

Vorlesung

# Netzalgorithmen

Prof. Dr.-Ing Günter Schäfer

Vorlesungsmitschrift von  
ADRIAN SCHOLLMAYER

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>3</b>
<b>2</b>	<b>Modeling Network Design Problems</b>	<b>4</b>
2.1	Modeling Design Problems in Link-Path Formulation . . . . .	4
2.2	Modeling Design Problems in Node-Link Formulation . . . . .	5
2.3	Link-Demand-Path-Identifier-Based Notation . . . . .	5
2.4	Shortest-Path Routing . . . . .	7
3.1	Extreme Points and Basic Solutions . . . . .	8
3.1.1	Phase 2 . . . . .	8
3.2	The Simplex-Algorithm . . . . .	8
3.2.1	Phase 1 . . . . .	8
3.2.2	Simplex Tableaus . . . . .	9
	<b>Stichwortverzeichnis</b>	<b>11</b>

# 1 Einführung

## 2 Modeling Network Design Problems

### 2.1 Modeling Design Problems in Link-Path Formulation

Preliminary notation for undirected demands:

- $x - y$  ...Link between node  $x$  and  $y$
- $\hat{h}_{xy} = b$  ...Demand  $d$  between nodes  $x$  and  $y$  (i. e. Demand  $b$  bandwidth between nodes  $x$  and  $y$ )
- $\hat{x}_{v_1 v_2} = f$  ...Flow on the path between  $v_1$  and  $v_2$
- $\hat{x}_{v_1 v_2 v_3} = f$  ...Flow on the path  $v_1, v_2, v_3$
- $\hat{c}_{v_1 v_2}$  ...Capacity of the link between  $v_1$  and  $v_2$

This generally yields equations for satisfying the demands and inequalities for limiting the bandwidth to the link capacities, e. g.

$$\hat{x}_{12} + \hat{x}_{132} = 5 \quad (2.1)$$

$$\hat{x}_{13} + \hat{x}_{123} = 7 \quad (2.2)$$

$$\vdots \quad (2.3)$$

$$\hat{x}_{132} + \hat{x}_{13} + \hat{x}_{213} \leq 10 \quad (2.4)$$

$$\hat{x}_{132} + \hat{x}_{123} + \hat{x}_{23} \leq 15 \quad (2.5)$$

As such a system usually has multiple solutions, an **objective function** is defined, which can then be optimized, e. g.:

$$F = \hat{x}_{12} + 2\hat{x}_{132} + \hat{x}_{13} + 2\hat{x}_{123} + \hat{x}_{23} + 2\hat{x}_{213} \quad (2.6)$$

In this example, flow routed over two links are weighted with factor two, so direct links are preferred.

Such an optimization is called a **multi-commodity flow problem**. The *optimal solution* to this problem is marked with an asterisk, e. g.:

$$\hat{x}_{12}^* = 5 \quad \hat{x}_{13}^* = 7 \quad \hat{x}_{23}^* = 8 \quad (2.7)$$

Important observations:

- Changing the objective function usually affects the optimal solution to a problem.
- Formulation a good objective function for the particular network is important for obtaining meaningful solutions.

## 2.2 Modeling Design Problems in Node-Link Formulation

In this section, links and demands are assumed to be directed. Undirected links  $x - y$  are replaced with two directed links  $x \rightarrow y$  and  $y \rightarrow x$ .

Notation:

- $v_1 \rightarrow v_2$  ...Directed link from node  $v_1$  to node  $v_2$
- $\langle v_1 : v_2 \rangle$  ...Demand from node  $v_1$  to node  $v_2$
- $\tilde{x}_{a,d}$  ...Flow over arc  $a$  for demand  $d$  (e.g.  $\tilde{x}_{v_1 v_2, v_1 v_2}$  flow over arc  $v_1 \rightarrow v_2$  for demand  $\langle v_1 : v_2 \rangle$ )

Backflows (e.g.  $\tilde{x}_{21,12}$ ) are also possible, but make practically no sense, so they are set to 0.

## 2.3 Link-Demand-Path-Identifier-Based Notation

Demands and links are assigned label indexes. Thus, tables for mapping indexes to actual demands and links are required.

Notation:

- $h_i$  ...Demand with index  $i$
- $c_e$  ...(Known) capacity of link  $e$
- $y_e$  ...*Unknown* capacity of link  $e$
- $P_i$  ...Number of candidate paths for demand  $i$
- $P_{ij}$  ... $j$ th candidate path for demand  $i$
- $(v_1, e_1, v_2, e_2, \dots, e_n, v_{n+1})$  ... $n$ -hop path
- $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_{n+1}$  ...node representation of a path (directed, use  $-$  instead of  $\rightarrow$  for undirected)
- $\{e_1, e_2, \dots, e_n\}$  ...link representation of undirected paths
- $(e_1, e_2, \dots, e_n)$  ...link representation of directed paths
- $v$  ...Node
- $e$  ...Link
- $d$  ...Demand
- $p$  ...Path
- $V, E, D, P$  ...total numbers of the aforementioned items

- $\xi_e$  ...Cost of a link  $e$

Example equations:

$$x_{11} = 15 \quad (2.8)$$

$$x_{21} + x_{22} = 20 \quad (2.9)$$

$$x_{31} + x_{32} = 10 \quad (2.10)$$

Demands:

$$\sum_{p=1}^{P_d} x_{dp} = h_d \quad d = 1, \dots, D \quad (2.11)$$

Short form, when iterating over all candidate paths:

$$\sum_p x_{dp} = h_d \quad d = 1, \dots, D \quad (2.12)$$

The vector of all flows (path flow variables) is called the **flow allocation vector** or short **flow vector**:

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_D) \quad (2.13)$$

$$= (x_{11}, x_{12}, \dots, x_{1P_1}, \dots, x_{D1}, x_{D2}, \dots, x_{DP_D}) \quad (2.14)$$

$$= (x_{dp} : d = 1, 2, \dots, D; p = 1, 2, \dots, P_d) \quad (2.15)$$

Important: vectors are represented with bold letters  $\mathbf{x}$  and scalar values are represented with normal letters  $x$ .

The relationship between links and paths is written down with the **link-path incidence relation**  $\delta_{edp}$ :

$$\delta_{edp} = \begin{cases} 1 & \text{if link } e \text{ belongs to path } p \text{ for demand } d \\ 0 & \text{otherwise} \end{cases} \quad (2.16)$$

$\delta_{edp}$  can be written as a table:

$e$	$P_{11} = \{2, 4\}$	$P_{21} = \{5\}$	$P_{22} = \{3, 4\}$
1	0	0	0
2	1	0	0
3	0	0	1
4	1	0	1
5	0	1	0

The **load** in link  $e$  can be written as:

$$\underline{y}_e = \underline{y}_e(\mathbf{x}) = \sum_{d=1}^D \sum_{p=1}^{P_d} \delta_{edp} x_{dp} \quad (2.17)$$

*Important:* The actual link loads  $\underline{y}_e$  are determined by the path flow variables  $x_{dp}$  of a solution and are not the same as the link capacity variables  $y_e$ .

Cost of a path  $P_{dp}$ :

$$\zeta_{dp} = \sum_e \delta_{edp} \xi_e, \quad d = 1, 2, \dots, D \quad p = 1, 2, \dots, P_d \quad (2.18)$$

**Shortest-Path Allocation Rule for Dimensioning Problems** For each demand, allocate its entire demand to its shortest path with respect to link costs and candidate paths. If there is more than one shortest path for a given demand, then the demand volume can be arbitrarily split among the shortest paths.

## 2.4 Shortest-Path Routing

For shortest-path routing, demands will only be routed on their shortest paths. The path length is determined by adding up link costs  $w_e$  according to some weight system  $\mathbf{w} = (w_1, w_2, \dots, w_E)$ .

**Single Shortest Path Allocation Problem** For given link capacities  $\mathbf{c}$  and demand volumes  $\mathbf{h}$ , find a link weight system  $\mathbf{w}$  such that the resulting shortest paths are unique and the resulting flow allocation vector is feasible.

Zusammenfassung  
fortführen

# 3

## 3.1 Extreme Points and Basic Solutions

### 3.1.1 Phase 2

$$S(x) \text{ linear unabhängig, wenn } \sum_{i=1}^k \alpha_i \vec{x}_i = 0 \implies \forall i \in \{1, \dots, k\} : \alpha_i = 0 \quad (3.1)$$

## 3.2 The Simplex-Algorithm

Note: Solving the system of equations means transforming the target vector into the basis of the input vector.

If all  $d_i$  are non-negative, the basic solution ist the optimal solution as the  $d_i y_i$  are always subtracted and  $y_i$  is always non-negative, thus positive  $d_i$  can only decrease the result and not increase it.

$\vec{z}_k$  is non-negative for all three cases as  $t_{k,j}$  is non-positive and  $\alpha > 0$ . For items that are not  $j$  and not part of the base, the result is 0 and thus it can be removed from the sum.

If we find that all  $d_j$  are negative, the objective function is unbounded from above.

In case 3,  $\varepsilon$  is always positive due to the way it is defined and the constrains on the values in this case. As  $\varepsilon$  is the minimum of the given set, it can not be  $< 0$ .

Thus, the resulting vector  $\vec{z}$  can be in the feasible set.

Basically, we have walked from one corner to the next and restarted the algorithm. In each calculation, we display  $\vec{b}$  in another basis. Thus, we need to watch out for cycling between multiple results. This is what Bland's rule is used for.

### 3.2.1 Phase 1

For distinction, a new variable vector  $\vec{y}$  is introduced. This is later transformed by extending  $\vec{x}$ . All  $y_i$  are weighted 1.

This auxiliary problem can be solved by setting  $\vec{x} = \vec{0}$ .

Note that the auxiliary problem is to minimize, not maximize the result. As all  $y_i$  need to remain positive, the algorithm will approach  $\forall i : y_i = 0$  eventually.



### 3.2.2 Simplex Tableaus

Page 35 last line:  $\vec{\hat{b}}$  is actually  $\vec{b}$ ...

Example 1:

$$\max x_1 + x_2 \tag{3.2}$$

such that:

$$-x_1 + x_2 \leq 1 \tag{3.3}$$

$$x_1 \leq 3 \tag{3.4}$$

$$x_2 \leq 2 \tag{3.5}$$

$$x_1, x_2 \geq 0 \tag{3.6}$$

This needs to be transformed from canonical form into the standard form. For this, additional variables  $x_3, x_4, \dots$  need to be introduced:

$$-x_1 + x_2 + x_3 = 1 \tag{3.7}$$

$$x_1 + x_4 = 3 \tag{3.8}$$

$$x_2 + x_5 = 2 \tag{3.9}$$

$$\vec{x} \geq \vec{0} \tag{3.10}$$

A basic solution can be obtained directly:

$$x_1 = 0 \tag{3.11}$$

$$x_2 = 0 \tag{3.12}$$

$$x_3 = 1 \tag{3.13}$$

$$x_4 = 3 \tag{3.14}$$

$$x_5 = 2 \tag{3.15}$$

From this, the current maximum value is  $z = 0$ .

We now work with the simplex tableau.

$$\begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & b \\ -1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{array} \tag{3.16}$$

The last row represents the objective function and the last cell if the last row is  $z$ .

This matrix can now be transformed using Gauss-Jordan elimination. During elimination, we need to ensure that  $\vec{b}$  remains positive!

$$\begin{array}{cccccc}
 -1 & 1 & 1 & 0 & 0 & 1 & \leftarrow + \\
 1 & 0 & 0 & 1 & 0 & 3 & \leftarrow 1 \\
 0 & 1 & 0 & 0 & 1 & 2 & \\
 1 & 1 & 0 & 0 & 0 & 0 & \leftarrow + \\
 & & & & & & \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} -1
 \end{array} \tag{3.17}$$

$$\begin{array}{cccccc}
 0 & 1 & 1 & 1 & 0 & 4 & \leftarrow + \\
 1 & 0 & 0 & 1 & 0 & 3 & \\
 0 & 1 & 0 & 0 & 1 & 2 & \leftarrow -1 \\
 0 & 1 & 0 & -1 & 0 & -3 & \leftarrow + \\
 & & & & & & \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} -1
 \end{array} \tag{3.18}$$

Row 1 can not be subtracted from row 3 as it would result in a negative  $b_3$ . We can, however, subtract row 3 from row 1. We obtain a matrix where all coefficients in the last row are negative. Thus, we have obtained an optimal solution and  $-z$  is in the last cell of the matrix.

$$\begin{array}{cccccc}
 0 & 0 & 1 & 1 & -1 & 2 \\
 1 & 0 & 0 & 1 & 0 & 3 \\
 0 & 1 & 0 & 0 & 1 & 2 \\
 0 & 0 & 0 & -1 & -1 & -5
 \end{array} \tag{3.19}$$

Thus, we obtain:

$$\vec{x} = \begin{pmatrix} 3 \\ 2 \\ 2 \\ 0 \\ 0 \end{pmatrix} \tag{3.20}$$

This solution indeed satisfies all constraints.

# Stichwortverzeichnis

capacity, [4](#), [5](#)

cost, [6](#), [7](#)

demand, [5](#)

    directed, [5](#)

    undirected, [4](#)

flow, [4](#), [5](#)

flow allocation vector, [6](#)

flow vector, [6](#)

Link

    undirected, [4](#)

link, [5](#)

    directed, [5](#)

link-path incidence relation, [6](#)

load, [6](#)

multi-commodity flow problem, [4](#)

node, [5](#)

objective, [4](#)

path, [5](#)

$n$ -hop, [5](#)

    candidate, [5](#)

simplex tableau, [9](#)