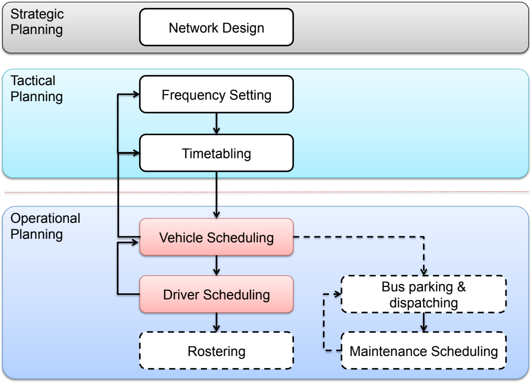
Vehicle Scheduling: Models and Algorithms

Stefano Gualandi
Università di Pavia, Dipartimento di Matematica

email: stefano.gualandi@unipv.it
twitter: @famo2spaghi
blog: http://stegua.github.com

Overview of Planning Activities (Desaulniers&Hickman2007)



Strategic Planning: Network Design (Urban)

Strategic Planning: Network Design (Regional)

Tactical Planning: Frequency Setting and Timetabling

41 **42**

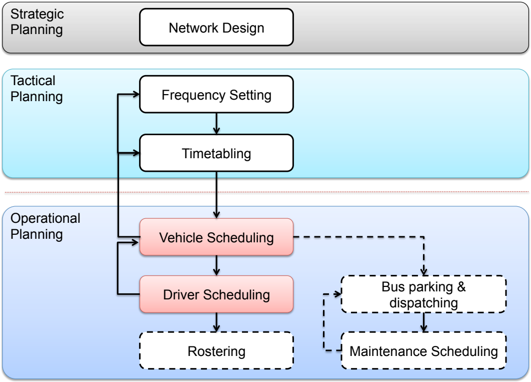
Odense Banegård - Syddansk Universitet (SDU)

Hverdag

Rutenr.	OBC Plads B	Hans Mules Gade	Nyborgvej / Frederiksgade	Palnatokesvej	Nansensgade	Ejbygade	L.A. Rings Vej	Rosengårdc. / Gul indg.	Bilka	Rosengårdc. / Ørbækvej	Niels Bohrs Allé	Campusvej	SDU
42	**05.25**	05.26	05.28	05.29	05.30	05.31	05.33			05.34	05.36	05.37	**05.40**
41	**06.23**	06.24	06.26	06.27	06.28	06.29	06.31	06.33	06.35			06.37	**06.40**
41	**07.20**	07.21	07.23	07.24	07.25	07.26	07.29	07.31	07.34			07.36	**07.40**
42	**07.28**	07.30	07.32	07.33	07.34	07.35	07.39			07.41	07.44	07.46	**07.50**
42	**07.36**	07.38	07.40	07.41	07.42	07.44	07.46			07.48	07.51	07.53	**07.56**
42	**07.44**	07.46	07.48	07.49	07.50	07.51	07.55			07.57	08.00	08.02	**08.06**
42	**07.52**	07.54	07.56	07.56	07.57	07.58	08.01			08.03	08.06	08.07	**08.12**
151P	**08.00**	08.02	08.04	08.04	08.05	08.06	08.09			08.11	08.14	08.15	**08.20**
41	**08.10**	08.12	08.14	08.15	08.16	08.18	08.21	08.23	08.27			08.29	**08.33**
41	**08.25**	08.27	08.29	08.30	08.31	08.32	08.35	08.37	08.41			08.43	**08.47**
41	**08.35**	08.37	08.39	08.40	08.41	08.42	08.45	08.47	08.51			08.53	**08.57**
41	**08.45**	08.47	08.49	08.50	08.51	08.52	08.55	08.57	09.01			09.03	**09.07**
41	**08.55**	08.57	08.59	09.00	09.01	09.02	09.05	09.07	09.11			09.13	**09.17**
41	**09.05**	09.07	09.09	09.10	09.11	09.12	09.15	09.17	09.21			09.23	**09.27**
41	**09.15**	09.17	09.19	09.20	09.21	09.22	09.25	09.27	09.31			09.33	**09.37**

OBS:
Ikke gyldig mellem 13. maj - 23. august. Se sommerkøreplanen på side 70-71.

Overview of Planning Activities (Desaulniers&Hickman2007)



Leuthardt Survey

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

bus costs (DM)	*urban*	%	*regional*	%
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

www.thequestforoptimality.com/smart-models-start-small/

Home   About Me & This Blog

the quest for optimality

Using solvers & heuristics to solve complex problems

HOME › MODELING › SMART MODELS START SMALL

Smart models start small

Posted on **SEPTEMBER 9, 2013** *Written by* **MARC-ANDRE** **LEAVE A COMMENT**

There is only one good way to build large-size or complex optimization models: to start by a small model and adding elements gradually until you get the model you wanted in the first place. I have seen so many people (including myself) try to build large-size, complex models from scratch, only to spend countless frustrating hours trying to debug all kinds of problems. It just doesn't work.

A better approach is to start with the simplest version of the model. On or two

Vehicle Scheduling

Given a timetable as a set $V = \{v_1, \ldots, v_n\}$ of **trips**, where for each trip v_i we have:

t_i : departure time

a_i : arrival time

o_i : origin (departure terminal)

d_i : destination (arrival terminal)

v_i	t_i	a_i	o_i	d_i
v_1	7:10	7:30	T_a	T_b
v_2	7:20	7:40	T_c	T_d
v_3	7:40	8:05	T_b	T_a
v_4	8:00	8:30	T_d	T_c
v_5	8:35	9:05	T_c	T_d

Given the **deadheading trips** (i.e. trips without passengers) of duration h_{ij} between every pair of terminals

h_{ij}	T_a	T_b	T_c	T_d
T_a	0	15	20	20
T_b	15	0	25	10
T_c	20	25	0	15
T_d	20	10	15	0

Definition (Compatible Trips)

A pair of trips (v_i, v_j) is compatible if and only if $a_i + h_{ij} \leq t_j$.

Vehicle Scheduling

Definition (Vehicle Duty)

A subset $C = \{v_{i_1}, \ldots, v_{i_k}\}$ of V is a **vehicle duty (or block)** if $(v_{i_j}, v_{i_{(j+1)}})$ is a compatible pair of trips, for $j = 1, \ldots, k-1$

Definition (Vehicle Schedule)

A collection $C_1, \ldots, C_r$ of *vehicle duties* such that each trip v in V belongs to exactly one C_j with $j \in \{1, \ldots, r\}$ is said to be a **Vehicle Schedule**

Vehicle Scheduling: Example

v_i	t_i	a_i	o_i	d_i
v_1	7:10	7:30	T_a	T_b
v_2	7:20	7:40	T_c	T_d
v_3	7:40	8:05	T_b	T_a
v_4	8:00	8:30	T_d	T_c
v_5	8:35	9:05	T_c	T_d

h_{ij}	T_a	T_b	T_c	T_d
T_a	0	15	20	20
T_b	15	0	25	10
T_c	20	25	0	15
T_d	20	10	15	0

Example: These 5 trips can be scheduled with 2 vehicle duties:

- $C_1 = \{v_1, v_3\}$
- $C_2 = \{v_2, v_4, v_5\}$

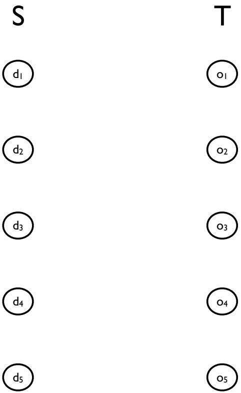
Further Features of the Problem

- Limited number of vehicles
- Minimize fleet size (number of vehicles)
- Minimize operational costs (given by pull-out and pull-in from depots and deadheading trips)
- Multiple depots
- Different types of vehicles with different operational costs located at a single depot

Vehicle Scheduling and Matchings

We build a complete bipartite graph $G = (S, T, A_1 \cup A_2)$

- $S = \{d_1, \ldots, d_n\}$: a node for each **arrival terminal**
- $T = \{o_1, \ldots, o_n\}$: a node for each **departure terminal**



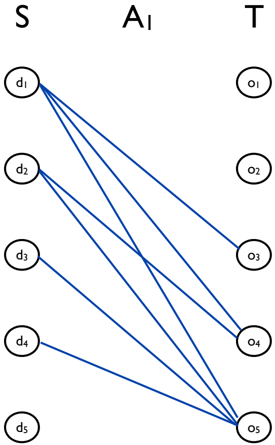
Vehicle Scheduling and Matchings

We build a complete bipartite graph $G = (S, T, A_1 \cup A_2)$

- $A_1 = \{(d_i, o_j) \mid (v_i, v_j) \text{ is a compatible pair of trips}\}$

v_i	t_i	a_i	o_i	d_i
v_1	7:10	7:30	T_a	T_b
v_2	7:20	7:40	T_c	T_d
v_3	7:40	8:05	T_b	T_a
v_4	8:00	8:30	T_d	T_c
v_5	8:35	9:05	T_c	T_d

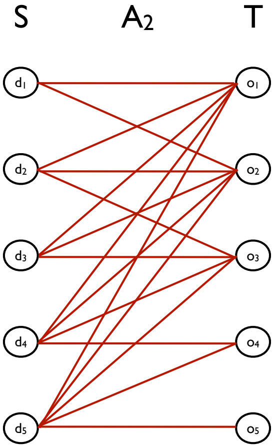
h_{ij}	T_a	T_b	T_c	T_d
T_a	0	15	20	20
T_b	15	0	25	10
T_c	20	25	0	15
T_d	20	10	15	0



Vehicle Scheduling and Matchings

- $A_2 = A \setminus A_1$, where each $(d_i, o_j) \in A_2$ corresponds to
  1. **pull-out**: deadheading trip from d_i to the depot
  2. **pull-in**: deadheading trip from the depot to o_j

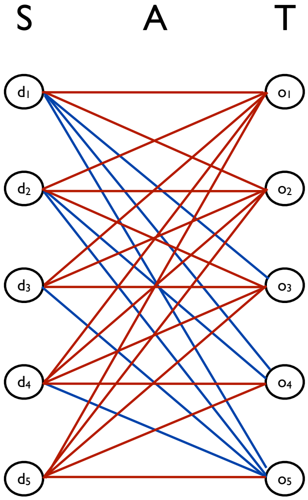
v_i	t_i	a_i	o_i	d_i
v_1	7:10	7:30	T_a	T_b
v_2	7:20	7:40	T_c	T_d
v_3	7:40	8:05	T_b	T_a
v_4	8:00	8:30	T_d	T_c
v_5	8:35	9:05	T_c	T_d



Single Depot VS: Matching

Complete bipartite graph

v_i	t_i	a_i	o_i	d_i
v_1	7:10	7:30	T_a	T_b
v_2	7:20	7:40	T_c	T_d
v_3	7:40	8:05	T_b	T_a
v_4	8:00	8:30	T_d	T_c
v_5	8:35	9:05	T_c	T_d

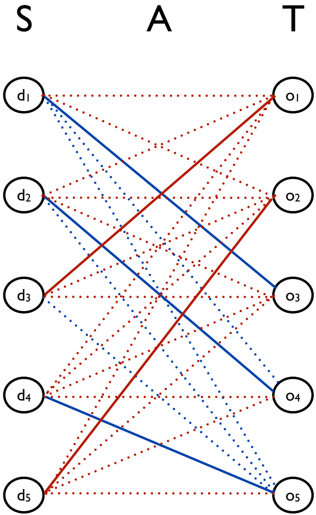


Single Depot VS: Matching

Example of solution:

- $C_1 = \{v_1, v_3\}$
- $C_2 = \{v_2, v_4, v_5\}$

v_i	t_i	a_i	o_i	d_i
v_1	7:10	7:30	T_a	T_b
v_2	7:20	7:40	T_c	T_d
v_3	7:40	8:05	T_b	T_a
v_4	8:00	8:30	T_d	T_c
v_5	8:35	9:05	T_c	T_d



Single Depot VS and Integer Linear Programming

Integer Linear Programming formulation:

$$\min \quad \sum_{ij \in A} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in S} x_{ij} = 1 \qquad \forall j \in T \tag{2}$$

$$\sum_{j \in T} x_{ij} = 1 \qquad \forall i \in S \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A. \tag{4}$$

To **minimize the fleet size** we set:

1. $c_{ij} = 0$ for each $(i, j) \in A_1$
2. $c_{ij} = 1$ for each $(i, j) \in A_2$

Single Depot VS and Integer Linear Programming

Integer Linear Programming formulation:

$$\min \quad \sum_{ij \in A} c_{ij} x_{ij} \tag{5}$$

$$\text{s.t.} \quad \sum_{i \in S} x_{ij} = 1 \qquad \forall j \in T \tag{6}$$

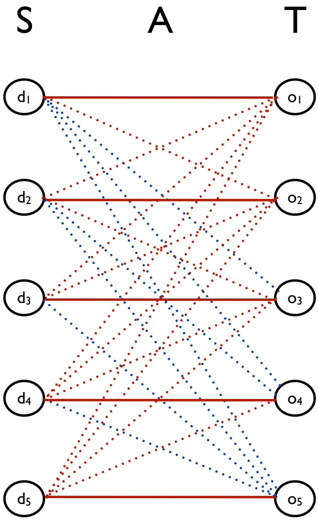
$$\sum_{j \in T} x_{ij} = 1 \qquad \forall i \in S \tag{7}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{8}$$

To **minimize the operational costs** we set:

1. if $(i,j) \in A_1$, c_{ij} is the deadheading costs from d_i to o_j plus the idle time cost before the starting of v_j
2. if $(i,j) \in A_2$, c_{ij} is the sum of the pull-out and pull-in costs

Question: with very high idle time costs?



Single Depot VS: Questions?

What if the number of vehicles is limited?

How can we modify the ILP formulation?

How can we modify the Assignment formulation?

Single Depot VS: Capacitated Matching

Integer Linear Programming formulation:

$$\min \quad \sum_{ij \in A} c_{ij} x_{ij} \tag{9}$$

$$\text{s.t.} \quad \sum_{i \in S} x_{ij} = 1 \qquad \forall j \in T \tag{10}$$

$$\sum_{j \in T} x_{ij} = 1 \qquad \forall i \in S \tag{11}$$

$$\sum_{ij \in A_2} x_{ij} \leq k \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A. \tag{13}$$

How can we modify the Assignment formulation?

(Recall) Minimum Cost Flow Problem

Given a directed graph $G = (N, A)$, where

- each node i has a **flow balance** parameter b_i (if $b_i > 0$ is a source node, if $b_i < 0$ sink node, if $b_i = 0$ transhipment node)
- each arc (i, j) has a **non negative cost** c_{ij}
- each arc (i, j) has a **non negative capacity** u_{ij}

the problem of finding a *feasible* flow f_{ij} on each arc that respects the node flow balances and the arc capacities, and which minimize the summation $\sum_{ij \in A} c_{ij} f_{ij}$, is called the

Minimum Cost Flow Problem

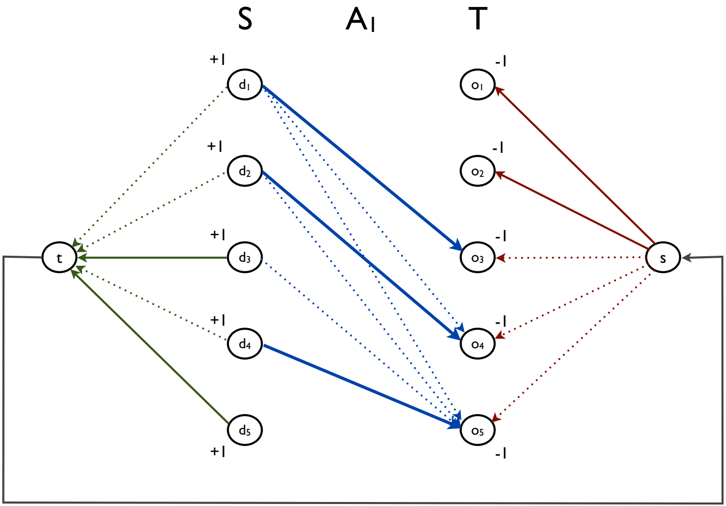
Min Cost Flow: Computational Complexity

Good news: Min Cost Flow is **Polynomially Solvable!**

$O(nU \cdot SP_+(n, m))$	Edmonds and Karp [24]; Tomizawa [70] *successive shortest path*
$O(m \log U \cdot SP_+(n, m))$	Edmonds and Karp [24] *capacity-scaling*
$O(m \log n \cdot SP_+(n, m))$	Orlin [60] *enhanced capacity-scaling*
$O(nm \log(n^2/m) \log(nC))$	Goldberg and Tarjan [38] *generalized cost-scaling*
$O(nm \log \log U \log(nC))$	Ahuja, Goldberg, Orlin, and Tarjan [1] *double scaling*
$O((m^{3/2}U^{1/2} + mU \log(mU)) \log(nC))$	Gabow and Tarjan [30]
$O((nm + mU \log(mU)) \log(nC))$	Gabow and Tarjan [30]

Table 1: Best theoretical running time bounds for the MCF problem

Capacitated Matching: Min Cost Flow Formulation

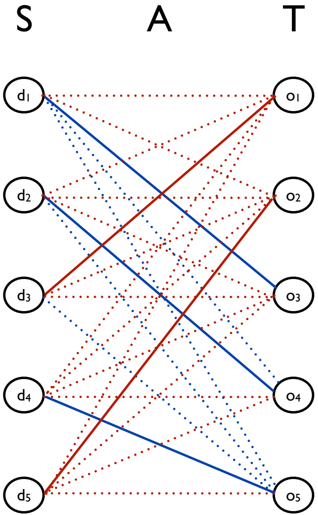


Single Depot VS: Matching

Example of solution:

- $C_1 = \{v_1, v_3\}$
- $C_2 = \{v_2, v_4, v_5\}$

v_i	t_i	a_i	o_i	d_i
v_1	7:10	7:30	T_a	T_b
v_2	7:20	7:40	T_c	T_d
v_3	7:40	8:05	T_b	T_a
v_4	8:00	8:30	T_d	T_c
v_5	8:35	9:05	T_c	T_d



Min Cost Flow: LP formulation

- $N = S \cup T \cup \{s, t\}$
- $A = A_1 \cup \{(s, i) | i \in S\} \cup \{(t, i) | i \in T\} \cup \{(t, s)\}$
- $b_i = \begin{cases} +1 & \text{if } i \in S \\ -1 & \text{if } i \in T \\ 0 & \text{otherwise} \end{cases}$

$$\min \quad \sum_{ij \in A} c_{ij} x_{ij} \tag{14}$$

$$\text{s.t.} \quad \sum_{ij \in A} x_{ij} - \sum_{ji \in A} x_{ji} = b_i \qquad \forall i \in N \tag{15}$$

$$x_{ts} \leq k \tag{16}$$

$$x_{ij} \leq 1 \qquad \forall ij \in A \setminus \{t, s\} \tag{17}$$

$$x_{ij} \geq 0 \qquad \forall ij \in A \tag{18}$$

Capacitated Single Depot VS: Questions?

Matching and Min Cost Flow: which is the difference in term of graph sizes?

What if the vehicles are located in different depots?

What if there is a single depot, but the vehicles have different types, and hence different operational costs?

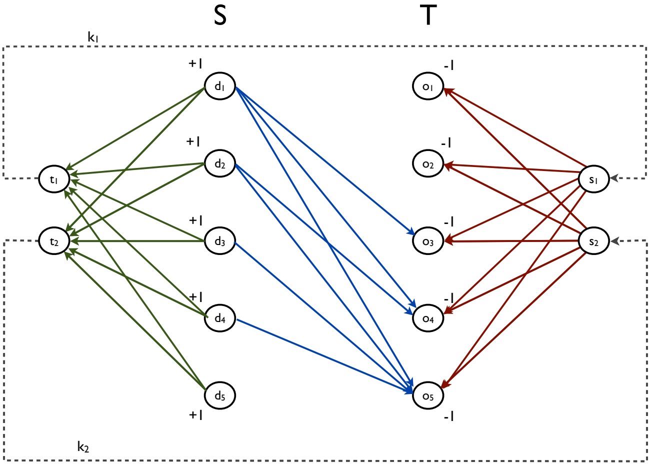
Multi Depot Vehicle Scheduling

Real life: Société de Transport de Montreal [HMS2006]

- 665 Bus Lines
- 7 Depots, capacities between 130 and 250
- 17.037 trips

Multi Depot Vehicle Scheduling

Let D be the set of depots, and let k_h be the capacity of depot h. For each depot h we introduce the pair $\{s^h, t^h\}$.



Multi Depot Vehicle Scheduling: First Formulation

- $N = S \cup T \cup \{\{s^h, t^h\} \mid h \in D\}$
- $A = A_1 \cup \{(t^h, s^h), h \in D\} \cup$
  $\{(s^h, i) \mid i \in S, h \in D\} \cup \{(t^h, i) \mid i \in T, h \in D\}$
- $b_i = \begin{cases} +1 & \text{if } i \in S \\ -1 & \text{if } i \in T \\ 0 & \text{otherwise} \end{cases}$

$$\min \sum_{ij \in A} c_{ij} x_{ij} \tag{19}$$

$$\text{s.t.} \quad \sum_{ij \in A} x_{ij} - \sum_{ji \in A} x_{ji} = b_i \qquad \forall i \in N \tag{20}$$

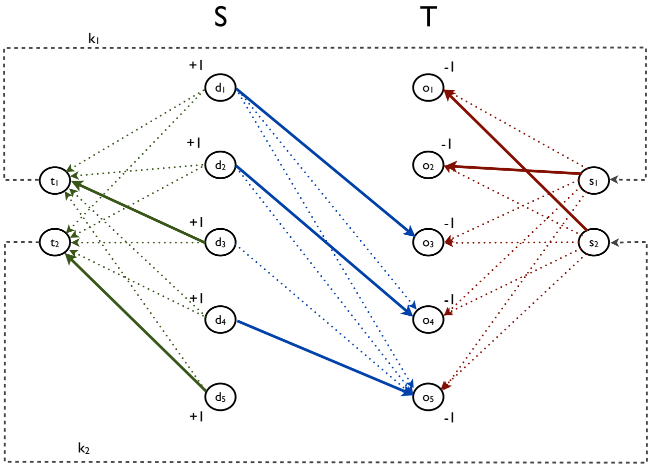
$$x_{t^h s^h} \le k_h \qquad \forall h \in D \tag{21}$$

$$x_{ij} \le 1 \qquad \forall ij \in A \setminus \{\{t^h, s^h\}, \forall h \in D\} \tag{22}$$

$$x_{ij} \ge 0 \qquad \forall ij \in A \tag{23}$$

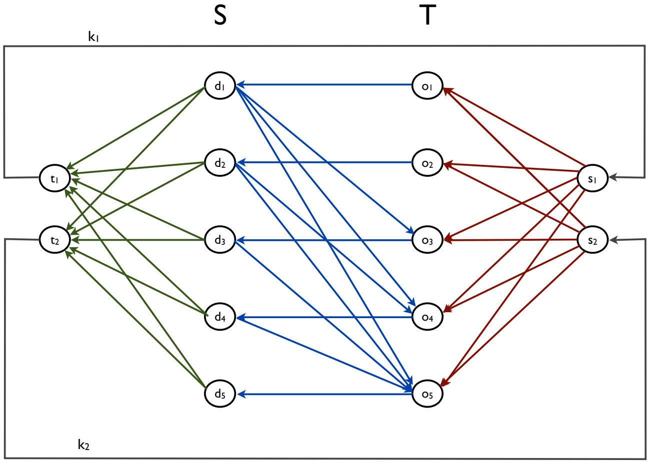
Multi Depot Vehicle Scheduling

Does each vehicle return to the origin depot?



Min Cost Flow: ILP formulation

- $N = S \cup T \cup \{\{s^h, t^h\} \mid h \in D\}$
- $A = \bigcup_{h \in D} \{A_1 \cup \{(s^h, o_i), (o_i, d_i), (d_i, t^h) \mid i \in V\} \cup \{(t^h, s^h)\}\}$



Min Cost Flow: ILP formulation

- $N = S \cup T \cup \{\{s^h, t^h\} \mid h \in D\}$
- $A = \bigcup_{h \in D} \{A_1 \cup \{(s^h, o_i), (o_i, d_i), (d_i, t^h) \mid i \in V\} \cup \{(t^h, s^h)\}\}$

$$
\begin{aligned}
(MDVS) \quad \min \quad & \sum_{h \in D} \sum_{ij \in A} c_{ij}^h x_{ij}^h && && (24)\\
\text{s.t.} \quad & \sum_{h \in D} \sum_{ij \in A} x_{ij}^h = 1 && \forall i \in S && (25)\\
& \sum_{ij \in A} x_{ij}^h - \sum_{ji \in A} x_{ji}^h = 0 && \forall i \in N, \forall h \in D && (26)\\
& x_{ts}^h \leq k_h && \forall h \in D && (27)\\
& x_{ij}^h \in \{0, 1\} && \forall h \in D, \forall ij \in A \setminus \{s^h, t^h\} && (28)
\end{aligned}
$$

Min Cost Flow: LP relaxation

- $N = S \cup T \cup \{\{s^h, t^h\} \mid h \in D\}$
- $A = \bigcup_{h \in D}\{A_1 \cup \{(s^h, o_i), (o_i, d_i), (d_i, t^h) \mid i \in V\} \cup \{(t^h, s^h)\}\}$

$$
\begin{aligned}
\min \quad & \sum_{h \in D} \sum_{ij \in A} c_{ij}^h x_{ij}^h \\
\text{s.t.} \quad & \sum_{h \in D} \sum_{ij \in A} x_{ij}^h = 1 \qquad \forall i \in S \\
& \sum_{ij \in A} x_{ij}^h - \sum_{ji \in A} x_{ji}^h = 0 \qquad \forall i \in N, \forall h \in D \\
& x_{ts}^h \leq k_h \qquad \forall h \in D \\
& {\color{red} 0 \leq x_{ij}^h \leq 1} \qquad \forall h \in D, \forall ij \in A \setminus \{s^h, t^h\}
\end{aligned}
$$

Lagrangian Relaxation

We keep the integrality constraint, but
we relax the assignment constraint:

$$z_{LB} = \Phi(\lambda) = \min \sum_{h \in D} \sum_{ij \in A} c_{ij}^h x_{ij}^h - \sum_{i \in S} \lambda_i \left(\sum_{h \in D} \sum_{ij \in A} x_{ij}^h - 1 \right) \quad (29)$$

$$\text{s.t.} \sum_{ij \in A} x_{ij}^h - \sum_{ji \in A} x_{ji}^h = 0 \qquad \forall i \in N, \forall h \in D \quad (30)$$

$$x_{ts}^h \leq k_h \quad (31)$$

$$x_{ij}^h \in \{0, 1\} \qquad \forall ij \in A \quad (32)$$

Lagrangian Relaxation

$$\begin{aligned}
\Phi(\lambda) = \sum_{i \in S} \lambda_i + \min \quad & \sum_{h \in D} \left(\sum_{ij \in A} (c_{ij}^h - \lambda_i) x_{ij}^h \right) \\
\text{s.t.} \quad & \sum_{ij \in A} x_{ij}^h - \sum_{ji \in A} x_{ji}^h = 0 \qquad \forall i \in N, \forall h \in D \\
& x_{ts}^h \leq k_h \\
& x_{ij}^h \in \{0,1\} \qquad \forall ij \in A \setminus \{(t^h, s^h)\}
\end{aligned}$$

We get $|D|$ independent subproblems that can be solved using any Min Cost Flow algorithms.

Remark: $\Phi(\lambda)$ yields a lower bound for each value of λ ...

Lagrangian Relaxation

$$\Phi_h(\lambda) = \min \quad \sum_{ij \in A} (c_{ij}^h - \lambda_i) x_{ij}^h \tag{33}$$

$$\text{s.t.} \quad \sum_{ij \in A} x_{ij}^h - \sum_{ji \in A} x_{ji}^h = 0 \qquad \forall i \in N \tag{34}$$

$$x_{ts}^h \leq k_h \tag{35}$$

$$x_{ij}^h \in \{0, 1\} \qquad \forall ij \in A \setminus \{(t^h, s^h)\} \tag{36}$$

We get $|D|$ independent subproblems that can be solved using any Min Cost Flow algorithms.

Lagrangian Relaxation

$$\Phi_h(\lambda) = \min \quad \sum_{ij \in A} (c_{ij}^h - \lambda_i) x_{ij}^h \tag{37}$$

$$\text{s.t.} \quad \sum_{ij \in A} x_{ij}^h - \sum_{ji \in A} x_{ji}^h = 0 \qquad \forall i \in N \tag{38}$$

$$x_{ts}^h \leq k_h \tag{39}$$

$$0 \leq x_{ij}^h \leq 1 \qquad \forall ij \in A \tag{40}$$

Min Cost Flow problems are Totally Unimodular

MD-VS: Subgradient Optimization

Among all vector λ, we look for the vector that solves:

$$\max_{\lambda} \Phi(\lambda) = \sum_{i \in S} \lambda_i + \max_{\lambda} \sum_{h \in D} \Phi_h(\lambda)$$

Since $\Phi(\lambda)$ is a concave piecewise linear function, this optimization problem can be solved with a subgradient algorithm.

Core idea:

$$\lambda^{k+1} \leftarrow \lambda^k + T\, g$$

where

- T is a scalar (step size)
- g is a search direction (subgradient)

MD-VS: Subgradient Optimization

Algorithm 1: Subgradient

$\lambda_i^0 \leftarrow 0$ (init multipliers);
foreach $k = 1, \ldots, maxiter$ **do**
  foreach $h \in D$ **do**
    Solve $\Phi_h(\lambda)$ and get $\bar{x}_{ij}^h$ and z_{LB}^h;
  Compute $z_{LB} = \sum_{i \in S} \lambda_i + \sum_{h \in D} z_{LB}^h$;
  If $z_{LB} > z_{LB}^*$ then $z_{LB}^* \leftarrow z_{LB}$;
  If $\bar{x}_{ij}^h$ is feasible for (24)–(28) update z_{UB};
  If $z_{LB}^* = z_{UB}$: **stop** z_{UB} is the optimal solution;
  Update subgradients $g_i = 1 - \sum_{h \in D} \sum_{ij \in A} \bar{x}_{ij}^h$ for all $i \in S$;
  Update step size $T = \frac{f(z_{UB} - z_{LB})}{\sum_{i \in S} g_i^2}$;
  Update multipliers $\lambda_i^{k+1} = \lambda_i^k + T\, g_i$ for all $i \in S$;

MD-VS: Lagrangian-based Heuristic

Once we solve $\max_\lambda \Phi(\lambda)$, we consider:

- $Q_1 = \{i \mid \sum_{h \in D} \sum_{ij \in A} \bar{x}_{ij}^h > 1\}$ (trips overassigned)

  We empty Q_1 (easy)

- $Q_2 = \{i \mid \sum_{h \in D} \sum_{ij \in A} \bar{x}_{ij}^h = 0\}$ (trips unassigned)

  We *try to empty* Q_2 (capacity constraint must still hold!)

If we are not able to empty Q_2, we solve a **Minimum Fleet Size** problem with the trips in Q_2 and assign greedly the resulting vehicle duties to the *free* depots.

MD-VS: Disjoint Path Cover Formulation

Yet Another Formulation and Yet Another Graph!

Consider the multigraph $G = (N, A)$ where:

- N has a vertex for each trip v_i with $i = 1..n$, and a pair of vertices s_h and t_h for each depot h (in total $n + 2|D|$ vertices)
- there is a pair of arcs (s_h, v_i) and (v_i, t_h) for each trip and each depot
- there is an arc $(v_i, v_j)^h$ for each pair of compatible trips and each depot (i.e. $|D|$ parellel arcs)

A path from s_h to t_h corresponds to a feasible vehicle duty assigned to a vehicle housed in depot h.

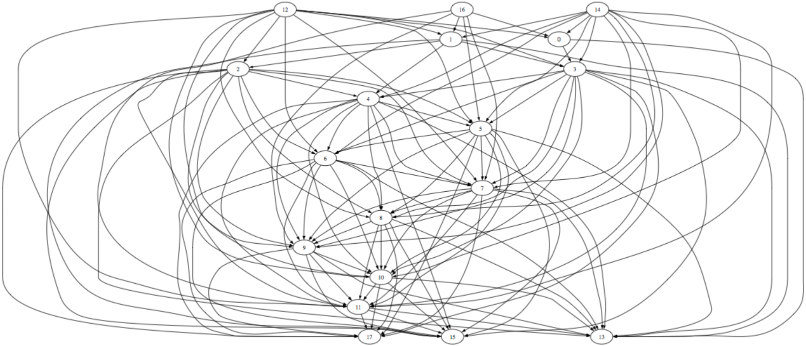
Example

Given 3 depots and 12 trips:

ID	Da	A	Inizio	Fine
0	NETTPO	RMANAG	04:30	06:20
1	NETTPO	RMLAUREN	04:40	06:20
2	RMLAUREN	NETTPO	06:20	08:15
3	APRILI	LATINA	07:25	08:05
4	ANZICO	NETTPO	13:00	13:40
5	NETTPO	ANZIO	14:00	14:25
6	ANZIO	NETTPO	14:30	14:50
7	NETTPO	ANZIO	14:50	15:20
8	ANZIO	NETTPO	15:30	16:00
9	NETTPO	ANZIO	16:00	16:20
10	ANZIO	NETTPO	16:30	16:55
11	NETTPO	ANZIO	17:30	18:00

Example

Given 3 depots and 12 trips:



MD-VS: Multicommodity Formulation

$$\min \quad \sum_{ij\in A} \sum_{h\in} c_{ij}^h x_{ij}^h \tag{41}$$

$$\text{s.t.} \quad \sum_{h\in D} \sum_{ij\in A} x_{ij}^h = 1 \qquad \forall i \in V \tag{42}$$

$$\sum_{ji\in A} x_{ji}^h - \sum_{ij\in A} x_{ij}^h = 0 \qquad \forall h \in D, i \in V \tag{43}$$

$$\sum_{j\in V} x_{s_h,j}^h \le k_h \qquad \forall h \in D \tag{44}$$

$$x_{ij}^h \in \{0,1\} \qquad \forall (i,j) \in A, h \in D. \tag{45}$$

Drawback: still huge number of variables and constraints!

MD-VS: Path-based Formulation

Given the set of every path $\mathcal{P}$, let $a_{ip} = 1$ iff trip i is covered by p, and let b_p^h iff path p starts (and ends) at depot h

Set Partitioning formulation:

$$\min \sum_{p \in \mathcal{P}} c_p \lambda_p \tag{46}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}} a_{ip} \lambda_p = 1 \qquad \forall i \in V \tag{47}$$

$$\sum_{p \in \mathcal{P}} b_p^h \lambda_p \leq k_h \qquad \forall h \in D \tag{48}$$

$$\lambda_p \in \{0, 1\} \qquad \forall p \in \mathcal{P}. \tag{49}$$

This is solved by Column Generation!

MD-VS: Column Generation and Pricing Subproblem

Start with $\bar{\mathcal{P}} \subset \mathcal{P}$ and generate new paths on demand

$$\min \sum_{p \in \bar{\mathcal{P}}} c_p \lambda_p \tag{50}$$

$$\text{dual multipliers } \alpha_i \leftarrow \sum_{p \in \bar{\mathcal{P}}} a_{ip} \lambda_p = 1 \qquad \forall i \in V \tag{51}$$

$$\text{dual multipliers } \beta_h \leftarrow \sum_{p \in \bar{\mathcal{P}}} b_p^h \lambda_p \leq k_h \qquad \forall h \in D \tag{52}$$

$$\lambda_p \geq 0 \qquad \forall p \in \bar{\mathcal{P}}. \tag{53}$$

Given α_i^* and β_h^*, set the reduced cost on the arcs

- $\bar{c}_{ij}^h = c_{ij}^h - \alpha_i$ for $i = 1..n$
- $\bar{c}_{ij}^h = c_{ij}^h - \beta_h$ for $i = t_h,\ h \in D$

(recall: $c_p^h = \sum_{ij \in A} c_{ij}^h$)

MD-VS: Pricing Subproblem

The pricing subproblem is a shortest path problem:

$$z_{rc} = \min \quad \sum_{ij \in A} \sum_{h \in D} \bar{c}_{ij}^h x_{ij}^h \tag{54}$$

$$\text{s.t.} \quad \sum_{h \in D} \sum_{(s_h, i) \in A} x_{s_h, i}^h = 1 \tag{55}$$

$$\sum_{ji \in A} x_{ji}^h - \sum_{ij \in A} x_{ij}^h = 0 \qquad \forall h \in D, i \in V \tag{56}$$

$$0 \leq x_{ij}^h \leq 1 \qquad \forall (i,j) \in A, h \in D. \tag{57}$$

which is **separable** by depot

If a path $p \notin \bar{\mathcal{P}}$ with $z_{rc} < 0$ exists, then:

$$\bar{\mathcal{P}} \leftarrow \{p\} \cup \bar{\mathcal{P}}$$

Problem (50)–(53) is solved anew, and the algorithm iterates

MD-VS: Column Generation

One drawback of column generation is that becomes less efficient as the average number of trips per path increases.

In real life instances there is not a take-all winner algorithm