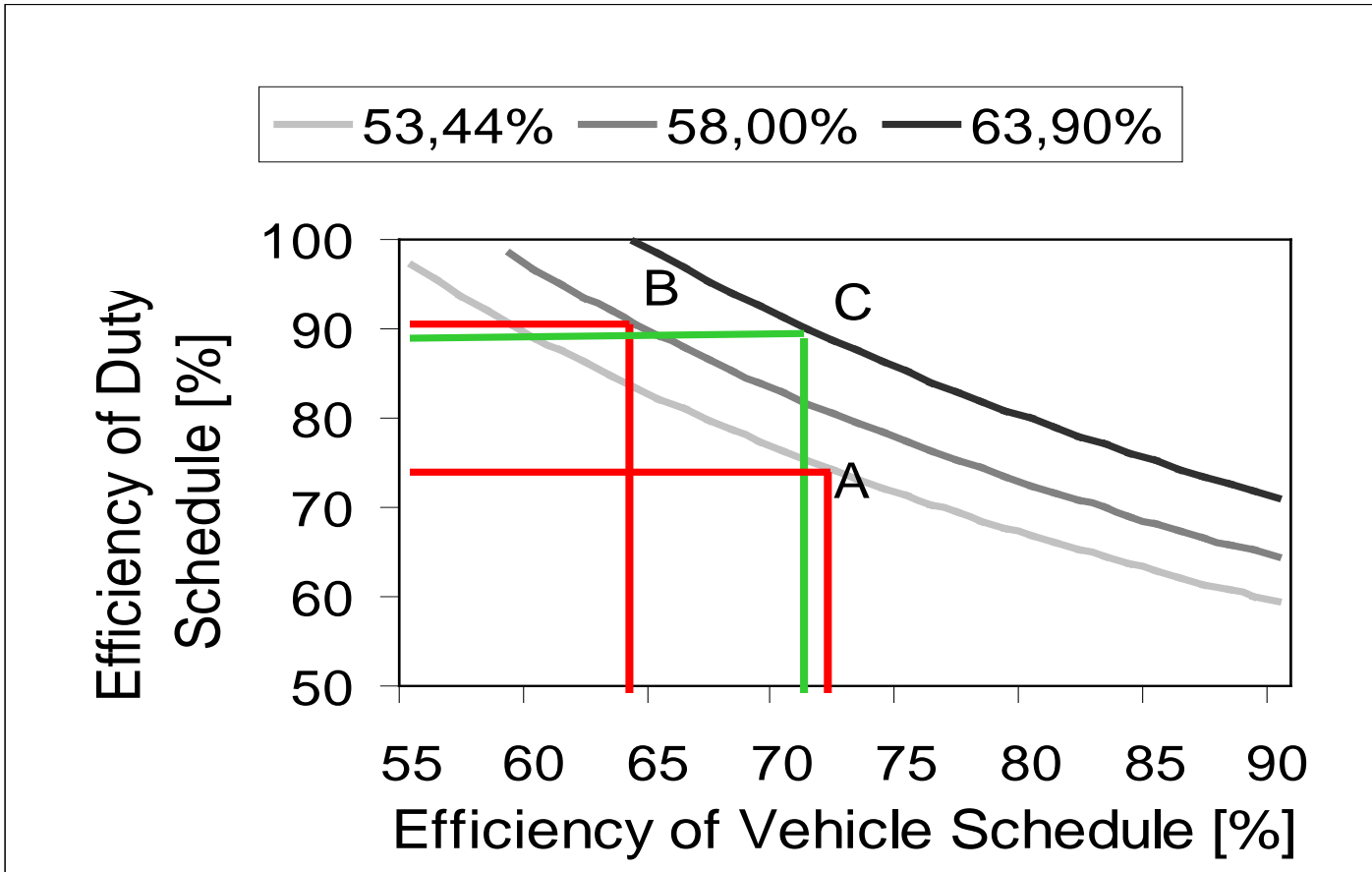


Integer Programming Model

(Multicommodity Flow Problem)

$$\begin{aligned} \min \quad & \sum_{d,\varepsilon} \sum_{ij} c_{ij\varepsilon}^d x_{ij\varepsilon}^d \\ & \sum_i x_{ij\varepsilon}^d - \sum_k x_{jk\varepsilon}^d = 0 \quad \forall j,d,\varepsilon \quad \text{Vehicle flow} \\ & \sum_{d,\varepsilon} \sum_i x_{ij\varepsilon}^d - \sum_{d,\varepsilon} \sum_k x_{jk\varepsilon}^d = 0 \quad \forall j \quad \text{Aggregated flow} \\ & \sum_{d,\varepsilon} \sum_i x_{ij}^d = 1 \quad \forall j \quad \text{Timetabled trips} \\ & \sum_{\varepsilon,j} x_{0j\varepsilon}^d \leq \kappa_d \quad \forall d \quad \text{Depot capacities} \\ & x_{ij\varepsilon}^d \in \{0,1\} \quad \forall ij,d,\varepsilon \quad \text{Deadhead trips} \end{aligned}$$

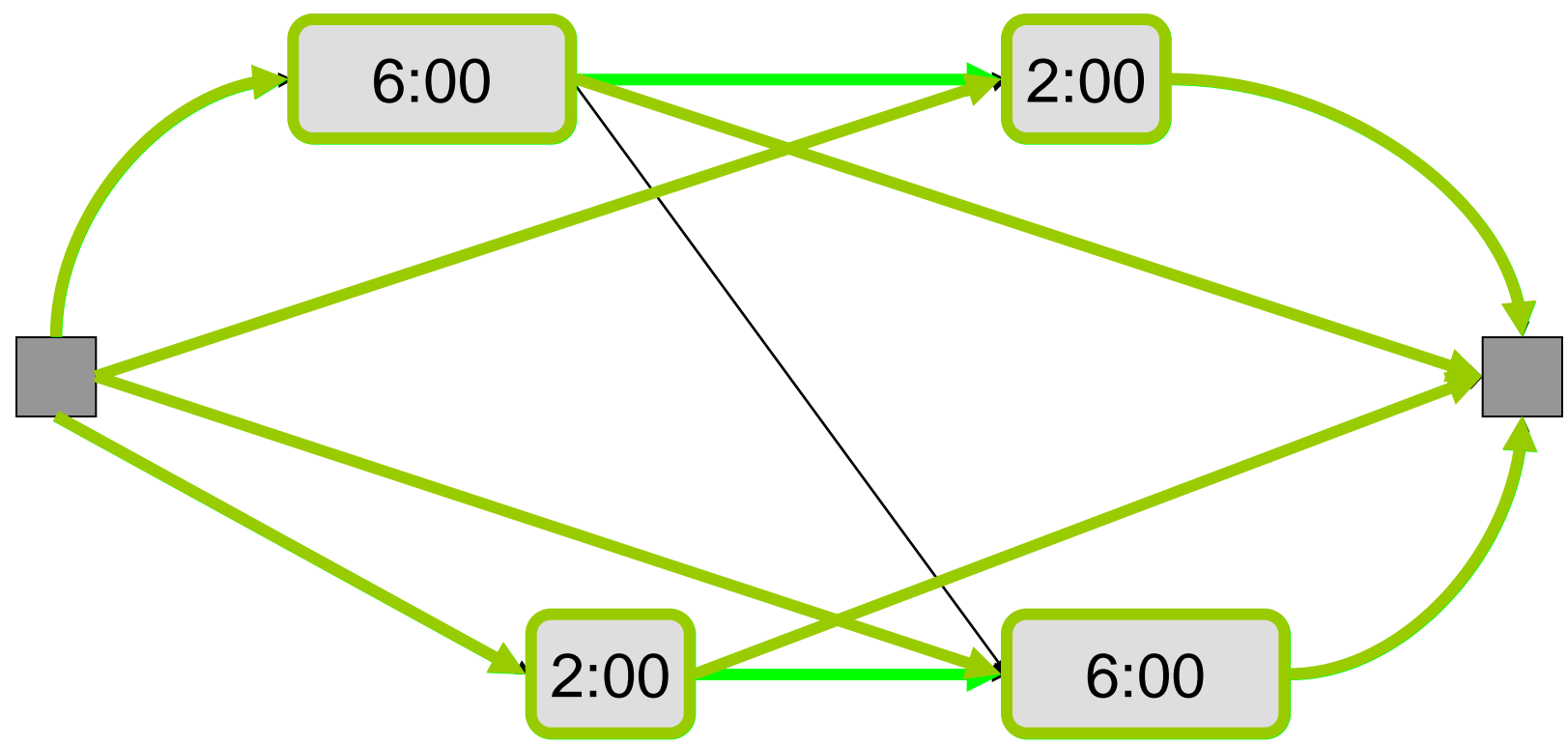
- ▶ Introduction
- ▶ Single Depot Vehicle Scheduling
- ▶ Multiple Depot Vehicle Scheduling
- ▶ Lagrangean Relaxation
- ▶ Extensions
 - ▶ Trip Shifting
 - ▶ **Integrated Vehicle and Duty Scheduling**
 - ▶ Trip Scheduling



- (A) Sequential Scheduling
- (B) Duty Oriented Vehicle Scheduling
- (C) Integrated Scheduling



Regional Scenarios



hh:mm timetabled trip \longrightarrow deadhead trip (1:00)

▶ Max. duty time 8:00

Integer Programming Model

$$\begin{aligned} \text{(ISP) min} \quad & c^T x + d^T y \\ \text{(1a)} \quad & N^D x = 0, \quad \forall \text{ depots } D \\ \text{(1b)} \quad & x(\delta^{\text{out}}(v)) = 1, \quad \forall v \in V \setminus \{t\} \\ \text{(1c)} \quad & x(\delta^{\text{in}}(v)) = 1, \quad \forall v \in V \setminus \{s\} \\ \text{(2a)} \quad & Ay = 1 \\ \text{(2b)} \quad & By \leq b \\ \text{(3)} \quad & Cx - Dy = 0 \\ & x \in \{0, 1\}^m, \quad y \in \{0, 1\}^n \end{aligned}$$

$$C_{dt} := \begin{cases} 1, & \text{if deadhead } d \text{ covers duty element } t \\ 0, & \text{else} \end{cases}$$

$$D_{dt} := \begin{cases} 1, & \text{if duty } d \text{ covers duty elements } t \\ 0, & \text{else} \end{cases}$$



Lagrangian Relaxation

$$\begin{aligned}
 & f(\lambda) := \\
 \max_{\lambda} & \underbrace{\min_{\substack{x \text{ fulfills (1) and} \\ x \in \{0,1\}^m}} (c^T - \lambda^T C)x}_{=: f_V(\lambda)} + \underbrace{\max_{\substack{y \text{ fulfills (2) and} \\ y \in [0,1]^n}} (d^T - \lambda^T D)y}_{=: f_D(\lambda)}
 \end{aligned}$$

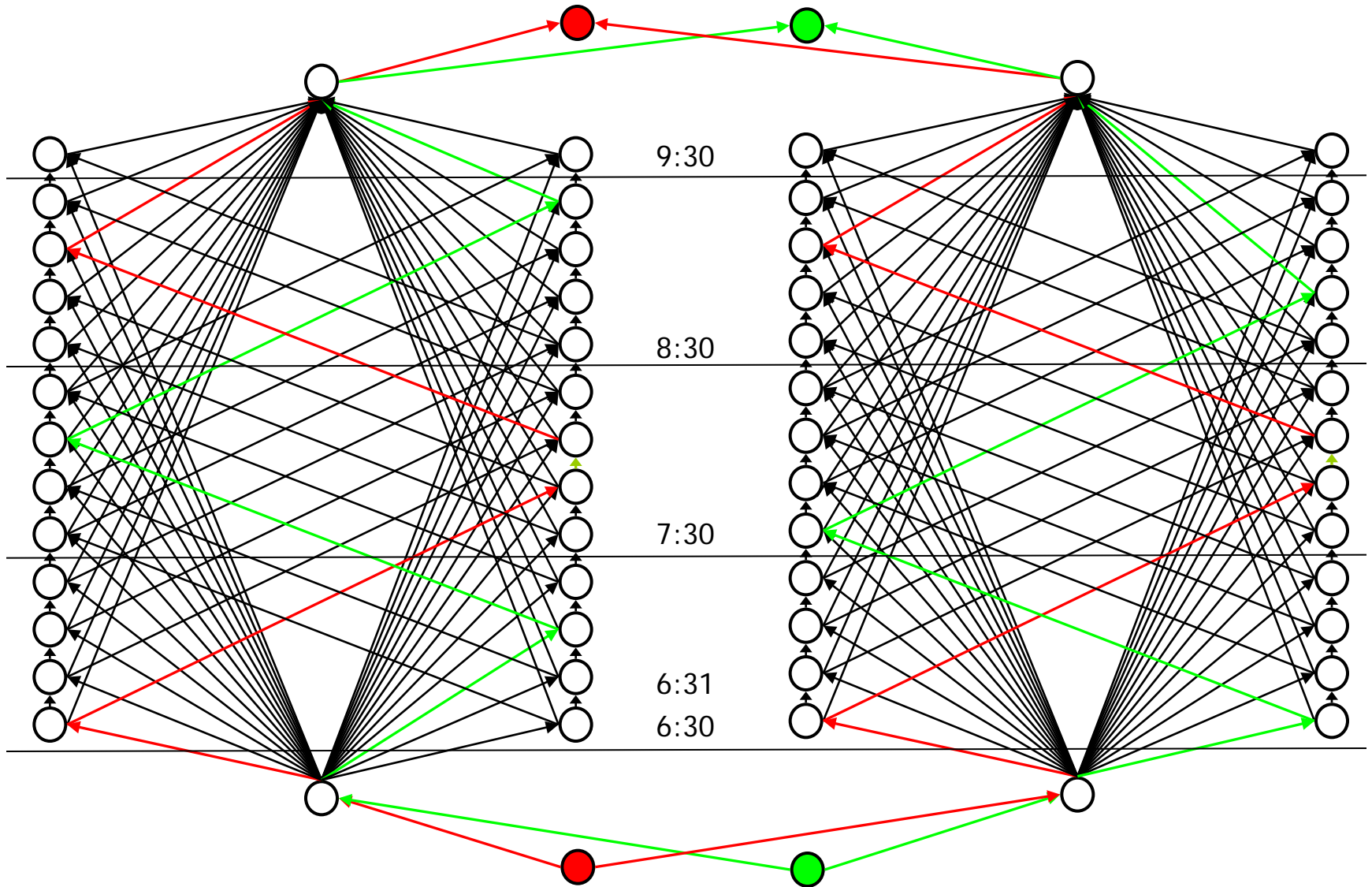
- ▶ Dual Problem: Find multipliers λ maximizing f
- ▶ Primal Problem: Find fractional solution $(x, y) \in [0, 1]^{m \times n}$ fulfilling (1), (2), (3)
- ▶ Note: Integrality relaxed

- ▶ Introduction
- ▶ Single Depot Vehicle Scheduling
- ▶ Multiple Depot Vehicle Scheduling
- ▶ Lagrangean Relaxation
- ▶ Extensions
 - ▶ Trip Shifting
 - ▶ Integrated Vehicle and Duty Scheduling
 - ▶ **Trip Scheduling**

Trip Scheduling

- ▶ Input
 - Lines with lengths and speed profiles
 - Passenger volume, frequency, and regularity profiles
 - Vehicle types with speeds and capacities
 - Depot capacities
- ▶ Output
 - Trips and vehicle rotations for all lines
- ▶ Problem
 - Schedule trips and vehicle rotations to transport passengers
- ▶ Goals
 - Maximize trip regularity
 - Minimize number of passengers left behind
 - Minimize number of vehicles

Graph Theoretic Model



Integer Programming Model

$$\begin{array}{llll} \min & \sum_d \sum_{0j} x_{0j}^d & & \\ & \sum_i x_{ij}^d - \sum_k x_{jk}^d = 0 & \forall j, d & \text{Vehicle flow} \\ & \sum_d \sum_i x_{ij}^d \leq 1 & \forall j & \text{Timetabled trips} \\ & \sum_d \sum_{ij \in H_\ell} x_{ij}^d \geq f_\ell & \forall H_\ell & \text{Frequencies} \\ & \sum_d \sum_{ij \in H_\ell} \kappa^d x_{ij}^d \geq d_\ell & \forall H_\ell & \text{Pax volume} \\ & \sum_j x_{0j}^d \leq \kappa_d & \forall d & \text{Depot capacities} \\ & x_{ij}^d \in \{0,1\} & \forall ij, d & \text{Deadhead trips} \end{array}$$

Computational Results

	1	2
lines	11	38
vehicle types	2	2
frequency	12	38
hours	3	3
pax/line/hour	> 1,000	> 1,000
vehicles	~ 300	~ 700

Thank you for your attention.